Numerical Estimation of Surface Parameters by Level Set Methods

Ashok Kumar Vaikuntam

Vom Fachberich Mathematik der Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften (Doctor rerum naturalium,
Dr. rer. nat.) genehmigte Dissertation.

1. Gutachter: Prof. Dr. Axel Klar
2. Gutachter: Prof. Dr. David Adalsteinsson

Vollzug der Promotion: 14 February 2008

D 386

Acknowledgment

I would like to express my gratitude to my supervisor Prof. Dr. Axel Klar for his patience and cooperation during the course of my research work. I am thankful to Prof. Dr. Adalsteinsson, North Carolina Sate University, for his suggestions and critical remarks while reviewing my thesis. I am extremely grateful to Prof. Dr. H. Neunzert for giving my an opportunity to work in ITWM.

I am deeply indebted to my guide Dr. Andreas Wiegmann, who introduced the wonderful world of "Level set " after my arrival at ITWM. I am thankful to his guidance and stimulating discussions. My sincere thanks to my Head of the Department Dr. Steiner for constant encouragement and suggestions.

I would like to thank The Graduate School, Department of Mathematics, TU- Kaiserslautern for funding my research studies towards doctoral degree.

I am highly indebted to Dr. Stefan Rief for his critical remarks during final version of the thesis. I am thankful to Dr. Jürgen Becker, Dr. Guido Thömmes, Dr. Ettrich, and Dipl.-Math Inga Shklyar for helping me in simulations.

I am thankful to my wife Neeraja who gave me inspiration and encouragement during the course of my research.

Finally I thank all members of my group, administrative and SLG staff of ITWM.

Abstract

A modular level set algorithm is developed to study the interface and its movement for free moving boundary problems. The algorithm is divided into three basic modules *initialization*, *propagation* and *contouring*. Initialization is the process of finding the signed distance function Φ from closed objects. We discuss here, a methodology to find an accurate Φ from a closed, simply connected surface discretized by triangulation. We compute Φ using the direct method and it is stored efficiently in the neighborhood of the interface by a narrow band level set method. A novel approach is employed to determine the correct sign of the distance function at convex-concave junctions of the surface. The accuracy and convergence of the method with respect to the surface resolution is studied. It is shown that the efficient organization of surface and narrow band data structures enables the solution of large industrial problems. We also compare the accuracy of Φ by direct approach with Fast Marching Method (FMM). It is found that the direct approach is more accurate than FMM.

Contouring is performed through a variant of the marching cube algorithm used for the isosurface construction from volumetric data sets. The algorithm is designed to keep foreground and background information consistent, contrary to the neutrality principle followed for surface rendering in computer graphics. The algorithm ensures that the isosurface triangulation is closed, non-degenerate and non-ambiguous. The constructed triangulation has desirable properties required for the generation of good volume meshes. These volume meshes are used in the boundary element method for the study of linear electrostatics.

For estimating surface properties like interface position, normal and curvature accurately from a discrete level set function, a method based on higher order weighted least squares is developed. It is found that least squares approach is more accurate than finite difference approximation. Furthermore, the method of least squares requires a more compact stencil than those of finite difference schemes. The accuracy and convergence of the method depends on the surface resolution and the discrete mesh width.

This approach is used in propagation for the study of mean curvature flow and bubble dynamics. The advantage of this approach is

that the curvature is not discretized explicitly on the grid and is estimated on the interface. The method of constant velocity extension is employed for the propagation of the interface. With least squares approach, the mean curvature flow has considerable reduction in mass loss compared to finite difference techniques.

In the bubble dynamics, the modules are used for the study of a bubble under the influence of surface tension forces to validate Young-Laplace law. It is found that the order of curvature estimation plays a crucial role for calculating accurate pressure difference between inside and outside of the bubble. Further, we study the coalescence of two bubbles under surface tension force. The application of these modules to various industrial problems is discussed.

Contents

Chapter 1. Introduction 1
1.1. Classification of different methods 1
1.2. Level set methods 3
1.3. Estimation of surface properties 7
1.4. Outline of the thesis 8

Chapter 2. Computation of signed distance functions from surface triangulations 11
2.1. Surface data structure 11
2.2. Narrow band data structure 17
2.3. Initialization 21
2.4. Results 29
2.5. Applications 33

Chapter 3. Marching cube algorithm for isosurface construction 39
3.1. Introduction 39
3.2. Definitions and Conventions 41
3.3. Extension of topological cases 44
3.4. Implementation 45
3.5. Quality of the surface triangulation 47
3.6. Limitation due to resolution 49

Chapter 4. Higher order estimation of surface parameters 51
4.1. Least squares approach 52
4.2. Analytic test case 69

Chapter 5. Propagation - Application to moving interface problems 81
5.1. Estimation of F on grid points 81
5.2. Applications 82

Chapter 6. Summary and Conclusions 97

Future Aspects 98

Appendix A. Higher order finite difference scheme 103

Appendix. Bibliography 105

CHAPTER 1

Introduction

Interface modeling is a vital step in the study of the free surface of a moving boundary problem. These free surfaces are represented implicitly or explicitly in the simulations and propagated by a prescribed velocity given on the interface or on an underlying grid. The algorithm designed for the surface representation must be fast, memory efficient, and accurate. Apart from these desired properties, the algorithm must also be robust to handle topological changes like tearing, stretching and merging during surface evolution and should be generic for solving various applications from elasticity in solids or from two phase flows in fluid dynamics. During propagation, it may be also necessary to estimate surface parameters like normals and curvatures accurately on the interface.

1.1. Classification of different methods

There are different approaches used for the treatment of the interface. Commonly, these approaches can be classified into two main categories viz., *tracking methods* and *capturing methods*. We review these two methods briefly.

1.1.1. Tracking methods. In the tracking method the interface is tracked explicitly along the trajectories. It can be purely Lagrangian as in the boundary integral [**67**] and particle schemes [**54**]. In the Eulerian set-up it is further divided into *surface* and *volume* tracking methods. The surface tracking Eulerian method constructs the interface explicitly as a series of interpolated curves from discrete points. In the conventional front tracking algorithm, these interface points and connections are saved at each time step [**90**]. In the new front tracking approach these connections are saved as level curves instead of storing the connectivity information [**84**]. In the volume tracking Eulerian

method, the interface is reconstructed from cell by cell with marker particles, as in the classical marker-and-cell (MAC) approach [**36**]. These markers indicate the status of the cell. For example, in the simulation of viscous fluids with the free surface, this marker indicates the position of the fluid cell which are then moved with a prescribed velocity [**56**]. There are also other types of tracking methods which combine Lagrangian and Eulerian approaches. For example in the the partial moving mesh algorithm [**41**], some part of the grid is fixed and some part around the interface is moved in a Lagrangian way.

1.1.2. Capturing methods. In the capturing method, the interface is constructed from field values. These field values may be discontinuous variables like fluid fractions, or continuous zero level sets of some implicit function.

The algorithm which captures the interface from the discontinuous field of fluid fractions is referred to as volume of fluid (VOF) method. The original VOF model by Noh and Woodward uses the **S**imple **L**ine **I**nterface **C**onstruction (SLIC) [**60**]. This was later improved upon by Chorin [**16**] and Hirt and Nichols [**38**]. Youngs [**98**], [**99**] designed a **P**iecewise **L**inear **I**nterface **C**onstruction (PLIC) which was analyzed in detail by Pilliod [**66**]. Presently, there are many variations of VOF methods available in the literature. For details of different state-of-the-art VOF models see Pucket et al. [**68**], Rider and Kothe [**69**] and Scardovelli and Zaleski [**76**], [**77**].

The continuous representation on the other hand, captures the interface from the zero level sets of some implicit function. This is chiefly referred to as level set methods which was started by Osher and Sethian [**63**]. The power of this approach can be evidenced by widespread use of this method in the literature to variety of problems ever since its introduction. Mentioning here the different applications of this method is quite exhaustive. But to name a few, this method has been used and validated in the field of material science [**1**], [**2**], [**4**], fluid mechanics [**89**], [**11**], [**48**], image enhancement [**50**], image segmentation and vision [**51**], [**26**], [**30**], [**88**], [**100**], geometry and grid generation [**81**] and so on. The books of Sethian [**83**] and Osher and Fedkiw [**62**] gives a comprehensive overview and covers the application of this method to a variety of problems.

1.2. Level set methods

Both tracking and capturing methods have their own merits and de-merits and it is beyond the scope of this thesis to go into details of each method. In the numerical modeling, choosing a type or combination of methods is often a strategy for treating the interface. For our study, we require, a method which is robust with respect to changes of the interface structure and is designed in a modular way for various applications. In this context, *level set* methods are better suited for our study as it can easily handle complex topological changes of the interface during the evolution process. As we have discussed in the last section, it has also been proven to be very efficient in a wide variety of problems. Furthermore, the surface parameters like normals and curvatures can be estimated from the level set function in a straight forward way [**63**], [**83**], [**62**]. Before we go into the details of our objective, we describe here different steps involved in modeling the interface by a level set method.

In the level set approach, the treatment of the interface can be divided into three basic steps: (i) *initialization*, (ii) *propagation* and (iii) *contouring*. In the simulation, these three processes are performed iteratively and each is a research topic of its own as one can find a variety of techniques and algorithms in the literature.

1.2.1. Initialization. Given the surface in an implicit or in a discretized (say triangulated surface) form, the first step is to construct the level set function Ψ. The level set function can be a signed distance function which we denote by Φ.

Definition 1. *Signed distance function: Let Ω be a closed domain and $\Omega^- \subset \Omega \subset \mathbb{R}^3$ with piecewise smooth boundary Γ. Then the signed distance function Φ is defined by*

$$\Phi(\vec{x}) = \begin{cases} -d(\vec{x},\Gamma) & \vec{x} \in \Omega^-, \\ d(\vec{x},\Gamma) & \vec{x} \in \Omega \notin \Omega^-, \end{cases} \tag{1.1}$$

where the distance function $d(\vec{x},\Gamma)$ is given by

$$d(\vec{x},\Gamma) := \inf_{\vec{p}\in\Gamma} |\vec{x} - \vec{p}|, \tag{1.2}$$

as shown in Figure 1.1.

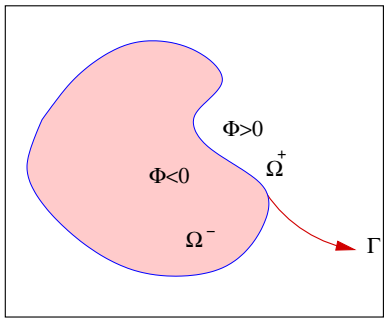
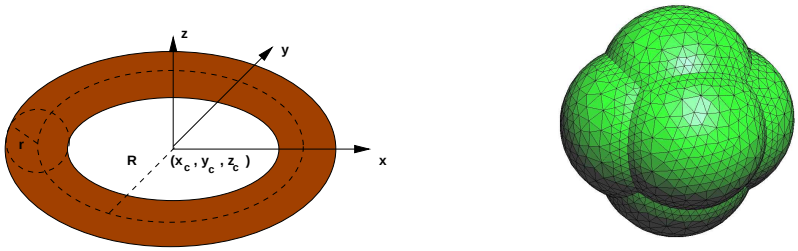


FIGURE 1.1. *Definition of the signed distance function* Φ.



(a) *(b)*

FIGURE 1.2. *Representation of (a) torus, and (b) initial seed for a dendrite crystal.*

The signed distance function for simple geometries can be obtained from an implicit representation. For instance, the implicit function of a torus with center (x_c, y_c, z_c) is

$$\Phi(x, y, z) = \sqrt{(a - R)^2 + (z - z_c)^2} - r = 0, \tag{1.3}$$

where $a = \sqrt{(x - x_c)^2 + (y - y_c)^2}$, r is the radius of the tube and R is the distance between the center of the tube to the center of the torus as shown in Figure 1.2(a). Similarly, for the study of dendrite crystal growth the initial seed Φ is given by

$$\Phi = min(\Phi_1, min(\Phi_2, min(\Phi_3, min(\Phi_4, min(\Phi_5, \Phi_6))))), \tag{1.4}$$

where $\Phi_1 \ldots \Phi_6$ are the implicit representations of six spheres of radius r with centers shifted by $r/2$ along six Cartesian directions as shown in Figure 1.2(b).

For a complex geometry, when the surface is not given by an implicit function, Φ is estimated from a discrete surface representation. In real life applications, most commonly surface triangulations are used.

There exists a variety of algorithms to compute Φ, which can be classified into two categories:

(i) *Direct methods*: The signed distance function is computed from the surface directly.

(ii) *PDE methods*: The signed distance function is found from a viscosity solution of the Eikonal equation, i.e.

$$|\nabla\Phi| - 1 = 0 \ \forall \ \vec{x} \in \Omega^- \cup \Omega^+ \text{ with } \ \Phi(\vec{x}) = 0 \ \forall \ \vec{x} \in \Gamma. \tag{1.5}$$

In practice, Φ is calculated by solving a time dependent pde [**89**]

$$\Phi_t(\vec{x}, t) = sgn(\Phi^0)(1 - |\nabla\Phi|) \text{ with } \Phi(\vec{x}, 0) = \Phi^0(\vec{x}), \tag{1.6}$$

where the zero level set Φ^0 represents the location of the interface. When this equation is solved for time t_c, then $\Phi(\vec{x}, t_c)$ is the signed distance for the points within the distance t_c from the interface as the speed is unity here. Sethian [**83**] employed a similar approach based on crossing times, by solving

$$\Phi_t + |\nabla\Phi| = 0, \tag{1.7}$$

forward and backward in time at a particular grid point from the initial estimate of Φ. To solve this equation efficiently, Adalsteinsson and Sethian proposed a Fast Marching Method (FMM) [**5**]. The initial estimate of Φ is computed from an implicit representation or by a direct method.

In the literature, category (ii) is preferred over (i), as (i) is slow and costly [**89**], [**5**]. On the other hand, direct computations can be improved by using a narrow band, also referred to as local level set methods [**5**], [**64**], where Φ is defined within a small region around the interface. In any case, if the given surface is complex and cannot be represented implicitly, then the signed distance function can only be obtained by a direct method from a discretized surface.

1.2.2. Propagation. From Φ and the velocity field on the discrete grid the normal velocity F is found on the interface and extended appropriately to the grid points. The interface is then propagated according to the level set equation [**63**], [**83**],

$$\Phi_t + F|\nabla\Phi| = 0. \tag{1.8}$$

Numerically, equation (1.8) is solved by explicit forward Euler stepping in time and by first order upwinding for $|\nabla\Phi|$. For high order approximations, methods like ENO [**37**] or WENO [**45**] for spatial derivatives and the Runge-Kutta approach for temporal discretization is used. The normal velocity F is found from the discrete velocity field and from the normal estimated by the level set function $\vec{n} = \frac{\nabla\Phi}{\|\nabla\Phi\|}$. In some applications, like in free moving boundary problems, the discrete velocity field is known only on one side of the interface. To estimate F on the other side, extrapolation based on upwinding information is used. Some of the commonly used approaches are constant velocity extensions [**5**], ghost fluid methods [**25**] and the method of characteristics [**6**].

1.2.3. Contouring: After propagating for finite time steps, it is desirable that Φ retains the property of signed distance function. The initialization of Φ to become a signed distance function again is usually referred to as *reinitialization*. There are two ways to perform this task.

(1) *Explicit methods:* In this method Φ is computed from the isosurface. This isosurface can be constructed from the zero level set, for instance, by a marching cube algorithm [**46**]. This is also referred to as *Contouring*.
(2) *Non-explicit methods:* In this method Φ is reinitialized to be a sign distance function by solving either equation (1.6) or equation (1.8) by a fast marching method, instead of constructing the isosurface explicitly.

The advantage of an explicit construction is that accuracy of Φ is maintained with respect to surface resolution, and the parameters needed for propagation, like projection points on the surface can be estimated accurately on the surface. On the other hand, constructing triangulations and reinitializing Φ by a direct method increases the computational time, especially when the narrow band is broad. In the non-explicit construction, the reinitialization is performed fast, and doesn't need a surface triangulation to compute Φ. However, estimating the projection point accurately on the surface is not straight forward. Moreover, it is found that improving the order of initial estimation of Φ proposed by Chopp [**14**], leads to a set of equations that are not trivial to solve [**62**].

1.3. Estimation of surface properties

Apart from these aforementioned three processes, certain surface properties, such as position, normal, and principal curvatures, and their directions, should be computable precisely on the interface from a discrete level set function. Applications, where knowledge of surface properties is necessary, are as follows:

(1) In the context of **E**xplicit-**J**ump **I**mmersed **I**nterface **M**ethods (EJIIM) [**94**], [**95**] for solving elliptic boundary value problems in a complicated geometry, the quality of the result depends crucially on the accuracy of the position and on the accuracy of the normal on the surface [**71**], [**72**], [**74**]. It is natural to look for the principal curvature directions rather than choosing arbitrary tangents at a particular point on the surface.

(2) To extract a surface triangulation from CAD data, an accurate estimate of position and curvature on the surface is desirable. The estimation is used for the construction of a better conformal triangulation on the surface. Extensive literature is now available on this topic. In [**55**], [**92**], [**65**], the triangulation of surfaces in the context of level set methods is discussed.

(3) In modeling the force on the interface in Stefan problems or in multi-phase flows, the surface tension balances the jump of the normal stress on the surface Γ [**35**], i.e.

$$[\sigma\vec{n}]_\Gamma \;=\; \tau H\vec{n}. \tag{1.9}$$

Here $\vec{n} = \vec{n}_\Gamma$ is the outer unit normal, τ is the surface tension coefficient, H is the curvature, and σ is the stress tensor on the interface. The accuracy of H is crucial for a good implementation of the jump condition.

(4) In surface diffusion flow, in metal reflow in semiconductor manufacturing, or in sintering and elastic membrane simulation, it is necessary to model the motion of curves and surfaces under the intrinsic Laplacian of curvature. The inherent difficulty in this method is the numerical estimation of 4th order derivatives in space, which is usually very unstable. Using finite differences for discretizing the higher order derivatives decreases accuracy [**15**].

1.4. Outline of the thesis

This dissertation discusses a level set algorithm for the treatment of interfaces which is particularly suited for estimating surface properties accurately. The algorithm is designed in a modular way to be applicable for various industrial problems. The objectives of this thesis are:

(1) To estimate the signed distance function accurately from a complicated geometry discretized by triangles. Moreover, the methodology of estimation must be fast and robust.
(2) Given one or more closed contours, the surfaces should be moved according to the velocity prescribed on the discrete grid. The velocity from the grid is extrapolated on the surface and the grid inherits the normal velocity from the interface.
(3) To construct an isosurface from the level set function using a variant of the marching cube algorithm. The resulting isosurface should be closed, non-degenerate and also account for non neutrality.
(4) To estimate accurately the interface position, normal, principal curvature and its direction from the level set function during the contouring process.
(5) To investigate the influence of curvature estimation for different applications like mean curvature flow and bubble dynamics.

The dissertation is structured exactly in accordance to these goals. The review of different existing methods for each module is explained in each chapter separately. Similarly the proposed new method/algorithm and their novel applications to industrial problems can be found in each chapter.

Chapter 2 deals with initialization. A detailed analysis of constructing the signed distance function especially from the discretized triangulated surface is explained. Here, the signed distance is calculated only within the *narrow band* of the interface. The methodology involved to get a appropriate data structure, and the speed the of signed distance computation are elaborated. We compare the estimation of Φ of our approach with the first and the second order FMM. The applications of this module to various industrial problems are also discussed.

1.4. Outline of the thesis

Chapter 3 deals with the isosurface construction from the Φ values i.e. contouring. A variant of the marching cube algorithm is explained in detail in this chapter. The algorithm is designed to keep foreground and background information consistent, contrary to the neutrality principle followed for surface rendering in computer graphics [**91**]. Chapter 4 discusses the method based on least squares for estimating surface properties like position, normal and curvature with high order of accuracy from a discrete level set function. An arbitrarily oriented torus is taken as test case to compare our results with known analytical solutions. This method is also compared with the conventional finite difference techniques, used predominantly in the literature [**15**]. In Chapter 5, we present some of the applications of our approach to the propagation problems. We investigate the effect of high order curvature estimation in the study of mean curvature flow and bubble dynamics. The method is compared to regular benchmark problems. We verify Young-Laplace law for spherical bubble from our least squares curvature estimation and study the coalescence of two bubbles under surface tension force. We also present briefly, a study of VOF-levelset coupling for the injection mold flow problems with software SIGMASOFT. Finally, in Chapter 6 we summarize and give an outlook on future extensions of this approach.

CHAPTER 2

Computation of signed distance functions from surface triangulations

In this chapter, a procedure is elaborated for estimating the discrete signed distance function from surface triangulations. The surface and narrow band data structures are constructed for computing and storing Φ efficiently. These data structures are not only important for the computation of Φ, but also for processing CAD data for various industrial problems.

2.1. Surface data structure

We assume that the triangulation of the surface fulfills the basic criteria required for the definition of a topological space like a simplex and a simplicial complex (for definitions refer to Spanier [**86**]). In brief, the simplicial complex relates points, line segments, triangles and the n-dimensional counterparts. It should be noted that a 0-simplex consists of single point, a 1-simplex is the line segment (here we refer to as an edge), a 2-simplex is a triangle with interior, a 3-simplex is a tetrahedron with interior and so on. In the triangle the points, edges and face are given by

$$\begin{aligned} \textit{points} &: \{P_1\}, \{P_2\}, \{P_3\} \\ \textit{edges} &: E_1 = \{P_1, P_2\}, E_2 = \{P_2, P_3\}, E_3 = \{P_3, P_1\} \\ \textit{faces} &: F = \{P_1, P_2, P_3\}, \end{aligned}$$

as shown in Figure 2.1. Here, we give definitions and notations required for signed distance computations from surface triangulations.

2.1.1. Definitions and conventions.

Convention 1. *Orientation: The points and edges are numbered in the counterclockwise direction as shown in Figure 2.1. That is, if the points are ordered as* $\{P_1\}, \{P_2\}, \{P_3\}$*, then the edges are defined*

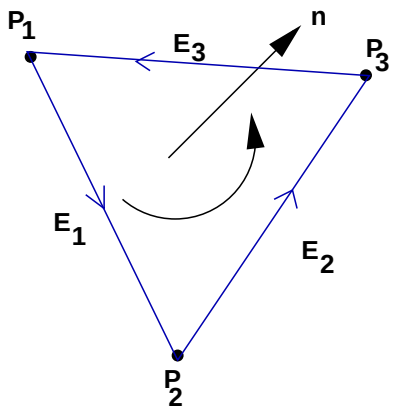


FIGURE 2.1. *Simplicial complex $\mathcal{K}$ of a triangle.*

in this order: $E_1 = \{P_1, P_2\}$, $E_2 = \{P_2, P_3\}$, $E_3 = \{P_3, P_1\}$, *and the face normal* $\vec{n_F}$ *is given by*

$$\vec{n}_F = \frac{(\vec{P_3} - \vec{P_2}) \times (\vec{P_2} - \vec{P_1})}{||(\vec{P_3} - \vec{P_2}) \times (\vec{P_2} - \vec{P_1})||},$$

where $\vec{P_i}$ *are the coordinates of* P_i, $i = 1, 2, 3$.

CONVENTION 2. *Surface triangulation information: The surface triangulation information is given as a list of unique point coordinates* $\vec{P_i}$, *and by their indices,* $i = 0, 1, 2, \ldots, n_p - 1$ *where* n_p *is the total number of points of a surface triangulation. The faces are represented as triples,* $\{P_i, P_j, P_k\}$, $i, j, k = 0, 1, 2, \ldots, n_p - 1$ *where the triangles are connected by three edges* $E_1^l = \{P_i, P_j\}$, $E_2^l = \{P_j, P_k\}$ *and* $E_3^l = \{P_k, P_i\}$ *for* $i \neq j, i \neq k, j \neq k$, $0 \leq l < n_f$.

DEFINITION 2. *Regular triangulation: The surface triangulation is regular if*

(i) *for any* i, k, $0 \leq i < n_p$ *and* $0 \leq k < n_p$, $i \neq k$, $\vec{P_i} \neq \vec{P_k}$ *(uniqueness of the point),*

(ii) *for any* l, $0 \leq l \leq n_f$, *the edge* $E_m^l = \{P_i, P_j\}$, $m \in \{1, 2, 3\}$, *does not repeat again in the same orientation (uniqueness of the edge). Here, it must be noted that,* $\{P_i, P_j\} \neq \{P_j, P_i\}$, *if* $P_i \neq P_j$.

In the above Definition 2 the points violating (i) are called **collapsed points**, and the edges violating (ii) are called **repeating edges**.

2.1. Surface data structure

DEFINITION 3. *Closed triangulation: The surface triangulation is closed if* $\forall\, m \in \{1,2,3\}$ *and for* $\forall\, l \in \{0,1,\ldots,n_f\}$, $\exists\, l' \in \{0,1,\ldots,n_f\}$ *and* $\exists\, m' \in \{1,2,3\}$ *such that the edge* $E_m^l = \{P_i, P_j\}$ *repeats once in the reverse orientation i.e.* $E_{m'}^{l'} = \{P_j, P_i\}$.

Edges that violate the condition in Definition 3, are called **open edges** or **hanging edges**.

REMARK 1. *If the triangulation is regular and closed, then the number of edges* n_e *is given by*

$$n_e = \frac{3n_f}{2}.$$

REMARK 2. *Convention 2 forms the basis of polygonal models like* **o***bject* **f***ile* **f***ormat(off)* [**31**]*,* **P***o***LY***gon file format(PLY) or Stanford Triangle Format* [**9**]*. These formats contain the information:*[1]

(1) n_p, n_f *and* n_e
(2) $\vec{P}_i$, $i = 0, 1, \ldots, n_p - 1$
(3) P_{1_j}, P_{2_j} *and* P_{3_j}, $j = 0, 1, 2, \ldots, n_f - 1$

REMARK 3. *The industrial standard for surface triangulations is the* **st***ereo* **l***ithography (stl) format. Typically, the stl format (ASCII or binary) has information for each triangular face, the normal and its three coordinates, viz.,* $\vec{n}_{f_i}$, $\vec{P}_{1_i}$, $\vec{P}_{2_i}$ *and* $\vec{P}_{3_i}$, $i = 0, 1, \ldots, n_f - 1$.

For the computation of signed distance functions, we need information about the points, edges and faces. On the other hand, in the above formats, there is no explicit information about the edges. Hence, edges have to be constructed from the point coordinates in the stl and from the indices in off format. In the off format the edges can be constructed while processing the point indices, but for stl formats a good algorithm to compute point indices is necessary. It is found that storing and processing off or PLY type information is fast and useful for the estimation of Φ. Therefore, we process the triangular information based on the off format.

Also, in our algorithm we extensively use the neighborhood information (like neighborhood of faces, edges and points). Therefore, it is

[1]PLY format has additional information like color and transparency, surface normals, texture coordinates and data confidence values.

desirable to construct data structures according to geometrical information.

DEFINITION 4. *Second tangent of adjacent triangles: Let F_1 and F_2 be two triangles which are closed,regular and adjacent to each other. Let $\vec{n}_{F_1}$ and $\vec{n}_{F_2}$ are the normals and (P_1, P_2, P_3) and (P_1, P_4, P_2), are point indices of the triangles respectively. The second tangents due to F_1 and F_2 are given by*

$$\vec{p}_{F_1} = \frac{\vec{n}_{F_1} \times (\vec{P}_2 - \vec{P}_1)}{\|\vec{n}_{F_1} \times (\vec{P}_2 - \vec{P}_1)\|}, \quad \vec{p}_{F_2} = \frac{(\vec{P}_2 - \vec{P}_1) \times \vec{n}_{F_2}}{\|(\vec{P}_2 - \vec{P}_1) \times \vec{n}_{F_2}\|}.$$

PROPOSITION 1. *Let F_1 and F_2 be two triangles which are closed, regular and adjacent to each other having second tangents $\vec{p}_{F_1}$ and $\vec{p}_{F_2}$. Then, $\vec{p}_{F_1} \cdot \vec{n}_{F_2} > 0 (< 0)$ implies $\vec{p}_{F_2} \cdot \vec{n}_{F_1} > 0 (< 0)$ and vice versa.*

PROOF.

$$\begin{aligned} \vec{p}_{F_1} \cdot \vec{n}_{F_2} &= \frac{1}{C_1}(\vec{n}_{F_1} \times (\vec{P}_2 - \vec{P}_1)) \cdot \vec{n}_{F_2} \\ &= \frac{1}{C_1}(\vec{n}_{F_2} \times \vec{n}_{F_1}) \cdot (\vec{P}_2 - \vec{P}_1), \end{aligned} \tag{2.1}$$

where $C_1 = \|\vec{n}_{F_1} \times (\vec{P}_2 - \vec{P}_1)\| > 0$. Similarly

$$\begin{aligned} \vec{p}_{F_2} \cdot \vec{n}_{F_1} &= \frac{1}{C_2}((\vec{P}_2 - \vec{P}_1) \times \vec{n}_{F_2}) \cdot \vec{n}_{F_1} \\ &= \frac{1}{C_2}(\vec{n}_{F_2} \times \vec{n}_{F_1}) \cdot (\vec{P}_2 - \vec{P}_1), \end{aligned} \tag{2.2}$$

where $C_2 = \|(\vec{P}_2 - \vec{P}_1) \times \vec{n}_{F_2}\| > 0$. Thus, from equation (2.1) and (2.2), we find

$$C_1 \vec{p}_{F_1} \cdot \vec{n}_{F_2} = C_2 \vec{p}_{F_2} \cdot \vec{n}_{F_1}. \tag{2.3}$$

As $C_1, C_2 > 0$, we find

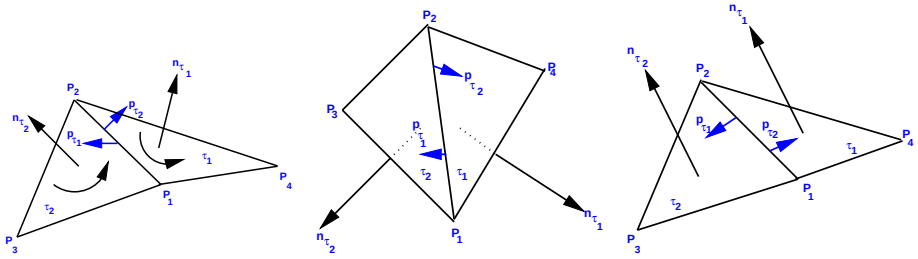
$$\vec{p}_{F_1} \cdot \vec{n}_{F_2} > 0 (< 0) \Leftrightarrow \vec{p}_{F_2} \cdot \vec{n}_{F_1} > 0 (< 0).$$

□

Motivated from Definition 4 and Proposition 1, we define different edge types.

DEFINITION 5. *Convex, concave and parallel edge types: An edge is called*

(i) **Convex** *if* $\vec{p}_{F_1} \cdot \vec{n}_{F_2} > 0$ *or* $\vec{p}_{F_2} \cdot \vec{n}_{F_1} > 0$,
(ii) **Concave** *if* $\vec{p}_{F_1} \cdot \vec{n}_{F_2} < 0$ *or* $\vec{p}_{F_2} \cdot \vec{n}_{F_1} < 0$,
(iii) **Parallel** *if* $\vec{p}_{F_1} \cdot \vec{n}_{F_2} = 0$.
This is shown schematically in Figure 2.2.



(a) *(b)* *(c)*

FIGURE 2.2. *(a) Convex, (b) concave and (c) parallel edges.*

The edge type E_t is defined by

$$E_t = \begin{cases} 1 & \text{if the edge is convex,} \\ -1 & \text{if the edge is concave,} \\ 0 & \text{if the edge is parallel.} \end{cases} \tag{2.4}$$

Having defined the quantities required to denote the neighborhood of faces and edges, we now study the neighborhood of points.

DEFINITION 6. *Convex, concave, parallel and CeCe point types : Let n_{P_i} be the number of edges that meet at point index P_i (see Figure 2.3). Then the point is called a*

(i) **Convex** *point if all edges are convex or combinations of convex and parallel types,*
(ii) **Concave** *point if all edges are concave or combinations of concave and parallel types, and*
(iii) **Parallel** *point if all edges are parallel,*
(iv) **Convex edge Concave edge (CeCe)** *point if the edges are of convex and concave type. The point type P_t is represented by*

$$P_t = \begin{cases} 1 & \text{if the point is convex,} \\ -1 & \text{if the point is concave,} \\ 0 & \text{if the point is parallel,} \\ 2 & \text{if the point is CeCe} \end{cases} \tag{2.5}$$

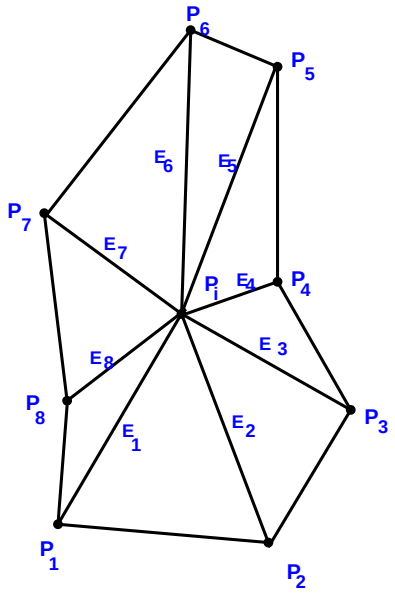


FIGURE 2.3. *Edges meeting at point P_i.*

Point data	Edge data	Face data
Number of points (n_p)	Number of edges (n_e)	Number of faces (n_f)
Coordinates ($\vec{P}_i$)	Point index (P_1, P_2)	Edge index (E_1, E_2, E_3)
Edges to point indices (ep)	Neighboring face index (f_l, f_r)	Normal to face ($\vec{n}_F$)
Point type (P_t) (convex/concave/ CeCe)	Edge type (E_t) (convex/concave/ parallel)	
	Neighboring face normal (n_l, n_r)	

TABLE 2.1. Surface triangulation data structure for points, edges and faces.

Later, we show that special care must be taken to get the correct sign of the distance function at CeCe points. The relevant parameters of the surface data structure are tabulated in Table 2.1.

2.1.2. Processing input triangulations. From stl-type real world coordinates, we extract the information about a surface, i.e. points and edges that are shared between faces. The first task is to construct point

indices from the coordinate information. The main aim in this task is to quickly decide if a new point coordinate $\vec{P}_i$ is already present. Naively comparing against all previously considered points gives an $O(N_p^2)$ algorithm, where $N_p = 3n_f$ in the stl format. To extract this information fast, we use a hashing method [**78**], [**52**]. Here, we form a hash table where the face index is represented by a distinct hash key constructed from the coordinates of the triangle. The performance of hashing is therefore $O(N_p)$, which is much faster than the naive method.

The next task is to quickly build the edge information of the surface. One way to construct edges is to form a look-up table from point indices. To compare against every previously constructed edge, again gives a quadratic algorithm. For a surface having a large number of triangles, this procedure is costly. To avoid this, we record the constructed edge with its two end points. Now for a new edge, only the edges where two end points has been generates must be compared against. This results in an $O(N_t c)$ algorithm, where N_t is the number of edges and c is the maximum number of edges emanating from a point given by the "edges to points" (ep) index.

From the above discussion we observe that accessing points from edges and faces and vice versa is organized efficiently. Therefore, this surface data structure can be used as a modular application to many geometrical processing problems. Before discussing these applications we would like to explain in detail the narrow band data structure used for our Φ computation.

2.2. Narrow band data structure

Let a discrete Cartesian mesh be given. There are two ways to treat the interface and its movement:

(1) Fixing a global cuboid around an object, and estimating Φ at each grid point within the box is referred to as *full grid* method. The advantage of this method is that the data structure is simple, but it is memory intensive. For large geometries it may not be practical.

(2) To restrict Φ only to a neighborhood of the interface known is referred to as *narrow band* method. The data structure in this case is intricate, but it is memory efficient. The efficiency in

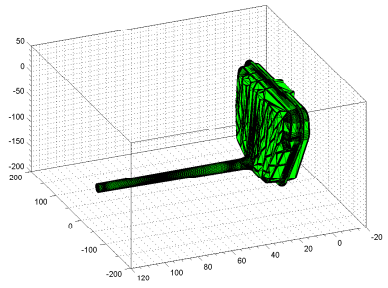


FIGURE 2.4. *Fixing a global box around the suction filter with long slender pipe.*

speed can also be attained by constructing the data structure appropriately.

Here we will consider only the narrow band method and its associated data structure. In this method, there are two ways to construct Φ:

(1) From *a discrete mesh*, i.e., finding the minimum distance to the surface by going through each grid point in the narrow band. Mauch [**53**] finds Φ by using this method for solving the static Hamiltonian - Jacobi equation.
(2) From a *surface triangulation*, i.e., iterating through grid points on the local cuboidal grid around each triangle, and estimating Φ by a direct method.

We use the second method to compute Φ. There are many algorithms to treat the narrow band method efficiently with respect to memory and time. The next subsection reviews some of the important narrow band methods used in various applications.

2.2.1. Brief review of (other) narrow band methods. The methodology even though known as narrow band, may require storage of full 3D grid indices [**64**]. Generally, a global box is fixed from the minimum and maximum coordinate values of the object coupled with the offset due to the narrow band width. This may result in unnecessary storage of grid index information. For example, Figure 2.4 shows a suction filter which has a long slender pipe attached to a thin box. Fixing a global box in this case, results in storing a large number of indices around the pipe outside the surface.

To avoid this there exist data structures like quad-trees in 2D [**87**] and octrees in 3D [**49**]. But these tree methods are slow when accessing

the neighboring indices of a particular grid point during propagation. Recently, Losasso et al. [**48**] showed, that these tree methods can be accessed efficiently by considering uniform grids, where every cell is an octree of its own. There are other techniques used to reduce memory and increase computational speed. Bridson [**10**] stores Φ on a coarse uniform grid, which nests a finer uniform grid that intersects the surface. He proposed to use hashing for the expansion of the grid, but did not demonstrate it. Nielson and Museth employed a method known as Dynamical Tubular (DT) grid [**58**]. In this method Φ is efficiently stored along the tubular region around the interface, by a compressed row storage technique derived from the well-known sparse matrix representation. The performance was compared with different octree and other narrow band methods. It was found that the DT-grid size is proportional to the size of the interface. Houston et al [**39**] used another approach known as Hierarchical run length encoding (H-rle) representation which combines the idea of rle sparse level set [**40**] and DT-grid technique to obtain a memory efficient algorithm.

2.2.2. Our approach. Our aim is to design a data structure being efficient for the propagation of the interface. Therefore, we propose two algorithms in this perspective: The first algorithm constructs the narrow band dynamically where we can access/modify the entries in a simple way. For memory intensive problems, the second algorithm based on a hashing procedure is used to access/modify the data inside the narrow band.

2.2.3. Dynamic narrow band construction. The schematic representation of a dynamic narrow band is shown in Figure 2.5. The two main data structures in this approach are termed *PhiList* and *Phi-Pointer*.

(1) *PhiList* stores the relevant information during the estimation of Φ. The following information is stored for computation:
   (i) Projection point $P(x_p, y_p, z_p)$,
   (ii) Projection object type $O_t \in \{0, 1, 2\}$. This is the region of projection. It may project onto the face ($O_t = 0$), edge ($O_t = 1$) or on to the corner point ($O_t = 2$) of the triangle.

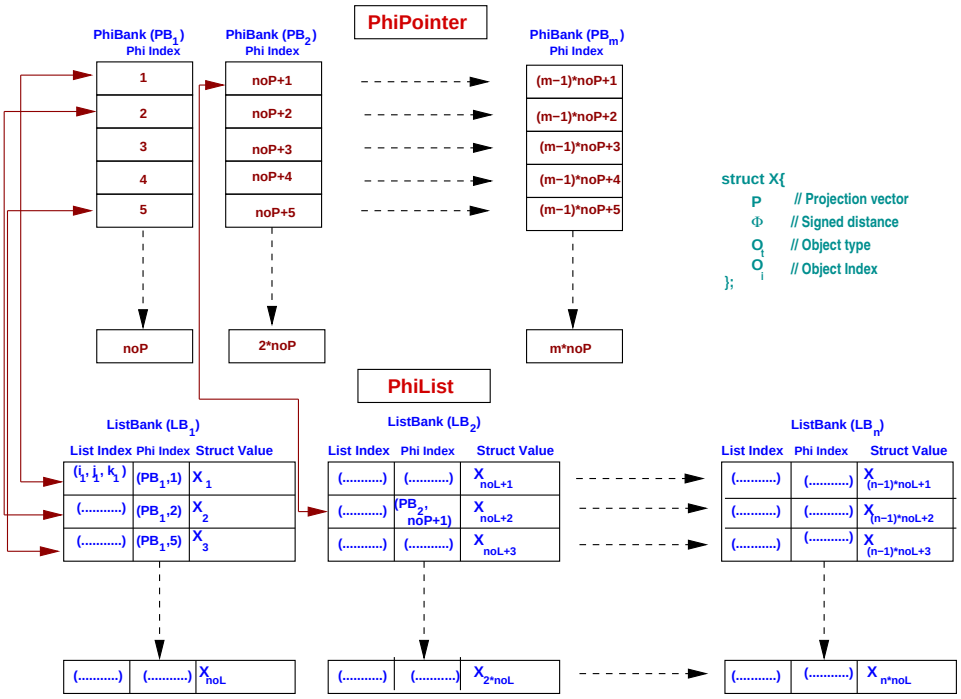


FIGURE 2.5. *Dynamic narrow band data structure.*

(iii) The object number O_i. This is the index corresponding to O_t. It may be the index number of the face, edge or point.

These information play a vital role in deciding the sign of the distance function at CeCe points. It is also required for the estimation of velocities on the surface during propagation. We can see from Figure 2.5 that the structure grows dynamically by memory chunks known as "ListBanks", denoted by $LB_1, \ldots, LB_n$ where n is the number of banks. This approach is similar to the idea of container classes used in the standard template library in C++. Φ is computed/updated only in the neighborhood of each triangle inside the narrow band. "ListIndex" gives 3D grid indices (i, j, k) to the *PhiPointer*. The number of ListIndex noL in each bank is fixed a priori.

(2) *PhiPointer* returns the index I_i of the grid if it is inside the narrow band, otherwise it returns null. This information is kept as "PhiIndex" and grows also in chunks called "PhiBanks" denoted by $PB_1, \ldots, PB_m$, where m is the number of banks.

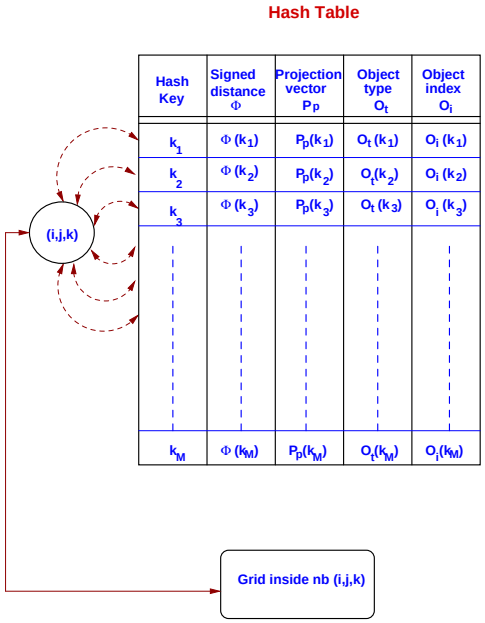


FIGURE 2.6. *Schematic representation of the hashing procedure.*

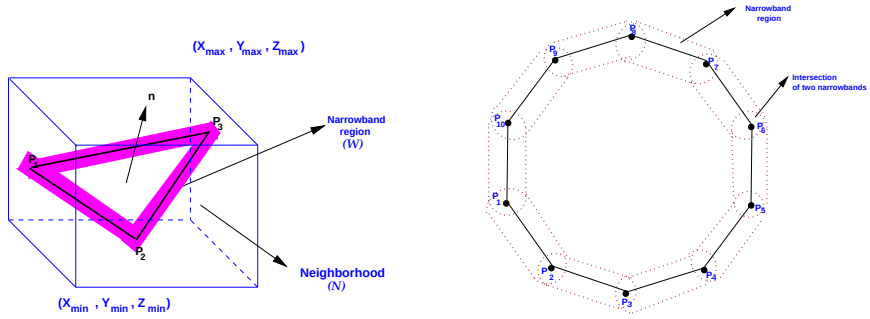
From the index number I_i we can get the bank number PB and the PhiIndex.

The basic memory storage depends on the size of the banks PB and LB which are certainly not optimum compared to DT- grid [**58**] or H-rle [**39**] methods. On the other hand, the above data structure is convenient for the computation of Φ and for the propagation.

2.2.4. Hash data structure. The basic memory storage in this method is proportional to the narrow band width w. We create a hash table of certain length M and a hash key K_i, $i = 1, \ldots, M$, is chosen on the basis of the hash function from the grid index (i, j, k) as shown in Figure 2.6. The speed of computation depends on fixing the initial size and finding the key without much collision of the hash table. The hash table is resized, whenever a narrow band entry increases above the fixed size. During propagation, to increase the speed of computation, we store the neighboring information a priori.

2.3. Initialization

Having established an efficient surface and narrow band data structure, we explain in detail the estimation of Φ from the triangulated surface. As mentioned earlier, we march through the neighborhood of



(a) *(b)*

FIGURE 2.7. *Narrow band around (a) a triangle in 3D and (b) a closed curve in 2D.*

each triangle and estimate Φ within the narrow band. The neighborhood $\mathcal{N}$ is a cuboid fixed from the coordinates of the triangle (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) and the narrow band offset w given by

$$\begin{aligned} \mathcal{N} \quad \in \quad & [\min(x_1, x_2, x_3) - w, \max(x_1, x_2, x_3) + w] \times \\ & [\min(y_1, y_2, y_3) - w, \max(y_1, y_2, y_3) + w] \times \\ & [\min(z_1, z_2, z_3) - w, \max(z_1, z_2, z_3) + w] \,. \end{aligned} \tag{2.6}$$

The narrow band for a 3D geometry is shown in Figure 2.7(a). The shaded region around the triangle $\mathcal{W}$ is the list of narrow band candidates. We show in Figure 2.7(b) the intersection between the neighboring narrow bands in 2D since it is very difficult to visualize in 3D.

Using expression (2.6) for fixing the local box may not be an optimum solution. To illustrate this in 2D, Figure 2.8(a) shows the local window over the line segments[2]. For the same segment this can be divided into two smaller windows as shown in Figure 2.8(b), reducing the number of grid points for the estimation of Φ.

In 2D, we can estimate a priori the number of local windows from the segment angle and narrow band width.

THEOREM 1. *Optimal box length depending on the angle of a segment: Let segment $\mathcal{S}$ of length $\mathcal{L}$ be inclined at an angle α to the x-direction on a uniform Cartesian mesh of width h and narrow band*

[2]In 2D we have line segments instead of triangles.

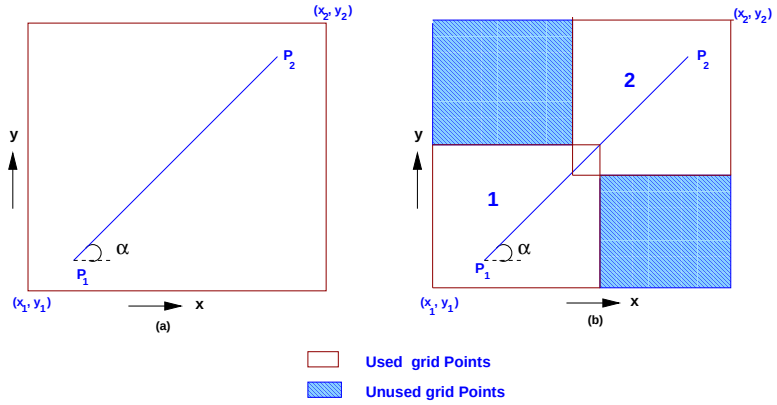


FIGURE 2.8. *Local window construction by (a) a rectangular window over the line segment and (b) by splitting into small windows.*

width w. Then, the optimum local window length l, is given by

$$l = h^2 w \frac{\cos\alpha + \sin\alpha}{\sin\alpha - \cos\alpha}, \tag{2.7}$$

such that $\mathcal{S}$ can be separated by $\mathcal{L}/l$ pieces.

PROOF. Let N_g be the number of grid points in the local window. Then for, given α, w and h, it can be seen from Figure 2.8, that

$$N_g = (l\cos\alpha/h + 2w)(l\sin\alpha/h + 2w). \tag{2.8}$$

The minimum local window length is therefore,

$$\frac{dN_g}{dl} = 0 \Rightarrow l = h^2 w \frac{\cos\alpha + \sin\alpha}{\sin\alpha - \cos\alpha}. \tag{2.9}$$

□

2.3.1. Regions of projection in 3D. In 3D finding an optimum box length is intricate as we have two different angles (polar and azimuth) for a triangle depending on three edges. Therefore, we use here the cuboid around each triangle F_i, where $i = 1, \ldots, n_f$ from the expression (2.6). Let us assume that the grid point P in this rectangular box is projected on the plane of the triangle at P_b. This ground projection point P_b can fall on any one of the seven regions in and around the neighborhood of F_i as shown in Figure 2.9.

The point of projection can be found from the definition of s_i and l_i.

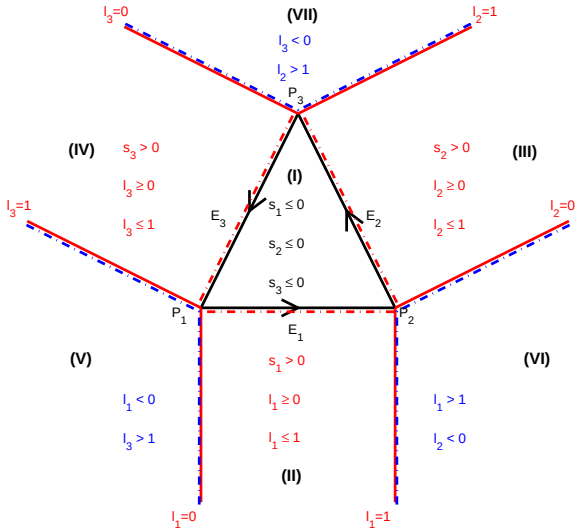


FIGURE 2.9. *The seven regions where the grid point P can be projected based on the signs of s_i,l_i, $i = 1, 2, 3$.*

DEFINITION 7. *Projection regions: Let $\vec{E}_i$, $i = 1, 2, 3$ be an edge vector corresponding to the point $\vec{P}_i$, $i = 1, 2, 3$ for a triangle F_i. Let $s_i := \|\vec{P}_b - \vec{E}_i\|_2$, $l_i := \vec{P}_b \cdot \vec{D}_i$, $i = 1, 2, 3$, where $\vec{D}_1, \vec{D}_2$, and $\vec{D}_3$ are given by $\vec{D}_1 = \dfrac{(\vec{P}_2 - \vec{P}_1)}{\|(\vec{P}_2 - \vec{P}_1)\|}$, $\vec{D}_2 = \dfrac{(\vec{P}_3 - \vec{P}_2)}{\|(\vec{P}_3 - \vec{P}_2)\|}$, and $\vec{D}_3 = \dfrac{(\vec{P}_1 - \vec{P}_3)}{\|(\vec{P}_1 - \vec{P}_3)\|}$. Then regions of projections (see Figure 2.9) are defined by*

$$
\begin{aligned}
&\textbf{(I)} \quad \textit{if} \quad s_1 \leq 0, s_2 \leq 0, s_3 \leq 0, \\
&\textbf{(II)} \quad \textit{if} \quad s_1 > 0, 0 \leq l_1 \leq 1, \\
&\textbf{(III)} \quad \textit{if} \quad s_2 > 0, 0 \leq l_2 \leq 1, \\
&\textbf{(IV)} \quad \textit{if} \quad s_3 > 0, 0 \leq l_3 \leq 1, \\
&\textbf{(V)} \quad \textit{if} \quad l_1 < 0, l_3 > 1, \\
&\textbf{(VI)} \quad \textit{if} \quad l_1 > 1, l_2 < 0 \ \textit{and} \\
&\textbf{(VII)} \quad \textit{if} \quad l_2 > 1, l_3 < 0.
\end{aligned}
$$

From Definition 7 we can get the projection point $\vec{P}_p$ and the local signed distance function ϕ_F for all seven regions provided we know the face normal of the triangle $\vec{n_F}$, the edge type E_t and the point type P_t. The signed distance Φ is found from minimum values of ϕ_F over all the triangles. Here, we define the local signed distance function ϕ_F of the grid point $\vec{P}$ from the surface triangle for all regions. *For $\vec{P}$ projecting*

to a CeCe point we find the sign of the distance function later using Proposition 2.

DEFINITION 8. *Local signed distance function ϕ_F and the projection point $\vec{P}_p$ from surface triangle: From Definition 7, $\vec{P}_p, \phi_F$ can be defined for each region as follows:*
For region **(I)** *(Projecting onto the face F):*

$$\begin{aligned} \vec{P}_p &= \vec{P}_b, \\ \phi_F &= sgn((\vec{P}_p - \vec{P}) \cdot \vec{n}_F) \ \ \|\vec{P} - \vec{P}_p\|_2. \end{aligned}$$

For region **(II)** *(Projecting onto the Edge E_1):*

$$\phi_F = \begin{cases} E_{t_1}\|\vec{P} - \vec{P}_p\|_2 & \text{if } E_1 \text{ is convex or concave,} \\ sgn((\vec{P} - \vec{P}_p) \cdot n_F)\|\vec{P} - \vec{P}_p\|_2 & \text{if } E_1 \text{ is parallel.} \end{cases}$$

For region **(III)** *(Projecting onto the Edge E_2):*

$$\phi_F = \begin{cases} E_{t_2}\|\vec{P} - \vec{P}_p\|_2 & \text{if } E_2 \text{ is convex or concave,} \\ sgn((\vec{P} - \vec{P}_p) \cdot n_F)\|\vec{P} - \vec{P}_p\|_2 & \text{if } E_2 \text{ is parallel.} \end{cases}$$

For region **(IV)** *(Projecting onto the Edge E_3):*

$$\phi_F = \begin{cases} E_{t_3}\|\vec{P} - \vec{P}_p\|_2 & \text{if } E_3 \text{ is convex or concave,} \\ sgn((\vec{P} - \vec{P}_p) \cdot n_F)\|\vec{P} - \vec{P}_p\|_2 & \text{if } E_3 \text{ is parallel.} \end{cases}$$

For region **(V)** *(Projecting onto the Point P_1):*

$$\begin{aligned} \vec{P}_p &= \vec{P}_1 \\ \phi_F &= \begin{cases} P_{t_1}\|\vec{P} - \vec{P}_1\|_2, & \text{if } P_1 \text{ is convex or concave,} \\ sgn((\vec{P} - \vec{P}_1) \cdot n_F)\|\vec{P} - \vec{P}_1\|_2 & \text{if } P_1 \text{ is parallel,} \\ \pm\|\vec{P} - \vec{P}_1\|_2, & \text{if } P_1 \text{ is CeCe.} \end{cases} \end{aligned}$$

For region **(VI)** *(Projecting onto the Point P_2):*

$$\begin{aligned} \vec{P}_p &= \vec{P}_2 \\ \phi_F &= \begin{cases} P_{t_2}\|\vec{P} - \vec{P}_2\|_2, & \text{if } P_2 \text{ is convex or concave,} \\ sgn((\vec{P} - \vec{P}_2) \cdot n_F)\|\vec{P} - \vec{P}_2\|_2 & \text{if } P_2 \text{ is parallel,} \\ \pm\|\vec{P} - \vec{P}_2\|_2, & \text{if } P_2 \text{ is CeCe.} \end{cases} \end{aligned}$$

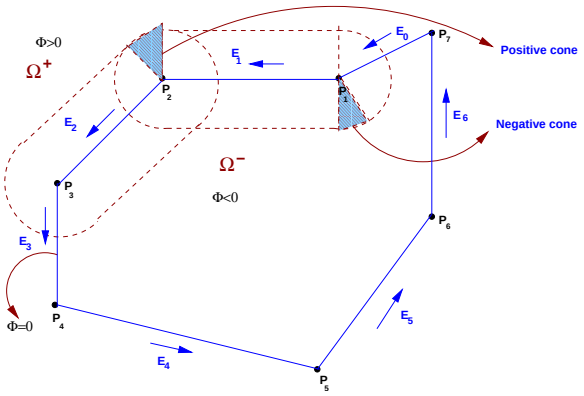


FIGURE 2.10. *Projection cones in 2D.*

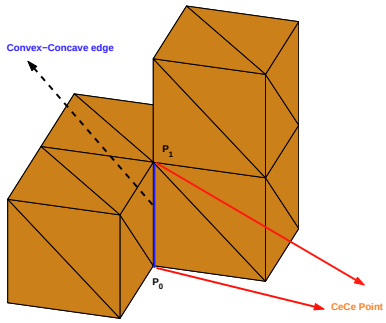


FIGURE 2.11. *Illustration of CeCe points P_0 and P_1.*

For region **(VI)** *(Projecting onto the Point P_3):*

$$\vec{P}_p = \vec{P}_3$$

$$\phi_F = \begin{cases} P_{t_3}\|\vec{P}-\vec{P}_3\|_2, & \text{if } P_3 \text{ is convex or concave,} \\ sgn((\vec{P}-\vec{P}_3)\cdot n_F)\|\vec{P}-\vec{P}_3\|_2 & \text{if } P_3 \text{ is parallel.} \\ \pm\|\vec{P}-\vec{P}_3\|_2, & \text{if } P_3 \text{ is CeCe.} \end{cases}$$

The signed distance function is then given by

$$\boxed{\Phi = \min_{0\le F\le n_f}(\phi_F)}. \tag{2.10}$$

To understand the projection to the corner point we illustrate the 2D analog of concave and convex edges in Figure 2.10. In 2D the angle between the edges determines whether they are convex or concave. For example, point P_1 between E_0 and E_1 is a concave (or negative) cone, i.e. it can have a projection only from Ω^-. Similarly point P_2 between E_1 and E_2 is a convex (or positive) cone as it can have a projection only from Ω^+. On the other hand in 3D we have a family of edges

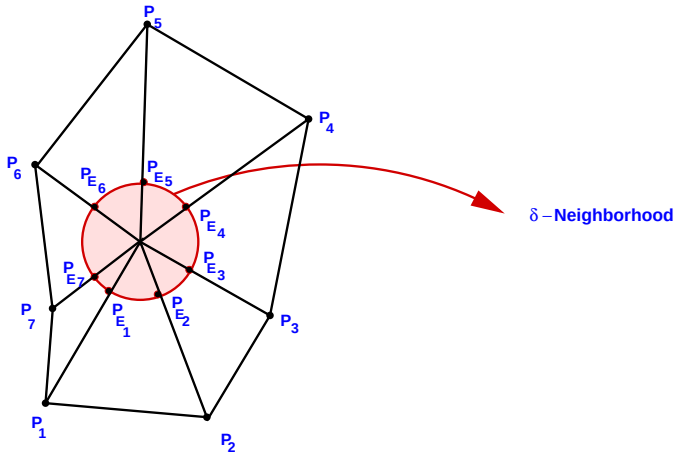


FIGURE 2.12. *Projection of the grid point P_g to the CeCe point P_e and its δ neighborhood.*

meeting at a point which can be concave or convex or a combination of both. Figure 2.11 shows an example of CeCe points. Here, all the edges are convex except the edge joining P_0 and P_1, which is a junction of a concave and convex edge. Therefore, P_0 and P_1 are CeCe points. For the determination of the sign in the distance function at a *CeCe* point we require a definition of an active plane and an active edge.

DEFINITION 9. *Active plane and active edge: Let a grid point P_g be projected to a triangular corner point P_e where n edges $E_1, E_2, \ldots, E_n$ meet as shown in Figure 2.12. Let $P_{E_1}, P_{E_2}, \ldots, P_{E_n}$ be a point on the edge $E_1, E_2, \ldots, E_n$, respectively, such that*

$$\|\vec{P_e} - \vec{P_{E_1}}\|_2 = \|\vec{P_e} - \vec{P_{E_2}}\|_2 = \cdots = \|\vec{P_e} - \vec{P_{E_n}}\|_2 = \delta.$$

Here, $0 < \delta < C$, where C is small (say $l/100$, where l is smallest edge length of all the triangles). Then E_i is called **active edge**, *if P_{E_i} satisfies*

$$d = \min_{j=1}^{n}(\|\vec{P_g} - \vec{P}_{E_j}\|_2) = \|\vec{P_g} - \vec{P}_{E_i}\|_2. \tag{2.11}$$

The corresponding triangular plane containing (P_g, P_{E_i}, P_e) is called **active plane** *(for the grid point P_g).*

We use Definition 9 in the following proposition for the determination of the sign of the distance function at CeCe points.

PROPOSITION 2. *Determination of the sign at CeCe points: Let P_g be a grid point projecting to a CeCe point P_e, and have n connecting*

edges $E_1, E_2, \ldots, E_n$. Then from equation (2.11) and Definition 8, the signed distance function at P_g is given by

$$\Phi_{P_g} = \begin{cases} E_{t_i}|\phi_F| & \text{if } (E_{t_i} = 1) \vee (E_{t_i} = -1) \\ sgn((\vec{P}_g - \vec{P}_i) \cdot \vec{n}_{f_j})|\phi_F| & \text{if } E_{t_i} = 0 \end{cases} \tag{2.12}$$

where ϕ_F is the local signed distance function and E_{t_i} is the edge type of E_i.

PROOF. Since E_i is an active edge, and (P_g, P_{E_i}, P_e) is an active plane, due to minimum distance criteria we can inherit the sign of the edge if $(E_{t_i} = 1) \vee (E_{t_i} = -1)$. Therefore, Φ given by,

$$\Phi_{P_g} = E_{t_i}|\phi_F| \text{ if } (E_{t_i} = 1) \vee (E_{t_i} = -1). \tag{2.13}$$

When the edges are parallel, i.e. $E_{t_i} = 0$, we can select the normal from the left or the right as they are equal[3]. Therefore,

$$\Phi = sgn((\vec{P}_g - \vec{P}_i) \cdot \vec{n}_{f_j})|\phi_F| \text{ if } E_{t_i} = 0. \tag{2.14}$$

□

REMARK 4. *The above Proposition 2 is a vital step in the determination of the sign of Φ for the grid point projecting to a CeCe point P_e. In the Φ computation, we check for the projection point object type O_{t_i}. If $O_{t_i} = 2$ and $P_{t_i} = 0$, then we correct the sign by iterating through different edges at P_e. Thus, the variable "edges to point" (ep) forms an important component in the surface data structure.*

To summarize, the following algorithm is used for the calculation of the signed distance function by the narrow band method.

ALGORITHM 1. *Signed distance calculation by the narrow band method*

1. *Find the global cuboid around the object and fix the initial size of PhiPointer and PhiList in the case of a dynamic narrow band. For hashing, fix the size of the hash table.*
2. *Iterate through each triangle F, and fix the local cuboid.*
3. *Find the regions of projection and ϕ_F. If it is on the:*
   - *face - find the sign from the normal of the triangle n_F,*

[3]The normal will not be equal only when the particular edge/face is collapsed, which is not possible from our assumption.

- *edge - multiply with the sign of the edge if* $E_t \neq 0$. *Otherwise, determine the sign from the neighboring face.*
- *non CeCe point - multiply with the sign of point type* P_t, *if* $P_t \neq 0$. *Otherwise, determine the sign from the neighboring face.*
- *CeCe point - find the correct sign from equation (2.12).*

2.4. Results

We grouped our results according to our two basic data structures, surface and the narrow band. In the next subsection we discuss the results of the surface data structure. Later, we investigate the estimation and accuracy of Φ due to the narrow band structure. We also performed a detailed analysis of speed and memory usage of the dynamic narrow band and hashing method. For the estimation of Φ, hashing is roughly 1.2 times faster than the dynamic narrow band method, provided we have a good a priori estimate of the hash table length and the proper hash function. Moreover, the advantage of hashing over the dynamic narrow band methods is that it requires less memory. On the other hand, the dynamic narrow bands method have a simple structure where data storage is sequential. Therefore, accessing/modifying in the dynamic narrow band method is simple compared to hashing where the entries in the table are almost random.

2.4.1. Results of surface data structure. We construct the surface triangulation by using Definition 2 and 3. *For each triangular face F if all three neighboring faces are processed, then it completes a closed object.* This concept is used to split the various closed components separately, if the given surfaces have many closed volumes. The software tool **LevelSplit**® is developed to split various components of stl files.

The algorithm is designed to process surface data structures fast. Figure 2.13 shows the plot of CPU seconds for processing different numbers of triangles for a torus geometry. (continuous line is the best linear fit). These tests were performed on a 32-bit dual AMD processor system of clock speed 2.5GHz. To process as many as 11.52 million triangles it takes just under 80 seconds to read the stl format and process the surface data structure.

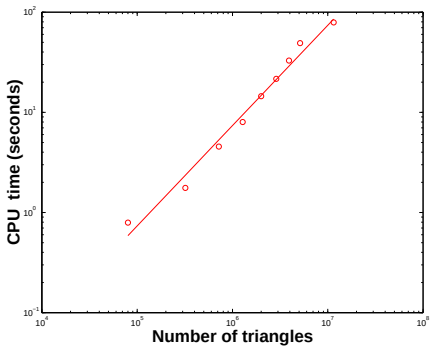


FIGURE 2.13. *Time to process the surface data structure.*

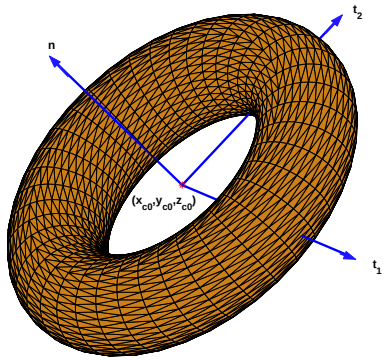


FIGURE 2.14. *Coordinate system of torus.*

2.4.2. Accuracy of Φ. We have taken a arbitrarily oriented torus as a test example to investigate the accuracy of the signed distance function. The estimated signed distance function from the surface triangulation is denoted by Φ^c and the known analytical function Φ is found from the implicit level set of the torus given by

$$\Phi \;=\; \sqrt{((a-R)^2 + (Z_{co} - Z_c)^2)} - r = 0, \tag{2.15}$$

where $a = \sqrt{(X_{co} - X_c)^2 + (Y_{co} - Y_c)^2}$ is described in the usual Cartesian coordinate system with torus center (X_c, Y_c, Z_c). (X_{co}, Y_{co}, Z_{co}) is the center of the torus in a local coordinate system $(\vec{t_1}, \vec{t_2}, \vec{n})$, as shown in Figure 2.14. The parameter r is the radius of the tube, and R is the distance of the center hole to the center of the tube.

Figure 2.15 shows the error in the estimation of Φ in the L_∞ norm for different surface resolutions. We measure the surface resolution by number of triangles n_f. Here, n_f varies from 80000 to 11.2 million triangles. From the best liner fit, it is found that the numerical accuracy

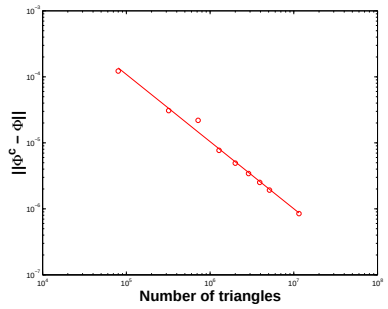
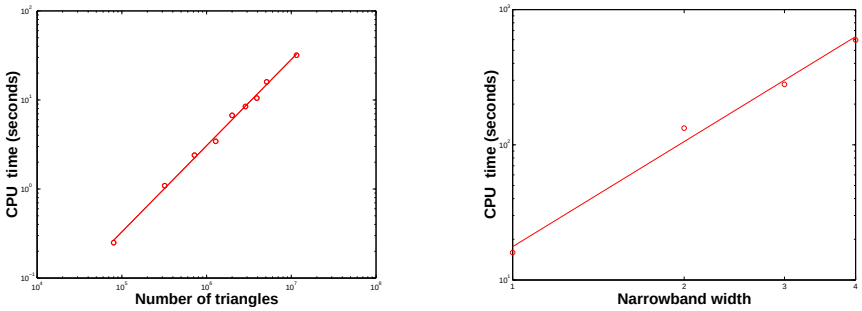


FIGURE 2.15. *Error in estimation of signed distance function.*



(a) *(b)*

FIGURE 2.16. *CPU time required to estimate* Φ *for (a) number of triangles and (b) narrow band width*

of Φ^c is of first order with respect to the triangulations. Therefore, a good surface resolution is necessary to get an accurate signed distance estimate from the triangular surface.

Figure 2.16(a) is the plot of the computation time versus number of triangles. In our approach, since we march through the neighborhood of each triangle to estimate Φ, we see that the computation time increases linearly with the number of triangles. Figure 2.16(b) is the plot of the computation time for different narrow band width. Here, we have taken $h = 0.01$ and the surface resolution as 2 million triangles. The computation speed increases cubically with the increase in narrow band width, as we fix locally a cuboid neighourhood $\mathcal{N}$ around each triangle (see equation (2.6) in Section 2.3). Thus, the speed of estimating Φ results in $n_f N_g$ operations, where N_g is the number of grid points inside the local cuboid window. To increase the speed of computation without loosing the accuracy, it is better to choose the narrow band width as small as possible.

Grid resolution $G_s = 4/h$	Direct		FMM1		FMM2	
	Accuracy of Φ (L_∞)	CPU speed (sec)	Accuracy of Φ (L_∞)	CPU speed (sec)	Accuracy of Φ (L_∞)	CPU speed (sec)
20^3	3.37×10^{-6}	302.42	0.1638	7.00	0.0545	7.01
40^3	3.42×10^{-6}	304.14	0.0707	7.16	0.0260	7.20
50^3	3.417×10^{-6}	305.68	0.0496	7.27	0.0173	7.24
100^3	3.426×10^{-6}	309.04	0.0210	7.50	0.0126	7.70
200^3	3.426×10^{-6}	318.15	0.0131	10.72	0.0060	12.36
400^3	3.42×10^{-6}	334.33	0.0083	69.51	0.0032	109.21
500^3	3.425×10^{-6}	336.81	0.0070	181.04	0.0026	294.52
600^3	3.417×10^{-6}	348.15	0.0053	670.60	0.0017	1084.78

TABLE 2.2. Accuracy and computational time for estimating Φ by the direct method and FMM for fine (5.2 million) surface triangle with $w = 4h$.

2.4.2.1. *Comparison of the direct method with FMM.* In the literature, the fast marching method (FMM) of Adalsteinsson and Sethian [**3**] is mostly preferred for the computation of Φ. In this method as mentioned in Chapter 1, the initial Φ is estimated around a thin layer which may be one or two mesh widths thick. Then Φ is computed at the desired grid point from these initial estimates by solving the Eikonal equation. The grid points near to the interface are updated first and then the subsequent layer inside the narrow band is processed. This process is efficiently performed by the heap sort algorithm [**5**], [**14**], [**82**].

Here, we compare the computational speed and accuracy of Φ estimated from the direct method with FMM. We fix the narrow band width $w = 4h$. We investigate the results for two cases: (1) with fine and (2) coarse surface triangles. For FMM, the initial estimate of the signed distance function around a layer of grid points i.e. within $w = h$ is obtained by the direct method. It should be noted that the accuracy of FMM depends on the grid width and not on the surface triangulation. The uniform mesh width h is given by $4/G_s$, where G_s is the grid resolution. For our test, we use $20^3, 40^3, 50^3, 100^3, 200^3, 400^3, 500^3$ and 600^3 grids.

Table 2.2 shows the errors in the L_∞- norm for the estimation of Φ by first order and the second order FMM (denoted by FMM1 and FMM2) and the direct method for fine surface triangles. Also shown in this table are the computational time for the FMM and the direct method. It is found that FMM2 is more accurate than FMM1 but the order of convergence for both methods is roughly of $O(h)$. On the other hand, the direct method are more accurate than FMM1 and FMM2 and as expected the order of convergence is independent of mesh width. With regards to the speed, for a coarse grid, FMM1 and FMM2 is much more faster than the direct method. But for very fine mesh width, we find the direct method is faster and still more accurate than FMM1 and FMM2. The reason is that the efficiency of the heap sort mechanism in FMM is proportional to $N_g log(N_g)$, where N_g is the number of trial values. For a very fine mesh width N_g becomes very large and hence the computation of FMM takes a longer time than the direct method. We have found a similar result also for very coarse triangles as shown in Table 2.3.

One way to tackle this problem is to increase further the order of FMM approximation. But this requires an initial estimate of Φ for a thick layer of grid points around the interface, which in turn increases the computation time. Later in Chapter 4, we show that that the direct method is also more accurate than FMM for determining the surface parameters. Therefore, for our application the direct method offer more advantages and hence preferable than FMM.

2.5. Applications

This algorithm has been applied to various industrial problems as pre- and post-processor. An obvious application of this approach is to estimate Φ for any complex geometry where the surface is given by triangulation. Figure 2.17 illustrates a complex object in casting where this approach is used for initializing Φ for the study of topological optimization by level set methods [**79**] , [**96**]. We list here some more applications of our algorithm to various industrial problems.

2.5.1. Smoothing of the surface geometry. The point edges data structure is also very useful during regularization of the surface geometry. For instance in the study of elastic deformation, to avoid

Grid resolution $G_s = 4/h$	Direct		FMM1		FMM2	
	Accuracy of Φ (L_∞)	CPU speed (sec)	Accuracy of Φ (L_∞)	CPU speed (sec)	Accuracy of Φ (L_∞)	CPU speed (sec)
20^3	1.20×10^{-4}	8.47	0.1638	0.25	0.0545	0.26
40^3	1.20×10^{-4}	8.71	0.0708	0.29	0.0260	0.31
50^3	1.230×10^{-4}	8.77	0.0497	0.31	0.0173	0.33
100^3	1.221×10^{-4}	9.29	0.0211	0.52	0.0126	0.64
200^3	1.231×10^{-4}	10.34	0.0131	3.16	0.0060	4.77
400^3	1.232×10^{-4}	12.64	0.0083	58.39	0.0033	98.42
500^3	1.233×10^{-4}	13.74	0.0070	167.30	0.0026	274.78
600^3	1.233×10^{-4}	15.95	0.0054	642.21	0.0017	1039.82

TABLE 2.3. Accuracy and computational time for estimating Φ by the direct method and FMM for coarse (80000) surface triangles with $w = 4h$.

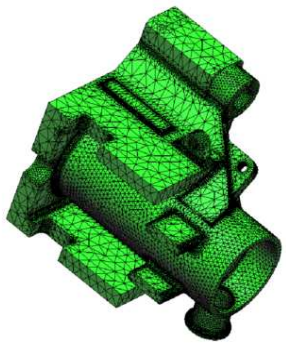


FIGURE 2.17. *Illustration of a complex object in casting where Φ is used as a "initialization step" for the study of topological optimization.*

singular sharp corners, a smoothing is required before proceeding to the solver part. Here, we propose a smoothing similar to the Laplace-Beltrami operator for smoothing of edges.

DEFINITION 10. *Let n_j be the number of edges meeting at point P_0, with coordinates $\vec{P}_i$, $i = 1, \ldots, n_j$. Then, the weight of smoothing w_i*

(a) *(b)*

FIGURE 2.18. *Structure of a (a) non-smooth and (b) smoothed fiber with* $w_0 = 0.1$.

at this point due to edge E_i *is given by*

$$w_i = w_0 + \frac{1}{d_i}\frac{(1-w_0)}{\bar{w}},$$

where

$$\bar{w} = \sum_{i=1}^{n_j} \frac{1}{d_i}.$$

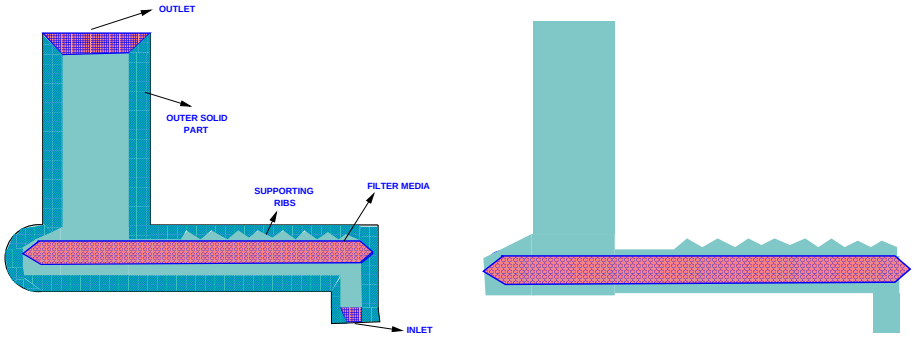
Here, $0 \leq w_0 \leq 1$ *is the global weighting parameter and* $d_i = \|\vec{P}_0 - \vec{P}_i\|_2$. *The new coordinate* $\vec{P}_0{}'$ *is given by*

$$\vec{P}_0{}' = w_0\vec{P}_0 + \sum_{i=1}^{n_j} w_i\vec{P}_i.$$

Figure 2.18(a) and (b) show the non-smoothed and a smooth fiber structure for $w_0 = 0.1$, respectively. This smoothing and the estimation of Φ thereafter, is used in **E**xplicit-**J**ump **I**mmersed **I**nterface **M**ethods (EJIIM), for solving elliptic boundary value problems in complicated geometries [**73**], [**74**].

2.5.2. Grid identification in flow solvers. Up to now, we have confined our discussion within the narrow band around the surface for a given closed object. Away from the narrow band, one can get the information about the volume enclosed by the surface commonly referred to as *out-of-box* level set methods in the literature [**58**]. For instance, in the study the flow through porous media, we have a finite volume solver called **SuFiS**, which uses this information to know the status of grid points.

Figure 2.19(a) shows a schematic representation of a suction filter which has inlet, outlet, and a porous filter medium supported by solid



(a) *(b)*

FIGURE 2.19. *A suction filter with (a) inlet and outlet closed and (b) fluid and porous medium.*

(a) *(b)*

FIGURE 2.20. *Separation of the (a) solid outer part and (b)fluid part.*

ribs. When standard CAD software like PRoE is used to triangulate (without filter medium) with closed inlet and outlet, it produces two parts, viz, outer solid part and the inner fluid region. For the flow computations we require only the fluid region and the filter medium as shown in Figure 2.19(b). This is performed by running "LevelSplit" which splits the inner fluid part from the solid part as shown in Figure 2.20(a) and (b), respectively. The grooves on the upper part in Figure 2.20(b) are the impression of the ribs from the solid portion.

With this triangulation, we estimated Φ in a neighborhood of the surface by a narrow band method. To calculate the status of other grid points, we used a ray tracing method for each xy plane and stored the information efficiently by **r**un **l**ength **e**ncoding (rle). Figure 2.21 shows the cross section of fluid cells at a particular z plane. This plane is the region, where the solid rib pierces the fluid cells. Here, the

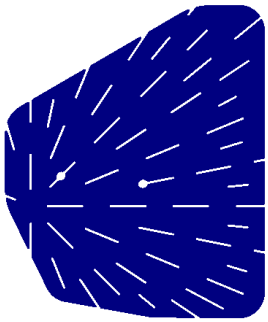


FIGURE 2.21. *Cross section of the suction filter at a particular z plane where solid ribs cut the fluid part. Blue region is the fluid part and the white is the solid part.*

blue color is the fluid and the void regions are the solid cells. With this crucial information, we can do a local refinement to get a better representation of the ribs. This method is found to be useful in the study of flow characteristics around the ribs. The important aspect in this application is the speed of preprocessing this information. For the surface resolution of around 2 million triangles, it takes around 28 seconds to process 12.31 million grid points on a 32-bit dual AMD processor system, of clock speed 2.5GHz. This approach is currently used for similar type of applications to extract the volume information from the surface triangulations.

CHAPTER 3

Marching cube algorithm for isosurface construction

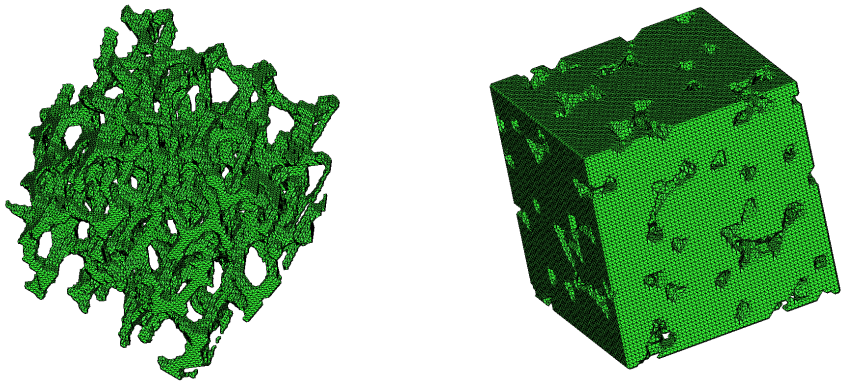
Reinitialization is a powerful tool used in level set methods, for keeping Φ to be a signed distance function. As discussed in Chapter 1, there are two ways of reinitializing Φ viz., the explicit contouring approach and the non-explicit representation. Here, we discuss the explicit contouring approach in detail. The aim of this chapter is as follows: Let $\Phi(x, y, z)$ be given on a discrete cuberille grid i.e. $\Phi_{i,j,k} = \Phi(x_i, y_j, z_k)$, $(i, j, k) \in I \subset \mathbb{Z}^3$. Then, we construct the isosurface Γ, such that

$$\Gamma = \Phi_0 = \{(x, y, z) : \Phi(x, y, z) = 0\}. \tag{3.1}$$

$\Phi(x_i, y_j, z_k)$ is also referred to as *discrete field function*. A variant of the marching cube algorithm is used for constructing the isosurface from discrete field function. The algorithm is designed to keep foreground and background information consistent, contrary to the neutrality principle followed for surface rendering in computer graphics.

3.1. Introduction

The Marching Cube(MC) algorithm due to Lorensen and Cline [**46**] is a popular method for isosurface contouring from discrete scalar field values. It essentially constructs the isosurface from thresholded field values on the corner points of each cell by marching along a particular direction. Even though there are 256 possible sign combinations in a cell, they can be generally grouped into 14 basic topological cases [**97**]. The construction of the isosurface using these 14 topological cases alone leads to ambiguity of joining edges [**20**]. To avoid this, more cases are proposed either by looking at the adjacent cube [**97**] or by consistent triangulation representation of the surface by asymptotic decider [**59**]. Alternative strategies to remove the ambiguity are saddle points [**57**],



(a) *(b)*

FIGURE 3.1. *The triangulation of (a) Ni - foam and its corresponding (b) porous structure discretized by a* $128 \times 128 \times 128$ *geometry.*

sub-classification of the topological cases [**13**], and trilinear surface representation [**17**]. In the field of computer graphics, certain objectives are followed during the isosurface construction. Van Gelder and Wilhelms [**91**] list six main objectives:

(1) The construction algorithm should yield a continuous surfaces.
(2) The surface should be a continuous function of the input data.
(3) The surface should be topologically correct.
(4) The algorithm should be neutral with respect to positive and negative sample data values (relative threshold). Multiplying the samples (threshold) by -1 should not alter the surface.
(5) The algorithm should not create artifacts not implied by the data.
(6) The algorithm should be efficient for real time visualization.

From the above objectives, (4) is considered as crucial and also controversial [**91**]. Many algorithms and definitions satisfy these criteria. In some applications, objective (4) may contradict the correct topological description of (3). For example, Figure 3.1(a) shows the structure of a nickel foam in a cubic volume. For the study of elastic deformation of foam, triangulation of these structures are necessary. On the other hand, for the treatment of porous media, we need the triangulation of the pore space shown in Figure 3.1(b).

For our application therefore, objective (4) should account for non-neutrality. Further, we need two more criteria:

(7) The algorithm should give closed contour.

(8) The surface triangles should not degenerate.

The algorithm described here, constructs the isosurface, continuous and *consistent* with the foreground and background information. Here we follow certain conventions with which we can get the surface construction correctly without ambiguity.

3.2. Definitions and Conventions

We present definitions of basic terminologies and conventions used in our MC algorithm.

DEFINITION 11. *Voxel: Voxel stands for volume element. It is analogous to pixel in 2D. A data sample is referred to as voxel.*

DEFINITION 12. *Cell and Cell vertices: The cubical region formed by eight grid points* (i, j, k), $(i+1, j, k)$, $(i+1, j+1, k)$, $(i, j+1, k)$, $(i, j, k+1)$, $(i+1, j, k+1)$, $(i+1, j+1, k+1)$ *and* $(i, j+1, k+1)$ *is a cell and its grid points are cell vertices.*

CONVENTION 3. *Sign Conventions: The grid point* (i, j, k) *is assigned positive if* $\Phi(i, j, k) > 0$ *and negative if* $\Phi(i, j, k) < 0$, *such that* $\Phi = 0$ *at the isosurface. The closed region where* $\Phi(i, j, k) < 0$ *is represented by* Ω^- *and its complimentary region as* Ω^+.

CONVENTION 4. *Intersection Point* P_i*: In any of the cell, if some of the vertices are positive and some negative, then the isosurface passes through that cell. The intersection point* P_i *is a point on the isosurface that crosses the edge between different signs of the cell vertices.*

CONVENTION 5. *Normal* $\vec{n}$ *to the isosurface : To be consistent with our notation (see Convention 1), the positive normal defined on the isosurface* Γ *points towards the region* Ω^+.

In our MC algorithm the 14 topological cases are grouped according to the number of negative signs and their position in a cell as shown in Figure 3.2. These cases are labeled as **A** (no negative), **B** (one negative), **C** (two negatives), **D** (three negatives) and **E** (four negatives).

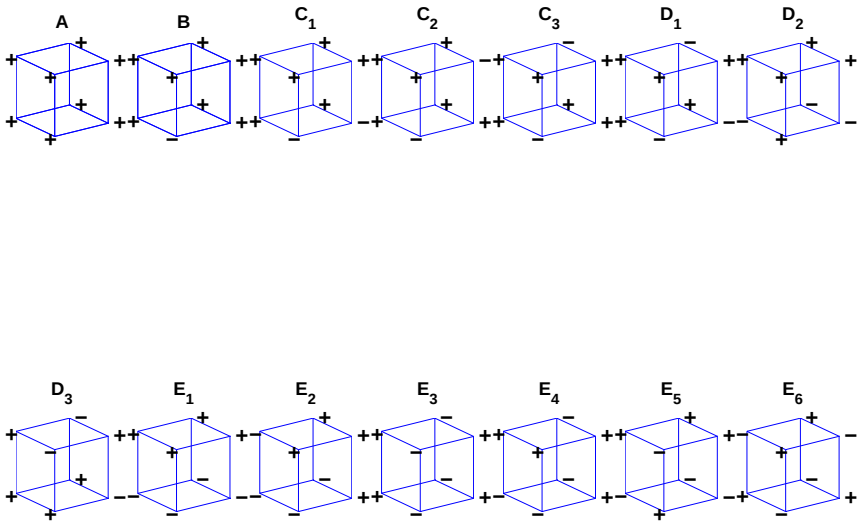


FIGURE 3.2. *The basic 14 topological cases in an MC algorithm.*

The sub cases are the relative position of negative signs within a cell. For instance $\mathbf{C_1}$ has negative signs on the same edge, $\mathbf{C_2}$ on the face diagonal and $\mathbf{C_3}$ on the leading diagonal of a cell. Similarly, $\mathbf{D}$ has three sub cases and $\mathbf{E}$ six sub cases.

The triangulation in all these topological cases follows certain convention, otherwise the isosurface connection will be *ambiguous* within the cell or on the face of the cell. The *ambiguous cell* contains diagonally opposite pair of positive vertices and a diagonally opposite pair of negative vertices in the cell (eg., case $\mathbf{C_3}$). Similarly an *ambiguous face* contains a diagonally opposite pair of positive vertices and a diagonally opposite pair of negative vertices on the face of the cell (eg., right face of $\mathbf{D_2}$). To make the connection non-ambiguous, we follow

CONVENTION 6. *In a cell the* Ω^- *part is connected.*

For instance if we have a face ambiguous case as shown in Figure 3.3(a) then from Convention 6, Figure 3.3(a) is right, and 3.3(b) is wrong. Following this convention, the triangulation of 14 and their symmetrical topological cases are constructed as shown in Figure 3.4. Due to non-ambiguous connections in the triangulation, the symmetry *need not* be maintained in all complimentary cases (for instance, $\mathbf{C_3}, \mathbf{D_3}$ cases).

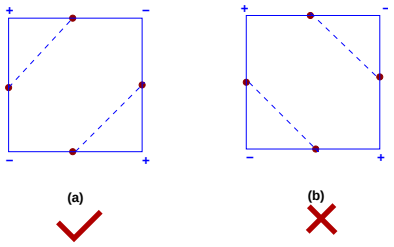


FIGURE 3.3. *Convention for joining the edges for a face ambiguous case: (a) is correct and (b) is false.*

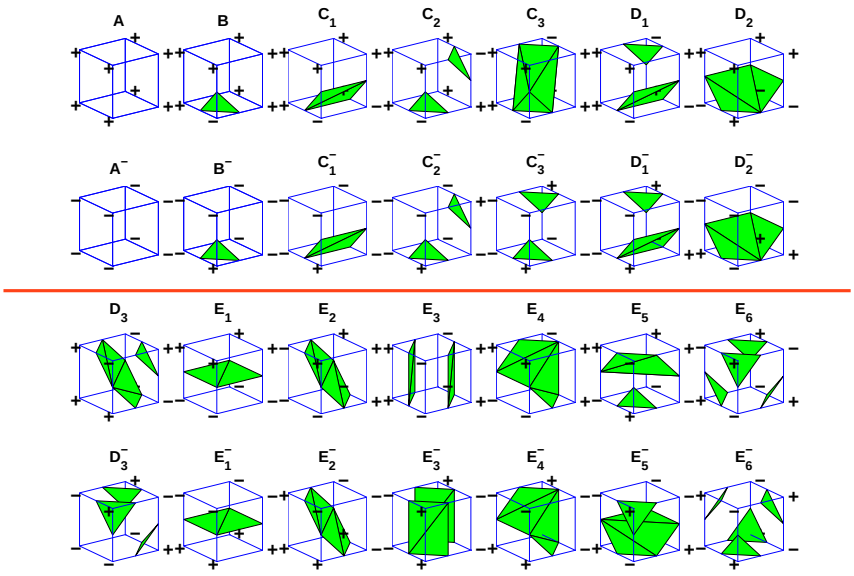


FIGURE 3.4. *The surface triangle construction in the basic 14 topological and its corresponding symmetrical case inside the cube.*

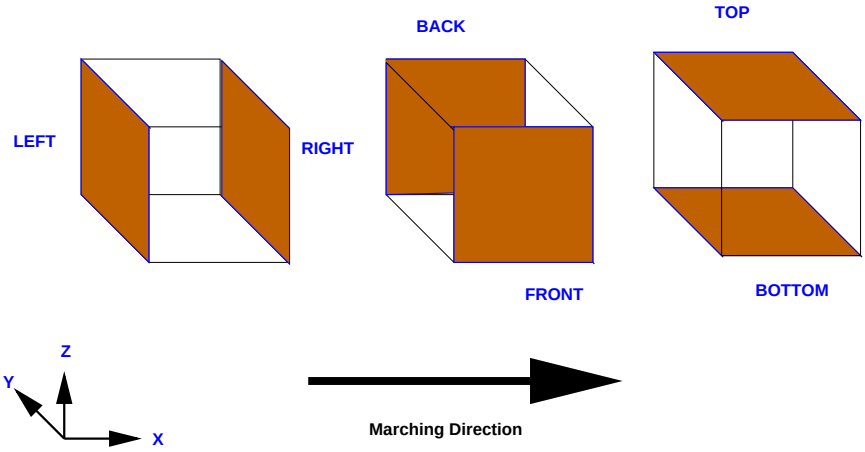


FIGURE 3.5. *Direction of marching and conventional names of six faces*

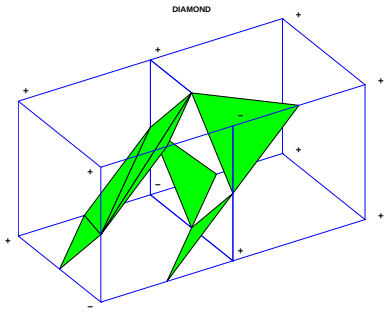


FIGURE 3.6. *A hole inside the surface due to the non-closure of planar triangles.*

CONVENTION 7. *The marching direction of visiting each cube is from left to right. The six faces are termed as left, right, front, back, bottom and top as shown in Figure 3.5.*

3.3. Extension of topological cases

The above convention solves the internal ambiguity of each cell in an MC algorithm, but it does not deal with other ambiguous cases like formation of holes. For example, let us consider a situation where $\mathbf{D_3}$ and $\mathbf{B}$ cases exist next to each other as shown in Figure 3.6. Considering the above topological cases only for isosurface generation leads to a hole between the cells as there are no rules to triangulate the cell face. For this problem, to avoid ambiguous representation, there are algorithms with extended cases [**13**], [**44**], [**17**].

In an usual MC algorithm due to neutrality principle a facial triangle on the cube is avoided [**44**], [**91**]. On the contrary as we account for non-neutrality we need facial triangles for the continuity and closedness of isosurface. The two facial triangles which we call *a diamond* is constructed based on the signs of cell vertices on the adjacent cube.

3.3.1. Criteria for placing diamonds. We discuss here the criteria for placing a diamond between the cells. Figure 3.7(a) show an example where $\mathbf{D_3}$ and $\mathbf{B}^-$ cases existing next to each other. From Convention 6 we construct triangulation such that Ω^- is connected in each cell. To be consistent with this convention we follow the same convention to the adjacent cube also i.e., the Ω^- in $\mathbf{D_3}$ and $\mathbf{B}^-$ is connected. Therefore there should *not* be a diamond between the cells in this case as all vertices of the right face of $\mathbf{B}^-$ are negative. On the

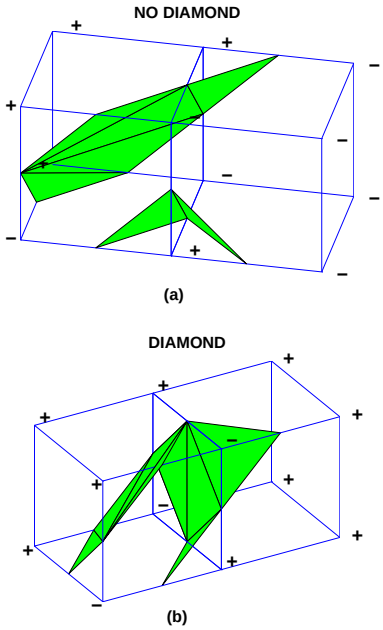


FIGURE 3.7. *Criteria for placing a diamond on the face of the cube. (a) No diamond constructed and (b) diamond constructed.*

other hand in Figure 3.7(b), following this convention, Ω^- from $\mathbf{D_3}$ should be closed within $\mathbf{D_3}$ as all vertices of the right face of $\mathbf{D_3}$ are positive. Therefore, we should construct a diamond on the face of the cube. Similar approach is followed on the back and on the top part of the cube. The placing of diamond is decided only when visiting the adjacent cube. This will clear the ambiguity and also ensure *no collapsed* surface is formed during marching. By this way, *we do not require to consider extra case* for solving the ambiguity which is usually handled in an MC algorithm (see [**44**], [**17**] for the details).

REMARK 5. *In a sense, having a decision to place a diamond leads to a large number of cases apart from 256 cases within the cell. It requires another 16 cases on the adjacent cubes along x, y and z directions which results in $256 \times 16 \times 3 = 12288$ cases as depicted in Figure 3.8.*

3.4. Implementation

In order to realize each case, we use certain notation here. Figure 3.9 shows the notation for vertices $\mathbf{v_1}, \ldots, \mathbf{v_8}$ and the intersection

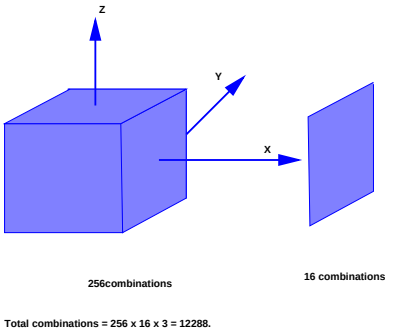


FIGURE 3.8. *Total number of combination in our MC algorithm*

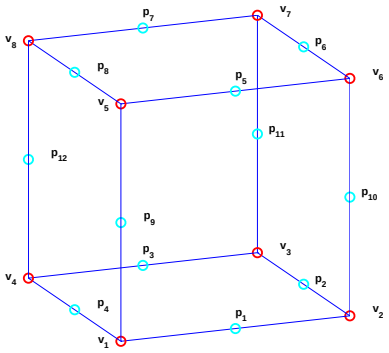


FIGURE 3.9. *Vertex representation and intersection point indices for MC algorithm.*

points on the surface $\mathbf{p_1}, \ldots, \mathbf{p_{12}}$. For clarity, we have shown the intersection points on the middle of cell edge. In a general situation it can lie anywhere on the edge depending on the field values at the corner of the cell. Each vertex is indexed by a bit representation, i.e., indices of $\mathbf{v_1} = 1, \mathbf{v_2} = 2, \ldots, \mathbf{v_8} = 128$ and is active when the sign is negative. For example if $\mathbf{v_1}, \mathbf{v_2}$ and $\mathbf{v_3}$ are negative the case number is $1 + 2 + 4 = 7$. If all the vertices are negative the case is 255 and their complimentary case is 0. This representation helps to identify different topological cases easily.

For using this algorithm for the propagation and reinitialization of the interface, it is necessary that triangulation notations are consistent with the definitions discussed in Chapter 2. Therefore, when a new point is added during marching we store its coordinates $\vec{P}_i$ and carry its indices P_i to the right, back and top of the cube as we march from left to right.

3.5. Quality of the surface triangulation

Apart from the eight objectives mentioned in Section 3.1, we investigate also quantitatively the constructed isosurface by our MC algorithm. For the generation of volume mesh from surface mesh, it is necessary that surface triangulation should have good properties. One of the important property is the angular characteristics i.e, the maximum and the minimum angle over all triangles in a surface mesh. We investigated the angles from various synchrotronic and Reflection Electron Microscopy (REM) grey scale images. If the field values are given as "on" and "off" information like in binary images then we have an important result:

PROPOSITION 3. *Let $\Phi \in [0,1] \subset I$ (as in binary images). Then the constructed triangle from an MC algorithm, has angle α in the range $30^0 \leq \alpha \leq 120^0$.*

PROOF. When the field values are given as 0 (inside) and 1 (outside), then the intersection point lies at the center of the edge in a cube. From the topological cases we can find the the maximum and the minimum angle of each triangle by law of cosines. It is seen that this angle α is bounded by $30^0 \leq \alpha \leq 120^0$. □

REMARK 6. *Relation between the aspect ratio a_r and the angle: For describing the quality of the mesh, sometimes aspect ratio is used instead of angular representation. The aspect ratio a_r is defined as the ratio of longest edge to the shortest altitude of the triangle. Nevertheless, we can relate these two properties. If α_1 and α_2 are the minimum and the maximum angle over all the triangles in the surface mesh, respectively then, a_r is bounded by* [**8**]

$$|1/sin\alpha_2| \leq a_r \leq |2/sin\alpha_1|.$$

In our case a_r is bounded by $1.1547 \leq a_r \leq 4$.

Figure 3.10 shows the fibre structure, triangulated from the binary representation. The histogram of the maximum and minimum angles and the aspect ratio of this fibre are shown in Figure 3.11. It is seen that the angle and aspect ratio are desirable for meshing the volume (say by tetrahedron) from the triangulation. This mesh is used

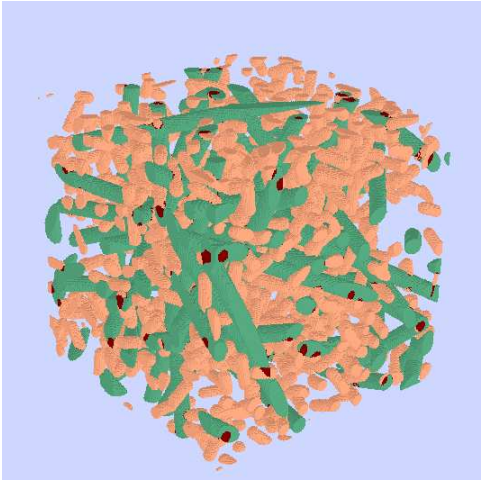


FIGURE 3.10. *Structure of fibre in* 256^3 *resolution.*

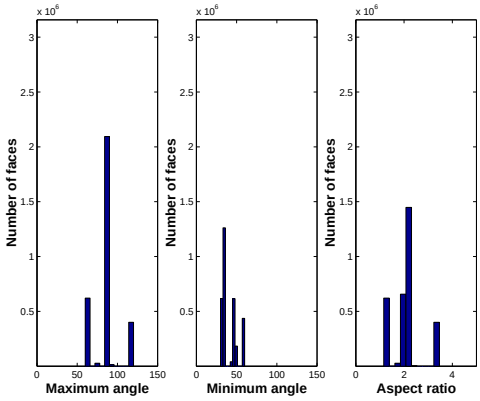


FIGURE 3.11. *Histogram of largest and smallest angle and the aspect ratio.*

in Boundary Element method to study linear electrostatics [**61**], and also in **E**xplicit **J**ump **I**mplicit **I**nterface **M**ethod (EJIIM) for linear elasticity problems [**74**].

It is found that on a standard desktop of 2.5Ghz clock speed, it takes around 32 seconds of CPU time to triangulate the surface from the field values of 512^3 by our MC method. Figure 3.12 shows the plot of CPU time with respect to different grid resolutions G_s ($32^3, 64^3, 128^3, 256^3$ and 512^3). Also, the algorithm is found to be robust and can triangulate for various spatial resolution. Figure 3.13, shows a portion of of nickel foam geometry for $32^3, 64^3$ and 128^3 resolutions, where the isosurface is constructed from synchrotronic data sets.

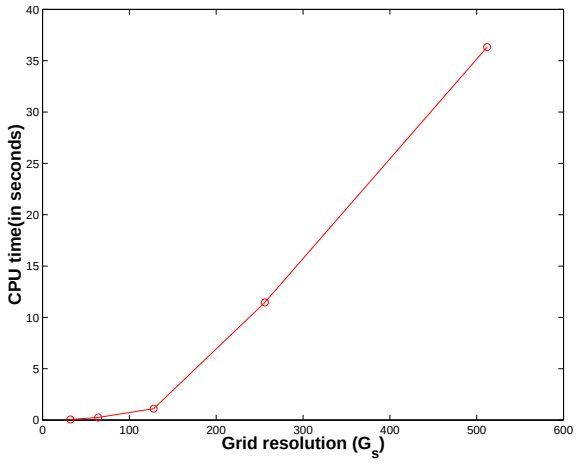
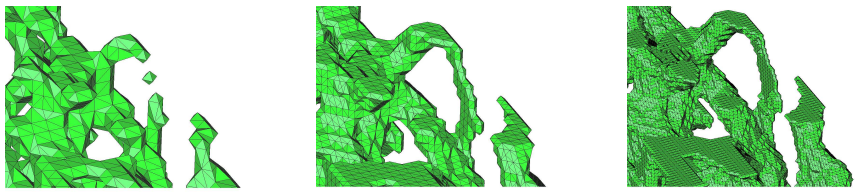


FIGURE 3.12. *CPU seconds to triangulate different grid resolutions.*



(a) *(b)* *(c)*

FIGURE 3.13. *A portion of nickel foam geometry having (a)* 32^3 *(b)* 64^3 *and (c)* 128^3 *resolutions.*

Therefore, our structure generator **GeoDict**® [**93**] uses this algorithm for isosurface generation, as the triangles have a good aspect ratio and are constructed fast from grey scale images.

3.6. Limitation due to resolution

The algorithm even though rectify the usual ambiguities, it may not yield a correct topology under very coarse resolution. It may produce locally a closed component or non existing bridge when a part or two different surfaces are not overlapped within a voxel. To illustrate a 2D representation of this problem is shown in Figure 3.14. Figure 3.14(a) is the original disjoint surface within a cube and 3.14(b) is the outcome as a result of our convention. We can also see these kinds of variation in topology in Figure 3.13 under different resolutions. For these special cases one can go for a local refinement of grid or field values at the center

(a) *(b)*

FIGURE 3.14. *Topological modification of (a) given surface and (b) constructed from marching cube algorithm with our convention.*

of the cube to get exact surface representation [**91**] or use trilinear surface representation [**17**] within the cell.

CHAPTER 4

Higher order estimation of surface parameters

In the previous chapter, we described a rigorous way of constructing an isosurface by a marching cube method from a discrete level set function. During the isosurface construction, our goal is also to compute the surface properties from the level set functions accurately. In other words, if Φ is known on grid points, with $\Phi_{i,j,k} = \tilde{\Phi}(x_i, y_j, z_k), (i,j,k) \in I \subset \mathbb{Z}^3$, then we estimate the position, normal, principal curvatures, and their directions accurately on the isosurface $\tilde{\Phi} = 0$. These properties, especially the normal and the curvature are needed for various applications as discussed in Chapter 1

Currently, various techniques are used, especially for the estimation of curvature. In the finite element framework the equations are rewritten so that the second derivatives emerging from the curvature terms are avoided. The forcing term is modeled based on a weak formulation of the Laplace-Beltrami operator [**21**], [**18**], [**35**]. In the finite difference framework, central differences and interpolation or correction are used to calculate the curvature term on the surface [**11**], [**12**]. For instance, in the non-degenerate case the normal to an isosurface

$$\vec{n} = \frac{\nabla\tilde{\Phi}}{|\nabla\tilde{\Phi}|} = \frac{(\Phi_x, \Phi_y, \Phi_z)}{(\Phi_x^2 + \Phi_y^2 + \Phi_z^2)^{1/2}},$$

is approximated at grid points by finite differences. The expression is then interpolated to obtain an approximation of the normal $\vec{n}^c$ at an approximate interpolation point P_i^c [**15**].

Similarly the mean curvature

$$\begin{aligned} H &= \left(\Phi_x^2\Phi_{yy} - 2\Phi_x\Phi_y\Phi_{xy} + \Phi_y^2\Phi_{xx} + \Phi_x^2\Phi_{zz} - 2\Phi_x\Phi_z\Phi_{xz} + \right. \\ &\left. \Phi_z^2\Phi_{xx} + \Phi_y^2\Phi_{zz} - 2\Phi_y\Phi_z\Phi_{yz} + \Phi_z^2\Phi_{yy}\right)/2|\nabla\Phi|^3, \end{aligned} \tag{4.1}$$

and the Gaussian curvature

$$
\begin{aligned}
K \;=\; & \big\{\Phi_x^2(\Phi_{yy}\Phi_{zz}-\Phi_{yz}^2)+\Phi_y^2(\Phi_{xx}\Phi_{zz}-\Phi_{xz}^2)+ \\
& \Phi_z^2(\Phi_{xx}\Phi_{yy}-\Phi_{xy}^2)+2\,[\Phi_x\Phi_y(\Phi_{xz}\Phi_{yz}-\Phi_{xy}\Phi_{zz})+ \\
& \Phi_y\Phi_z(\Phi_{xy}\Phi_{xz}-\Phi_{yz}\Phi_{xx})+\Phi_x\Phi_z(\Phi_{xy}\Phi_{yz}-\Phi_{xz}\Phi_{yy})]\big\}\,/ \\
& (\Phi_x^2+\Phi_y^2+\Phi_z^2)^2, \qquad (4.2)
\end{aligned}
$$

are discretized at grid points by a higher order central difference scheme and then interpolated to the surface by cubic splines [**15**], [**85**]. This information is also used for meshing the surface geometry. For instance, Persson [**65**] estimates the curvature at regular grid points by central differences and adds a correction term from the signs of the distance function.

Here, we propose an alternative method based on weighted least squares for finding the surface parameters from a discrete level set function. With this approach, it is possible to reach an arbitrarily high order of approximation of surface properties. This is achieved by choosing a high degree of the local polynomial model and a large number of grid points (referred to as stencil) in the neighborhood of the interface.

4.1. Least squares approach

The least squares method is derived via a local polynomial model. Depending on the desired order of accuracy, the stencil and degree of the local polynomial model are chosen. Let $(\bar{x},\bar{y},\bar{z})$ be the point of interest. For local coordinates $\xi = x-\bar{x}, \eta = y-\bar{y}, \zeta = z-\bar{z}$, the mth order local polynomial in $\mathbb{R}^3$ has $l = C_m^{m+3} = (m+1)(m+2)(m+3)/6$ coefficients, where by convention the constant functions are termed 0th order polynomials. First we order the coefficients according to the order of the polynomial term and second preferring the x direction over

the y and z directions, and the y direction over the z direction, e.g.

$$\begin{aligned} f(\xi,\eta,\zeta) &= \hat{c}_0 + \hat{c}_1\xi + \hat{c}_2\eta + \hat{c}_3\zeta + \frac{\hat{c}_4}{2}\xi^2 + \frac{\hat{c}_5}{2}\xi\eta + \frac{\hat{c}_6}{2}\xi\zeta + \\ &\quad \frac{\hat{c}_7}{2}\eta^2 + \frac{\hat{c}_8}{2}\eta\zeta + \frac{\hat{c}_9}{2}\zeta^2 + \frac{\hat{c}_{10}}{6}\xi^3 + \frac{\hat{c}_{11}}{6}\xi^2\eta + \frac{\hat{c}_{12}}{6}\xi^2\zeta + \\ &\quad \frac{\hat{c}_{13}}{6}\eta^2\xi + \frac{\hat{c}_{14}}{6}\xi\eta\zeta + \frac{\hat{c}_{15}}{6}\zeta^2\xi + \frac{\hat{c}_{16}}{6}\eta^3 + \frac{\hat{c}_{17}}{6}\eta^2\zeta + \\ &\quad \frac{\hat{c}_{18}}{6}\zeta^2\eta + \frac{\hat{c}_{19}}{6}\zeta^3 + \cdots + \frac{\hat{c}_l}{m!}\xi^m. \end{aligned} \tag{4.3}$$

This can be written in generalized form as

$$f(\xi,\eta,\zeta) = \sum_{k=0}^{m} \sum_{\substack{i=0,\\ p,q,r\geq 0,\\ p+q+r=k}} \frac{1}{(p+q+r)!}\hat{c}_{\neq(p,q,r)}\xi^p\eta^q\zeta^r. \tag{4.4}$$

In (4.3), $\hat{c}_0$ is the constant term, $\hat{c}_1, \hat{c}_2, \hat{c}_3$ are the first derivatives, $\hat{c}_4$ to $\hat{c}_9$ are the second derivatives, and the remaining terms are higher order derivatives of the polynomial model. For sufficiently smooth functions $\tilde{\Phi}$, one can obtain approximate derivatives to any desired accuracy by appropriate choice of mesh width and appropriate polynomial degree m.

This can also be used to derive the expressions for finite difference approximations [**43**]. For instance, if we choose the above polynomial to be of order $m = 2$, assume the uniform mesh width h, set the point of interest to the origin $(\bar{x}, \bar{y}, \bar{z}) = (x_0, y_0, z_0)$, and require f to interpolate Φ at three points along the x direction

$$f_1 := \Phi_{0,0,0},\ f_2 := \Phi_{-1,0,0},\ f_3 := \Phi_{1,0,0},$$

then the solution of the resulting three by three linear system of equations

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & -h & h^2/2 \\ 1 & h & h^2/2 \end{pmatrix} \begin{pmatrix} \hat{c}_0 \\ \hat{c}_1 \\ \hat{c}_4 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}, \tag{4.5}$$

is given by

$$\hat{c}_0 = f_1,\ \hat{c}_1 = \frac{f_3 - f_2}{2h},\ \hat{c}_4 = \frac{f_3 - 2f_1 + f_2}{h^2}. \tag{4.6}$$

$\hat{c}_1$ and $\hat{c}_4$ are the usual central difference approximations of the first and the second derivative of a function, respectively. The matrix in (4.5) is known as a (one-dimensional) Vandermonde matrix. Similarly,

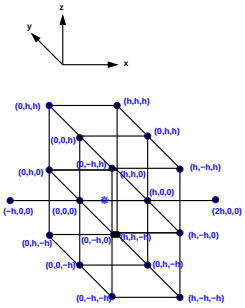


FIGURE 4.1. *The 20 points stencil for a tri-quadratic polynomial. The intersection passes through the point blue star when marching along the x-direction.*

one can also choose the stencil in a particular way and can get the mixed derivatives to the desired order of accuracy. Appendix A gives the sixth order finite difference expression for the first and the second derivatives. If a polynomial f of degree m with $l = C_m^{(m+1)}$ coefficients should approximate Φ at $N_r \geq l$ stencil points, the coefficients c should minimize

$$\min_{\hat{\mathbf{c}} \in \mathbb{R}^l} ||\mathbf{A}\hat{\mathbf{c}} - \mathbf{f}||_2, \tag{4.7}$$

where $A \in \mathbb{R}^{N_r \times l}$ is the three-dimensional Vandermonde matrix. In the case of interpolation, $\mathbf{A}$ is square (i.e. $N_r = l$) and has a non-vanishing determinant, and simply $\hat{\mathbf{c}} = \mathbf{A^{-1}f}$. The dependence of this determinant on the stencil geometry is investigated in [**47**]. When $N_r > l$, the minimum in (4.7) is in general not zero, and f does not interpolate Φ, but approximates Φ. Thus, the method to derive interpolation-based derivative information is viewed as a special case of an approximation-based setting. Recall that we consider a uniform mesh width h. To achieve $\mathcal{O}(h^3)$ error in position, $\mathcal{O}(h^2)$ error in the normal direction and $\mathcal{O}(h)$ error for the principal curvatures and directions, it is sufficient to choose a tri-quadratic polynomial as local model. For symmetry and compactness, a twenty point stencil is selected in the neighborhood of the interface point as shown in Figure 4.1. Grid function values at these stencil points are used to determine the least squares estimate of the 10 coefficients of the local tri-quadratic model. The values of f are given by

$$f_{ijk} := \Phi_{ijk} = f(ih, jh, kh). \tag{4.8}$$

The 10 unknown coefficients of the local model are overdetermined by the 20 equations at the 20 stencil points. In the x-direction, the system reads

(4.9)

$$\underbrace{\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -h & 0 & 0 & h^2/2 & 0 & 0 & 0 & 0 & 0 \\
1 & h & 0 & 0 & h^2/2 & 0 & 0 & 0 & 0 & 0 \\
1 & 2h & 0 & 0 & 2h^2 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -h & 0 & 0 & 0 & 0 & h^2/2 & 0 & 0 \\
1 & 0 & h & 0 & 0 & 0 & 0 & h^2/2 & 0 & 0 \\
1 & 0 & 0 & -h & 0 & 0 & 0 & 0 & 0 & h^2/2 \\
1 & 0 & 0 & h & 0 & 0 & 0 & 0 & 0 & h^2/2 \\
1 & h & -h & 0 & h^2/2 & -h^2/2 & 0 & h^2/2 & 0 & 0 \\
1 & h & h & 0 & h^2/2 & h^2/2 & 0 & h^2/2 & 0 & 0 \\
1 & h & 0 & -h & h^2/2 & 0 & -h^2/2 & 0 & 0 & h^2/2 \\
1 & h & 0 & h & h^2/2 & 0 & h^2/2 & 0 & 0 & h^2/2 \\
1 & 0 & -h & -h & 0 & 0 & 0 & h^2/2 & h^2/2 & h^2/2 \\
1 & h & -h & -h & h^2/2 & -h^2/2 & -h^2/2 & h^2/2 & h^2/2 & h^2/2 \\
1 & 0 & -h & h & 0 & 0 & 0 & h^2/2 & -h^2/2 & h^2/2 \\
1 & h & -h & h & h^2/2 & -h^2/2 & h^2/2 & h^2/2 & -h^2/2 & h^2/2 \\
1 & 0 & h & -h & 0 & 0 & 0 & h^2/2 & -h^2/2 & h^2/2 \\
1 & h & h & -h & h^2/2 & h^2/2 & -h^2/2 & h^2/2 & -h^2/2 & h^2/2 \\
1 & 0 & h & h & 0 & 0 & 0 & h^2/2 & h^2/2 & h^2/2 \\
1 & h & h & h & h^2/2 & h^2/2 & h^2/2 & h^2/2 & h^2/2 & h^2/2
\end{bmatrix}}_{:=\mathbf{A}}
\underbrace{\begin{bmatrix} \hat{c}_0 \\ \hat{c}_1 \\ \hat{c}_2 \\ \hat{c}_3 \\ \hat{c}_4 \\ \hat{c}_5 \\ \hat{c}_6 \\ \hat{c}_7 \\ \hat{c}_8 \\ \hat{c}_9 \end{bmatrix}}_{:=\hat{\mathbf{c}}}
=
\underbrace{\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \\ f_{11} \\ f_{12} \\ f_{13} \\ f_{14} \\ f_{15} \\ f_{16} \\ f_{17} \\ f_{18} \\ f_{19} \\ f_{20} \end{bmatrix}}_{:=\mathbf{f}}.$$

For each direction, the system can be expressed in the form $\mathbf{A}\hat{\mathbf{c}} = \mathbf{f}$ with $\mathbf{A} \in \mathbb{R}^{20\times 10}$, $\hat{\mathbf{c}} \in \mathbb{R}^{10}$ and $\mathbf{f} \in \mathbb{R}^{20}$. This in turn is solved in the least squares sense (4.7) by $\hat{\mathbf{c}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{f} = \mathbf{M}^T\mathbf{f}$, where $\mathbf{M} \in \mathbb{R}^{10\times 20}$ depends only on the stencil.

4.1.1. Stencil selection and weighted least squares. In the computation by least squares, selecting a good stencil is crucial. The two pitfalls are large stencils and singular $\mathbf{A^T A}$. Lorentz in [**47**] gives a closed expression for computing the determinant of $\mathbf{A^T A}$ for the 2D case, which could be extended to 3D models. Instead, in our case of a uniform grid and fixed stencils, we simply evaluate the determinant apriori. We choose the stencil by looking in the neighborhood of the interface, and then increase the search radius until $(\mathbf{A^T A})^{-1}$ is regular [**79**].

Distance-dependant weights ensure that the approximation is better at points close to the reference point $(\bar{x}, \bar{y}, \bar{z})$ at the cost of deteriorating the approximation at points further-away. In this weighted least

squares approach, $\hat{\mathbf{c}}$ is given as $(\mathbf{A}^T\mathbf{W}^2\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}^2\mathbf{f}$, where $\mathbf{W}$ is a diagonal matrix of size N_r. In the usual weighted least squares approach, weights are based on the Euclidean distance from the reference point. In our approach, we use weights based on Φ as in [**79**], i.e.

$$\mathbf{W}_{ij} = \begin{cases} \dfrac{h^2}{h^2 + |\Phi_{ij}|} & \text{when } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

Hence the weights are large near the interface and small far away from it. The weights based on Φ give more strength to our method as they limit the stencil to a narrow band around the interface. This is in contrast to a high order finite difference approach which selects points far away from the interface, since the stencil is chosen along certain preferred direction (see Appendix A). In a general setting, when the interface is moving with velocity $\mathbf{v}$, WENO methods choose the weights not only based on Φ, but also in accordance with *upwinding* directions.

4.1.2. Surface information estimation from a tri-quadratic local model. If the polynomial is tri-quadratic, an approximate implicit representation of the surface in local coordinates with reference point (x_i, y_j, z_k) is given by

$$\begin{aligned} f(\xi,\eta,\zeta) &= \hat{c}_0 + \hat{c}_1\xi + \hat{c}_2\eta + \hat{c}_3\zeta + \frac{\hat{c}_4}{2}\xi^2 + \\ &\quad \frac{\hat{c}_5}{2}\xi\eta + \frac{\hat{c}_6}{2}\xi\zeta + \frac{\hat{c}_7}{2}\eta^2 + \frac{\hat{c}_8}{2}\eta\zeta + \frac{\hat{c}_9}{2}\zeta^2 = 0. \end{aligned} \tag{4.11}$$

Let us consider the situation as in Figure 4.1, where the intersection point lies between two indices along the x direction. That is $\eta = 0$ and $\zeta = 0$ are fixed to identify a grid line. The intersection point may lie between (x_i, y_j, z_k) and (x_{i+1}, y_j, z_k) if either $\Phi_{i,j,k} > 0$ and $\Phi_{i+1,j,k} \leq 0$ or $\Phi_{i,j,k} \leq 0$ and $\Phi_{i+1,j,k} > 0$. Candidates for the local x-coordinate ξ^c_* of the intersection point $(\xi^c_*, 0, 0)$ are the roots of the quadratic equation

$$\hat{c}_0 + \hat{c}_1\xi + \frac{\hat{c}_4}{2}\xi^2 = 0. \tag{4.12}$$

These roots are

$$\xi^c_* = \begin{cases} -\frac{\hat{c}^c_1}{\hat{c}_4} \pm \sqrt{(\frac{\hat{c}_1}{\hat{c}_4})^2 - \frac{2\hat{c}_0}{\hat{c}_4}}, & \text{if } \hat{c}_4 \neq 0, \\ -\frac{\hat{c}_0}{\hat{c}_1} & \text{otherwise.} \end{cases} \tag{4.13}$$

Because the local polynomial does not interpolate the values $\Phi_{i,j,k}$ nor $\Phi_{i+1,j,k}$, f may not reproduce the sign change of Φ. To ensure $\xi_*^c \in (0,h]$, we need a linear correction.

REMARK 7. *Linear Correction: From the first row of equation (4.9), we observe that $\hat{c}_0 = f_1$. When solving in the sense of least squares, it may happen that $\hat{c}_0$ is not equal to f_1, resulting in $\xi_*^c \notin (0,h]$. To ensure $\xi_*^c \in (0,h]$, we correct the coefficients $\hat{c}_0$ and $\hat{c}_1$ by a linear function. If the intersection lies along the x direction, the corrected coefficients $\hat{c}_0^c$ and $\hat{c}_1^c$ are*

$$\hat{c}_0^c = \hat{c}_0 - g_0, \tag{4.14a}$$

$$\hat{c}_1^c = \hat{c}_1 - (g_1 - g_0)/h, \tag{4.14b}$$

where $g_0 := \hat{c}_0 - f_1$ and $g_1 := \hat{c}_0 + h\hat{c}_1 + \frac{\hat{c}_4}{2}h^2 - f_3$. This will make

$$\hat{c}_0^c = f_1 \text{ and } \hat{c}_1^c = \frac{\hat{c}_4}{2}h + \frac{f_3 - f_1}{h}. \tag{4.15}$$

Similar relations also hold for the y and z directions.

REMARK 8. *We use the corrected coefficients $\hat{c}_0^c, \hat{c}_1^c, \hat{c}_2^c$ and $\hat{c}_3^c$, only for the estimation of the interface position. For the estimation of the normal, we use $\hat{c}_1, \hat{c}_2$, and $\hat{c}_3$.*

The correction ensures that (4.13) has a unique root in $(0,h]$.

THEOREM 2. *Given $\Phi(x_i, y_j, z_k), (i,j,k) \in I \subset \mathbb{Z}^3$, such that the interface point lies between $\Phi_{i,j,k} \leq 0 (\Phi_{i,j,k} > 0)$ and $\Phi_{i+1,j,k} > 0 (\Phi_{i+1,j,k} \leq 0)$, and the coefficients are linearly corrected by equations (4.14a) and (4.14b), then there exists one and only one intersection point $(\xi_*^c, 0, 0)$ satisfying equation (4.11).*

PROOF. If $\hat{c}_4 = 0$, there is only one root because the equation is linear and $\hat{c}_1^c \neq 0$. Otherwise, it will violate the change in sign conditions between (i,j,k) and $(i+1,j,k)$. If $\hat{c}_4 \neq 0$, the polynomial equation is quadratic. We should first prove that the polynomial of order less than two i.e., first order does not vanish. That is, the derivative of the quadratic polynomial should not vanish in the interval $(0,h]$. This means

$$\hat{c}_1^c + \hat{c}_4 h \neq 0.$$

In other words, if $\hat{c}_1^c < 0$, then $\hat{c}_1^c + \hat{c}_4 h$ should also be less than zero or if $\hat{c}_1^c > 0$, then $\hat{c}_1^c + \hat{c}_4 h$ should also be greater than zero. From the linear correction equation, we find

$$\hat{c}_1^c + \hat{c}_4 h \;=\; 2(f_3 - f_1), \tag{4.16}$$

which cannot be equal since equation (4.16) implies $\Phi(i,j,k) = \Phi(i+1,j,k)$, violating the change in sign condition. It is also seen when $\hat{c}_1^c < 0$, then

$$\Phi(i+1,j,k) - \Phi(i,j,k) = f_3 - f_1 < 0 \;\Rightarrow \hat{c}_1^c + \hat{c}_4 h < 0,$$

and vice versa. Similarly, the intersection point along the y and z directions can be shown to be unique. □

For the estimation of the surface properties at the intersection point P_i^c, we follow four steps:[1]

(1) *Normal $\vec{n}$:* The normal, defined as $\vec{n} = \frac{\nabla\Phi}{\|\nabla\Phi\|_2}$ can be written in terms of the coefficients of the local polynomial. For the tri-quadratic polynomial, the approximate normal at the intersection point $(\xi_*^c, 0, 0)$, can be obtained through $\nabla f \approx \nabla\Phi$, given by

$$\vec{n}^c \;=\; [\hat{c}_1 + \hat{c}_4\xi_*^c, \hat{c}_2 + \hat{c}_5\xi_*^c/2, \hat{c}_3 + \hat{c}_6\xi_*^c/2]^T. \tag{4.17}$$

(2) *Normal curvature $\mathcal{H}^c$:* The normal curvature is found from the Hessian $\nabla\cdot\nabla f \approx \nabla\cdot\nabla\Phi(x_i + \xi_*^c, y_j, z_k)$. For our tri-quadratic polynomial approximation, it reads

$$\mathcal{H}^c = \begin{bmatrix} \hat{c}_4 & \hat{c}_5 & \hat{c}_6 \\ \hat{c}_5 & \hat{c}_7 & \hat{c}_8 \\ \hat{c}_6 & \hat{c}_8 & \hat{c}_9 \end{bmatrix}.$$

(3) *Tangent plane T_p:* Now let $\vec{t}$ and $\vec{s}$ be orthonormal tangents to $\vec{n}^c$, i.e. $\vec{n}^c\cdot\vec{t} = \vec{n}^c\cdot\vec{s} = \vec{s}\cdot\vec{t} = 0$, $\|\vec{s}\|_2 = \|\vec{t}\|_2 = 1$. These two vectors span the tangent plane.

[1]For the definition and analysis of surface properties we refer to [**19**], [**29**]

(4) *Principal curvatures λ_1^c, and λ_2^c:* These are the eigenvectors of the Weingarten's matrix $\mathbf{a}^c$ [**33**] with

$$\mathbf{a^c} = \begin{bmatrix} a_{11}^c & a_{12}^c \\ a_{12}^c & a_{22}^c \end{bmatrix} = \begin{bmatrix} \vec{t} \\ \vec{s} \end{bmatrix} \mathcal{H}^c \begin{bmatrix} \vec{t} & \vec{s} \end{bmatrix}$$

$$= \begin{bmatrix} t_1 & t_2 & t_3 \\ s_1 & s_2 & s_3 \end{bmatrix} \begin{bmatrix} \hat{c}_4 & \hat{c}_5 & \hat{c}_6 \\ \hat{c}_5 & \hat{c}_7 & \hat{c}_8 \\ \hat{c}_6 & \hat{c}_8 & \hat{c}_9 \end{bmatrix} \begin{bmatrix} t_1 & s_1 \\ t_2 & s_2 \\ t_3 & s_3 \end{bmatrix}.$$

The Eigenvalues of $\mathbf{a^c}$ are given by

$$\lambda_1^c = \frac{a_{22}^c + a_{11}^c}{2} + \sqrt{\frac{(a_{11}^c - a_{22}^c)^2}{4} + (a_{12}^c)^2} \text{ and} \tag{4.18a}$$

$$\lambda_2^c = \frac{a_{22}^c + a_{11}^c}{2} - \sqrt{\frac{(a_{11}^c - a_{22}^c)^2}{4} + (a_{12}^c)^2}, \tag{4.18b}$$

with Eigenvectors

$$\vec{e}_1 = \begin{bmatrix} a_{12}^c \\ \lambda_1^c \end{bmatrix}, \vec{e}_2 = \begin{bmatrix} \lambda_1^c \\ -a_{12}^c \end{bmatrix}. \tag{4.19}$$

The Eigenvalues λ_1^c, λ_2^c are the approximate principal curvature with corresponding principal direction estimates $\vec{pd}_1^c$ and $\vec{pd}_2^c$ given by

$$\begin{bmatrix} pd_{11}^c & pd_{21}^c \\ pd_{12}^c & pd_{22}^c \\ pd_{13}^c & pd_{23}^c \end{bmatrix} = \begin{bmatrix} t_1 & s_1 \\ t_2 & s_2 \\ t_3 & s_3 \end{bmatrix} \begin{bmatrix} a_{12}^c & \lambda_1^c \\ \lambda_1^c & -a_{12}^c \end{bmatrix}. \tag{4.20}$$

REMARK 9. **Mean and Gaussian curvature**
The approximate mean curvature can be found from

$$H^c = \frac{1}{2} trace(\mathbf{a}^c) = \frac{a_{22}^c + a_{11}^c}{2},$$

and the approximate Gaussian curvature from

$$K^c = det(\mathbf{a}^c) = a_{11}^c a_{22}^c - (a_{12}^c)^2.$$

Hence,

$$\lambda_{1,2}^c = H^c \pm \sqrt{(H^c)^2 - K^c} \tag{4.21}$$

is the relation of the principal curvature to the mean and Gaussian curvatures.

4.1.3. Surface properties from a general order local model: In subsection 4.1.2, we confine our discussion to the tri-quadratic polynomial case. In a general situation, the intersection point $P_i^c(\xi_*^c, \eta_*^c, \zeta_*^c)$ along the prescribed direction is found from

$$\begin{aligned} P_i^c = f(\xi,\eta,\zeta) &= \hat{c}_0 + \hat{c}_1\xi + \hat{c}_2\eta + \hat{c}_3\zeta + \frac{\hat{c}_4}{2}\xi^2 + \cdots + \\ &\frac{\hat{c}_{10}}{6}\xi^3 + \cdots + \frac{\hat{c}_l}{m!}\xi^m + \cdots = 0. \end{aligned} \tag{4.22}$$

For $m > 5$, a simple closed relation is not possible. Hence, one can form a companion matrix (even for $m < 5$) to get the roots from the eigenvalues. The coefficients are corrected as in the quadratic case by equation (4.14), with

$$g_0 = f_1 - \hat{c}_0,$$

and

$$g_1 = \hat{c}_0 + \hat{c}_1 h + \hat{c}_2 h + \hat{c}_3 h + \frac{\hat{c}_4}{2}h^2 + \cdots + \frac{\hat{c}_l}{m!}h^m + \ldots$$

The unique root is determined from the condition that it is real and it lies between the change in signs of Φ. If we follow the similar convention of polynomial expansion as in equation (4.3), the normal estimation is $\vec{n}^c = (n_x^c, n_y^c, n_z^c)$ where

$$\begin{aligned} n_x^c &= \hat{c}_1 + \frac{1}{2!}\left[2\hat{c}_4\xi_*^c + \hat{c}_5\eta_*^c + \hat{c}_6\zeta_*^c\right] + \frac{1}{3!}[3\hat{c}_{10}(\xi_*^c)^2 + \ldots] + \cdots + \\ &\frac{1}{(m-1)!}\hat{c}_l(\xi_*^c)^{m-1} + \ldots, \end{aligned} \tag{4.23a}$$

$$\begin{aligned} n_y^c &= \hat{c}_2 + \frac{1}{2!}[\hat{c}_5\xi_*^c + 2\hat{c}_7\eta_*^c + \hat{c}_8\zeta_*^c] + \frac{1}{3!}[\hat{c}_{11}(\xi_*^c)^2 + \ldots] + \cdots + \\ &\frac{1}{m!}\hat{c}_{l+1}(\xi_*^c)^{m-1} + \ldots \text{ and} \end{aligned} \tag{4.23b}$$

$$\begin{aligned} n_z^c &= \hat{c}_3 + \frac{1}{2}[\hat{c}_6\xi_*^c + \hat{c}_8\eta_*^c + 2\hat{c}_9\zeta_*^c] + \frac{1}{3}[\hat{c}_{12}(\xi_*^c)^2 + \ldots] + \cdots + \\ &\frac{1}{m!}\hat{c}_{l+2}(\xi_*^c)^{m-1} + \ldots. \end{aligned} \tag{4.23c}$$

The normal curvature, defined through the Hessian $\mathcal{H}^c$ by the directional derivative of the normal $\nabla \cdot \nabla f \approx \nabla \cdot \nabla \Phi(x_i + \xi_*^c, y_j, z_k)$, is

$$\mathcal{H}^c = \begin{bmatrix} \mathcal{H}_{11}^c & \mathcal{H}_{12}^c & \mathcal{H}_{13}^c \\ \mathcal{H}_{12}^c & \mathcal{H}_{22}^c & \mathcal{H}_{23}^c \\ \mathcal{H}_{13}^c & \mathcal{H}_{23}^c & \mathcal{H}_{33}^c \end{bmatrix}, \tag{4.24a}$$

where

$$\mathcal{H}^c_{11} = \hat{c}_4 + \frac{1}{3!}[6\hat{c}_{10}\xi^c_* + 2\hat{c}_{11}\eta^c_* + 2\hat{c}_{12}\zeta^c_*] + \cdots + \frac{\hat{c}_l}{(m-2)!}(\xi^c_*)^{m-2} + \ldots, \tag{4.24b}$$

$$\mathcal{H}^c_{12} = \frac{1}{2}\hat{c}_5 + \frac{1}{3!}[\hat{c}_{11}\xi^c_* + 2\hat{c}_{13}\eta^c_* + \hat{c}_{14}\zeta^c_*] + \cdots + \frac{(m-1)\hat{c}_{l+1}}{m!}(\xi^c_*)^{m-2} + \ldots, \tag{4.24c}$$

$$\mathcal{H}^c_{13} = \frac{1}{2}\hat{c}_6 + \frac{1}{3!}[2\hat{c}_{12}\xi^c_* + \hat{c}_{14}\eta^c_* + 2\hat{c}_{15}\zeta^c_*] + \cdots + \frac{(m-1)\hat{c}_{l+2}}{m!}(\xi^c_*)^{m-2} + \ldots, \tag{4.24d}$$

$$\mathcal{H}^c_{22} = \hat{c}_7 + \frac{1}{3!}[2\hat{c}_{13}\xi^c_* + 6\hat{c}_{16}\eta^c_* + 2\hat{c}_{17}\zeta^c_*] + \cdots + \frac{2\hat{c}_{l+3}}{m!}(\xi^c_*)^{m-2} + \ldots, \tag{4.24e}$$

$$\mathcal{H}^c_{23} = \frac{1}{2}\hat{c}_8 + \frac{1}{3!}[\hat{c}_{14}\xi^c_* 2 + 2\hat{c}_{17}\eta^c_* + 2\hat{c}_{18}\zeta^c_*] + \cdots + \frac{\hat{c}_{l+4}}{m!}(\xi^c_*)^{m-2} + \ldots \text{ and} \tag{4.24f}$$

$$\mathcal{H}^c_{33} = \hat{c}_9 + \frac{1}{3!}[2\hat{c}_{15}\xi^c_* + 2\hat{c}_{18}\eta^c_* + 6\hat{c}_{19}\zeta^c_*] + \cdots + \frac{(m-2)\hat{c}_{l+5}}{m!}(\xi^c_*)^{m-2} + \ldots. \tag{4.24g}$$

4.1.4. Convergence of the surface parameters with respect to the grid resolution. We introduce some notations and assumptions before proving the convergence of surface parameters. We denote the order of a polynomial by $O()$ and the order of convergence by $\mathcal{O}()$. The exact surface properties *viz.,* the intersection point is represented by $P_i(\xi_*, \eta_*, \zeta_*)$, the normal by $\vec{n}$, the mean curvature by H, and the Gaussian curvature by K. The corresponding approximate values are $P^c_i(\xi^c_*, \eta^c_*, \zeta^c_*)$, $\vec{n}^c$, H^c, and K^c, respectively. For the Hessian, the exact components are represented by $\mathcal{H}_{11}, \mathcal{H}_{12}, \ldots, \mathcal{H}_{33}$ corresponding to

their approximate values $\mathcal{H}_{11}^c, \mathcal{H}_{12}^c, \ldots, \mathcal{H}_{33}^c$, respectively. The error terms in the Hessian components are defined by

$$\epsilon_{11} := |\mathcal{H}_{11} - \mathcal{H}_{11}^c|,\ \epsilon_{12} := |\mathcal{H}_{12} - \mathcal{H}_{12}^c|,\ \ldots\ , \epsilon_{33} := |\mathcal{H}_{33} - \mathcal{H}_{33}^c|.$$

Let $c_0, c_1, c_2, \ldots, c_l$, be the exact coefficients corresponding to their approximate values $\hat{c}_0^c, \hat{c}_1, \hat{c}_2, \ldots, \hat{c}_l$. The error terms due to the approximation of coefficients are defined by

$$e_0 := c_0 - \hat{c}_0^c,\ e_1 := c_1 - \hat{c}_1,\ \ldots,\ e_l := c_l - \hat{c}_l.$$

Similarly, let a_{11}, a_{12} and a_{22} be the elements of the exact Weingarten matrix corresponding to their approximate elements a_{11}^c, a_{12}^c and a_{22}^c.

THEOREM 3. *Let f be the mth order polynomial which approximates the smooth function $\Phi : \mathbb{R}^3 \to \mathbb{R}$ in the least squares sense. Then, the estimated surface properties, through the sequence of functions f_h where h is the uniform mesh width, converge to the exact surface properties with the order of*

(i) $\mathcal{O}(h^{m+1})$ *for the interface position,*
(ii) $\mathcal{O}(h^m)$ *for the normal,*
(iii) $\mathcal{O}(h^{(m-1)})$ *for the mean curvature,*
(iv) $\mathcal{O}(h^{(m-1)})$ *for the Gaussian curvature,*
(v) $\mathcal{O}(h^{(m-1)})$ *for the principal curvatures and*
(vi) $\mathcal{O}(h^{(m-1)})$ *for the principal directions.*

PROOF. (i) *Interface position:*
Let us consider the intersection in x direction. Then, ξ_* can be found from

$$c_0 + c_1\xi + \frac{1}{2}c_4(\xi)^2 + \cdots + \frac{1}{m!}c_l(\xi)^m = 0, \tag{4.25}$$

and ξ_*^c from

$$\hat{c}_0^c + \hat{c}_1\xi^c + \frac{1}{2}\hat{c}_4(\xi)^2 + \cdots + \frac{1}{m!}\hat{c}_l(\xi)^m = 0, \tag{4.26}$$

where $l = m(m+1)(m+2)/6$.

The term $\hat{c}_0$ is the constant, $(\hat{c}_1, \hat{c}_2, \hat{c}_3)$ are the first derivatives, $(\hat{c}_4, \ldots, \hat{c}_9)$ are the second derivatives, and the remaining terms are higher order derivatives. If the approximate interface position is of $O(h^{m+1})$ then,

$$|\hat{c}_0^c - c_0| = O(h^{m+1}).$$

Similarly, if the first derivative is approximated by $O(h^m)$ then,

$$|\hat{c}_1 - c_1| = O(h^m),$$

and if the second derivative is of $O(h^{m-1})$ then,

$$|\hat{c}_4 - c_4| = O(h^{m-1}).$$

Proceeding in this manner, we find

$$|\hat{c}_l - c_l| = O(h).$$

For the estimation of the interface, we first ensure that the linearly corrected $\hat{c}_1$ does not alter the order of convergence. Therefore, from equation (4.14b) we find

$$(\hat{c}_1^c - c_1)\xi = (\frac{1}{2}c_4 h + \frac{1}{3!}c_{10}h^2 + \cdots + \frac{1}{m}c_l^{m-1} - \frac{f_1 - f_3}{h} - c_1)\xi$$

being of $O(h^{m+1})$ since $\xi \in (0, h]$ and f_1, f_3 are Φ values at the grid points. Therefore, the root estimated from equation (4.26) can be written as

$$\begin{aligned}(c_0 - e_0) + (c_1 - e_1)\xi + \frac{1}{2}(c_4 - e_4)\xi^2 \\ + \cdots + \frac{1}{m!}(c_l - e_l)\xi^m = 0.\end{aligned} \tag{4.27}$$

Separating the error from the exact terms yields

$$\begin{aligned}c_0 + c_1\xi + \frac{1}{2}c_4\xi^2 + \cdots + \frac{1}{m!}c_l\xi^m - \\ (e_0 + e_1\xi + \frac{1}{2}e_4\xi^2 + \cdots + \frac{1}{m!}e_l\xi^m) &= 0.\end{aligned} \tag{4.28}$$

As $\xi_* \in (0, h]$, each error term in equation (4.28) is $O(h^{m+1})$. Therefore,

$$\boxed{|\xi_* - \xi_*^c| \approx \mathcal{O}(h^{m+1}).} \tag{4.29}$$

Similar results hold for the intersection in y and z direction.

(ii) *Normal:*

The approximate normal n_x^c at $P_i^c(\xi_*^c, \eta_*^c, \zeta_*^c)$ along the x-direction is given by

$$\begin{aligned}n_x^c &= \hat{c}_1 + \frac{1}{2}[2\hat{c}_4\xi_*^c + \hat{c}_5\eta_*^c + \hat{c}_6\zeta_*^c] + \frac{1}{3!}[3\hat{c}_{10}(\xi_*^c)^2 + \ldots] + \\ &\frac{1}{4!}[4\hat{c}_{20}(\xi_*^c)^3 + \ldots] + \cdots + \frac{\hat{c}_l}{(m-1)!}(\xi_*^c)^{m-1}.\end{aligned} \tag{4.30}$$

The exact normal n_x at $P_i(\xi_*, \eta_*, \zeta_*)$ along the x-direction is

$$\begin{aligned} n_x &= c_1 + \frac{1}{2}[2c_4\xi_* + c_5\eta_* + c_6\zeta_*] + \frac{1}{3!}[3c_{10}(\xi_*)^2 + \dots] + \\ &\frac{1}{4!}[4c_{20}(\xi_*)^3 + \dots] + \cdots + \frac{c_l}{(m-1)!}(\xi_*)^{m-1}. \end{aligned} \tag{4.31}$$

As $|\hat{c}_1 - c_1| = O(h^m)$, $|\hat{c}_4 - c_4| = O(h^{m-1}), \dots$, we can compute the error in the estimation of the normal by

$$\begin{aligned} |n_x^c - n_x| &= \Big| e_1 + \frac{1}{2}\,[2\hat{c}_4(\xi_*^c - \xi_*) - 2\hat{e}_4\xi_* + \\ &\quad c_5(\eta_*^c - \eta_*) - e_5\eta_* + c_6(\zeta_*^c - \zeta_*) - e_6\zeta_*] + \\ &\quad \frac{1}{3!}[3c_{10}((\xi_*^c)^2 - (\xi_*)^2) - 3e_{10}(\xi_*)^2 + \dots] + \\ &\quad \frac{1}{4!}[4(c_{20}((\xi_*^c)^3 - (\xi_*)^3) + \dots] + \cdots + \\ &\quad \frac{1}{(m-1)!}[c_l((\xi_*^c)^{m-1} - (\xi_*)^{m-1}) + \dots] \Big|. \end{aligned} \tag{4.32}$$

Since $\xi_*, \eta_*, \zeta_* \in (0, h]$, we immediately see that each error term in the equation (4.32) is $O(h^m)$, i.e.

$$\boxed{|n_x - n_x^c| \approx \mathcal{O}(h^m).} \tag{4.33}$$

Likewise, we can show that the order of convergence is $\mathcal{O}(h^m)$ for n_y and n_z.

(iii) *Mean curvature:*
We can write the approximate Hessian entries as

$$\begin{aligned}
\mathcal{H}^c_{11} &= (c_4 - e_4) + \frac{1}{3!}\,[6(c_{10} - e_{10})\xi^c_* + 2(c_{11} - e_{11})\eta^c_* + \\
&\quad 2(c_{12} - e_{12})\zeta^c_*] + \cdots + \frac{c_l - e_l}{(m-2)!}(\xi^c_*)^{m-2} \\
(4.34a) \quad &= \mathcal{H}_{11} - \epsilon_{11}, \\
\mathcal{H}^c_{12} &= \frac{1}{2}(c_5 - e_5) + \frac{1}{3!}\,[2(c_{11} - e_{11})\xi^c_* + 2(c_{14} - e_{14})\eta^c_* + \\
&\quad (c_{19} - e_{19})\eta^c_*] + \cdots + \frac{(m-1)(c_{l+1} - e_{l+1})}{m!}(\xi^c_*)^{m-2} \\
(4.34b) \quad &= \mathcal{H}_{12} - \epsilon_{12}, \\
\mathcal{H}^c_{13} &= \frac{1}{2}(c_6 - e_6) + \frac{1}{3!}\,[2(c_{12} - e_{12})\xi^c_* + 2(c_{17} - e_{17})\zeta^c_* + \\
&\quad (c_{19} - e_{19})\eta^c_*] + \cdots + \frac{(m-1)(c_{l+2} - e_{l+2})}{m!}(\xi^c_*)^{m-2} \\
(4.34c) \quad &= \mathcal{H}_{13} - \epsilon_{13}, \\
\mathcal{H}^c_{22} &= (c_7 - e_7) + \frac{1}{3!}\,[6(c_{13} - e_{13})\eta^c_* + 2(c_{14} - e_{14})\xi^c_* + \\
&\quad 2(c_{15} - e_{15})\zeta^c_*] + \cdots + \frac{(m-2)(c_{l+3} - e_{l+3})}{m!}(\xi^c_*)^{m-2} \\
(4.34d) \quad &= \mathcal{H}_{22} - \epsilon_{22}, \\
\mathcal{H}^c_{23} &= \frac{1}{2}(c_8 - e_8) + \frac{1}{3!}\,[(c_{19} - e_{19})\xi^c_* + 2(c_{15} - e_{15})\eta^c_* + \\
&\quad 2(c_{18} - e_{18})\zeta^c_*] + \cdots + \frac{(c_{l+5} - e_{l+5})}{m!}(\xi^c_*)^{m-2} \\
(4.34e) \quad &= \mathcal{H}_{23} - \epsilon_{23} \text{ and} \\
\mathcal{H}^c_{33} &= (c_9 - e_9) + \frac{1}{3!}\,[(c_{15} - e_{15})\xi^c_* + (c_{16} - e_{16})\zeta^c_* + \\
&\quad (c_{18} - e_{18})\eta^c_*] + \cdots + \frac{(m-2)(c_{l+4} - e_{l+4})}{m!}(\xi^c_*)^{m-2} \\
(4.34f) \quad &= \mathcal{H}_{33} - \epsilon_{33}.
\end{aligned}$$

The approximate Weingarten matrix $\mathbf{a}^c$ is then given by

$$(4.35) \quad \begin{bmatrix} a^c_{11} & a^c_{12} \\ a^c_{12} & a^c_{22} \end{bmatrix} = \begin{bmatrix} \vec{s} \\ \vec{t} \end{bmatrix} \mathcal{H} \begin{bmatrix} \vec{s} & \vec{t} \end{bmatrix} - \begin{bmatrix} \vec{s} \\ \vec{t} \end{bmatrix} \epsilon \begin{bmatrix} \vec{s} & \vec{t} \end{bmatrix},$$

where $\vec{s} = (s_1, s_2, s_3)$, $\vec{t} = (t_1, t_2, t_3)$ and

$$\epsilon = \begin{bmatrix} \epsilon_{11} & \epsilon_{12} & \epsilon_{13} \\ \epsilon_{12} & \epsilon_{22} & \epsilon_{23} \\ \epsilon_{13} & \epsilon_{23} & \epsilon_{33} \end{bmatrix}. \tag{4.36}$$

The approximate mean curvature $H^c = \frac{1}{2} trace(\mathbf{a}^c)$, i.e.

$$\begin{aligned} \frac{a^c_{11} + a^c_{22}}{2} &= \frac{(s_1^2 + t_1^2)(\mathcal{H}_{11} - \epsilon_{11})}{2} + \frac{(s_2^2 + t_2^2)(\mathcal{H}_{22} - \epsilon_{22})}{2} + \\ &\quad \frac{(s_3^2 + t_3^2)(\mathcal{H}_{33} - \epsilon_{33})}{2} + \frac{(s_1 s_2 + t_1 t_2)(\mathcal{H}_{12} - \epsilon_{12})}{2} + \\ &\quad \frac{(s_1 s_3 + t_1 t_3)(\mathcal{H}_{13} - \epsilon_{13})}{2} + \frac{(s_2 s_3 + t_2 t_3)(\mathcal{H}_{23} - \epsilon_{23})}{2} \\ &:= A - A_e, \end{aligned} \tag{4.37}$$

where

$$\begin{aligned} A &= \frac{(s_1^2 + t_1^2)\mathcal{H}_{11}}{2} + \frac{(s_2^2 + t_2^2)\mathcal{H}_{22}}{2} + \frac{(s_3^2 + t_3^2)\mathcal{H}_{33}}{2} + \\ &\quad \frac{(s_1 s_2 + t_1 t_2)\mathcal{H}_{12}}{2} + \frac{(s_1 s_3 + t_1 t_3)\mathcal{H}_{13}}{2} \frac{(s_2 s_3 + t_2 t_3)\mathcal{H}_{23}}{2} \end{aligned} \tag{4.38}$$

and

$$\begin{aligned} A_e &= \left[\frac{(s_1^2 + t_1^2)\epsilon_{11}}{2} + \frac{(s_2^2 + t_2^2)\epsilon_{22}}{2} + \frac{(s_3^2 + t_3^2)\epsilon_{33}}{2} + \right. \\ &\quad \left. \frac{(s_1 s_2 + t_1 t_2)\epsilon_{12}}{2} + \frac{(s_1 s_3 + t_1 t_3)\epsilon_{13}}{2} + \frac{(s_2 s_3 + t_2 t_3)\epsilon_{23}}{2} \right]. \end{aligned} \tag{4.39}$$

The error in the estimation of the mean curvature is

$$|H - H^c| = |\frac{a_{11} + a_{22} - a^c_{11} - a^c_{22}}{2}| = A_e. \tag{4.40}$$

We observe that the error terms $\epsilon_{11} \ldots \epsilon_{33}$ of A_e from equations (4.34(a) - 4.34(f)) is $O(h^{m-1})$. Since $s_1, s_2, s_3, t_1, t_2, t_3 \in [0, 1]$, we have

$$\boxed{|H - H^c| \approx \mathcal{O}(h^{m-1}).} \tag{4.41}$$

(iv) *Gaussian curvature:*

The estimated Gaussian curvature K^c from equation (4.18) is $a^c_{11} a^c_{12} -$

$(a_{12}^c)^2$. This is given by

$$\begin{aligned} K^c &= ((\mathcal{H}_{11}-\epsilon_{11})(\mathcal{H}_{22}-\epsilon_{22})-(\mathcal{H}_{12}-\epsilon_{12})^2)(s_1t_2-s_2t_1)^2+ \\ &\quad (\mathcal{H}_{11}-\epsilon_{11})(\mathcal{H}_{33}-\epsilon_{33})-(\mathcal{H}_{13}-\epsilon_{13})^2)(s_3t_3-s_1t_3)^2+ \\ &\quad (\mathcal{H}_{22}-\epsilon_{22})(\mathcal{H}_{33}-\epsilon_{33})-(\mathcal{H}_{23}-\epsilon_{23})^2)(s_3t_2-s_2t_3)^2+ \\ &\quad 2((\mathcal{H}_{23}-\epsilon_{23})(\mathcal{H}_{11}-\epsilon_{11})-(\mathcal{H}_{12}-\epsilon_{12})(\mathcal{H}_{13}-\epsilon_{13})) \\ &\quad (t_1^2s_2s_3+s_1^2t_2t_3-t_1t_2s_1s_3-t_1t_3s_1s_2)+ \\ &\quad 2((\mathcal{H}_{12}-\epsilon_{12})(\mathcal{H}_{23}-\epsilon_{23})-(\mathcal{H}_{12}-\epsilon_{12})(\mathcal{H}_{13}-\epsilon_{13})) \\ &\quad (t_1t_2s_2s_3-t_1t_3s_2^2-t_2^2s_1s_3+t_2t_3s_1s_2). \end{aligned} \tag{4.42}$$

The error in the Gaussian curvature has two parts. The first term is the product of error and the exact Hessian such as $\mathcal{H}_{11}\epsilon_{22}$. The second is the product within the error terms such as $\epsilon_{11}\epsilon_{22}$. Therefore,

$$|K-K^c| = |K-(K_1+K_2)|, \tag{4.43}$$

where K_1 is

$$\begin{aligned} K_1 &= (s_1t_2-s_2t_1)^2(\epsilon_{11}\mathcal{H}_{22}+\epsilon_{22}\mathcal{H}_{11}-2\epsilon_{12}\mathcal{H}_{12})+(s_3t_3-s_1t_3)^2 \\ &\quad (\epsilon_{33}\mathcal{H}_{11}+\epsilon_{11}\mathcal{H}_{33}-2\epsilon_{13}\mathcal{H}_{13})+(s_3t_2-s_2t_3)^2(\epsilon_{33}\mathcal{H}_{22}+ \\ &\quad \epsilon_{22}\mathcal{H}_{33}-2\epsilon_{23}\mathcal{H}_{23})+2(\epsilon_{11}\mathcal{H}_{23}+\epsilon_{23}\mathcal{H}_{11}-\epsilon_{13}\mathcal{H}_{12}-\epsilon_{12}\mathcal{H}_{13}) \\ &\quad (t_1^2s_2s_3+s_1^2t_2t_3-t_1t_2s_1s_3-t_1t_3s_1s_2)+2(\epsilon_{23}\mathcal{H}_{12}+\epsilon_{12}\mathcal{H}_{23}- \\ &\quad \epsilon_{12}\mathcal{H}_{13}-\epsilon_{13}\mathcal{H}_{12})(t_1^2s_2s_3+s_1^2t_2t_3-t_1t_2s_1s_3-t_1t_3s_1s_2) \end{aligned} \tag{4.44}$$

and K_2 is

$$\begin{aligned} K_2 &= (\epsilon_{11}\epsilon_{22}-\epsilon_{12}^2)(s_1t_2-s_2t_1)^2+(\epsilon_{33}\epsilon_{11}-\epsilon_{13}^2)(s_3t_3-s_1t_3)^2+ \\ &\quad (\epsilon_{33}\epsilon_{22}-\epsilon_{23}^2)(s_3t_2-s_2t_3)^2+2(\epsilon_{11}\epsilon_{23}-\epsilon_{13}\epsilon_{12}) \\ &\quad (t_1^2s_2s_3+s_1^2t_2t_3-t_1t_2s_1s_3-t_1t_3s_1s_2)+ \\ &\quad 2(\epsilon_{23}E_{12}-\epsilon_{12}\epsilon_{13})(t_1^2s_2s_3+s_1^2t_2t_3-t_1t_2s_1s_3-t_1t_3s_1s_2). \end{aligned} \tag{4.45}$$

From equations (4.44) and (4.45), we find that K_1 is $O(h^{m-1})$ and K_2 is of $O(h^{2(m-1)})$. Therefore,

$$\boxed{|K-K^c| \approx \mathcal{O}(h^{m-1}).} \tag{4.46}$$

(v) *Principal curvature:*

For the principal curvature estimation, we need the error for the terms

$\frac{(a^c_{11}-a^c_{22})}{2}$ and a^c_{12}. Therefore, we define B by

$$\begin{aligned} B &:= \frac{1}{2}\left[(t_1^2-s_1^2)\mathcal{H}_{11}+(t_2^2-s_2^2)\mathcal{H}_{22}+(t_3^2-s_3^2)\mathcal{H}_{33}+\right. \\ &\left.(t_1t_2-s_1s_2)\mathcal{H}_{12}+(t_1t_3-s_1s_3)\mathcal{H}_{13}+(t_2t_3-s_2s_3)\mathcal{H}_{23}\right], \end{aligned} \tag{4.47}$$

and the error term B_e by

$$\begin{aligned} B_e &:= \frac{1}{2}\left[(t_1^2-s_1^2)\epsilon_{11}+(t_2^2-s_2^2)\epsilon_{22}+(t_3^2-s_3^2)\epsilon_{33}+(t_1t_2-s_1s_2)\epsilon_{12}+\right. \\ &\left.(t_1t_3-s_1s_3)\epsilon_{13}+(t_2t_3-s_2s_3)\epsilon_{23}\right]. \end{aligned} \tag{4.48}$$

Similarly,

$$\begin{aligned} C &:= [t_1s_1\mathcal{H}_{11}+t_2s_2\mathcal{H}_{22}+t_3s_3\mathcal{H}_{33}+(t_1s_2+t_2s_1)\mathcal{H}_{12}+ \\ &(t_1s_3+t_3s_1)\mathcal{H}_{13}+(t_2s_3+t_3s_2)\mathcal{H}_{23}], \end{aligned} \tag{4.49}$$

and the error term,

$$C_e := [t_1s_1\epsilon_{11}+t_2s_2\epsilon_{22}+t_3s_3\epsilon_{33}+(t_1s_2+t_2s_1)\epsilon_{12}+ \tag{4.50}$$

$$(t_1s_3+t_3s_1)\epsilon_{13}+(t_2s_3+t_3s_2)\epsilon_{23}]. \tag{4.51}$$

The approximate principal curvature is

$$\lambda^c_{1,2} = A-A_e \pm \sqrt{(B-B_e)^2+(C-C_e)^2}. \tag{4.52}$$

For λ^c_1, the error is given by

$$\begin{aligned} |\lambda^c_1-\lambda_1| &= |A_e+\sqrt{[B^2+C^2-2(BB_e-CC_e)+(B_e^2+C_e^2)]}- \\ &\sqrt{B^2+C^2}|. \end{aligned} \tag{4.53}$$

We see that A_e is of $O(h^{m-1})$ and the term

$$\sqrt{[B^2+C^2-2(BB_e-CC_e)+(B_e^2+C_e^2)]},$$

can be written as

$$\begin{aligned} &\sqrt{B^2+C^2}\left[1-\frac{2(BB_e-CC_e)+(B_e^2+C_e^2)}{B^2+C^2}\right]^{1/2} \\ &\approx \sqrt{B^2+C^2}\left[1-\frac{2(BB_e-CC_e)+(B_e^2+C_e^2)}{2(B^2+C^2)}\right] \end{aligned}$$

as $(BB_e-CC_e+B_e^2+C_e^2) < (B^2+C^2)$, which is of $O(h^{(m-1)})$. Therefore,

$$\boxed{|\lambda_1-\lambda^c_1| \approx \mathcal{O}(h^{(m-1)}).} \tag{4.54}$$

Likewise, we observe that

$$\boxed{|\lambda_2-\lambda^c_2| \approx \mathcal{O}(h^{(m-1)}).} \tag{4.55}$$

(vi) *Principal direction:*
The principal directions are given by equation (4.20) as

$$\begin{aligned} \vec{pd}_1 &= (pd_{11}, pd_{12}, pd_{13}) \\ &= (t_1 a_{12} + s_1 \lambda_1, t_2 a_{12} + s_2 \lambda_1, t_3 a_{12} + s_3 \lambda_1), \qquad (4.56a) \\ \vec{pd}_2 &= (pd_{21}, pd_{22}, pd_{23}) \\ &= (t_1 \lambda_1^c - s_1 a_{12}, t_2 \lambda_1^c - s_2 a_{12}, t_3 \lambda_1^c - s_3 a_{12}). \qquad (4.56b) \end{aligned}$$

Therefore, the error in the approximate $\vec{pd}_1^c$ is

$$\begin{aligned} |\vec{pd}_1 - \vec{pd}_1^c| &= (|(a_{12} - a_{12}^c)t_1 + (\lambda_1 - \lambda_1^c)s_1|, \\ &\quad |(a_{12} - a_{12}^c)t_2 + (\lambda_1 - \lambda_1^c)s_2|, \\ &\quad |(a_{12} - a_{12}^c)t_3 + (\lambda_1 - \lambda_1^c)s_3|)\,. \qquad (4.57) \end{aligned}$$

From equation (4.51), we know

$$(a_{12} - a_{12}^c) = C_e.$$

Therefore, from equation (4.51)

$$\begin{aligned} (a_{12} - a_{12}^c)t_1 + (\lambda_1 - \lambda_1^c)s_1 &= \big[t_1^2 s_1 \epsilon_{11} + t_2 s_2 t_1 \epsilon_{22} + t_3 s_3 t_1 \epsilon_{33} + \\ &\quad (t_1 s_2 + t_2 s_1) t_1 \epsilon_{12} + (t_1 s_3 + t_3 s_1) t_1 \epsilon_{13} + \\ &\quad (t_2 s_3 + t_3 s_2) t_1 \epsilon_{23}\big] + (\lambda_1 - \lambda_1^c) s_1. \qquad (4.58) \end{aligned}$$

Again, by looking at $\epsilon_{11}, \ldots, \epsilon_{22}$ and $(\lambda_1 - \lambda_1^c)$, we obtain the following result

$$\boxed{|\vec{pd}_1 - \vec{pd}_1^c| \approx \mathcal{O}(h^{(m-1)}).} \qquad (4.59)$$

Similarly, we show that $|\vec{pd}_2 - \vec{pd}_2^c|$ is also $\mathcal{O}(h^{(m-1)})$.

□

4.2. Analytic test case

To study the accuracy and convergence of our method, we compare our estimation of surface parameters with known analytical results. For this purpose, as in Chapter 2 we take an arbitrarily oriented torus as a test example. We recall the following notation which is used for describing surface parameters:

P_i^c, P_i	-	estimated and analytic interface position
$\vec{n}^c, \vec{n}$	-	estimated and analytic normal vector
H^c and H	-	estimated and analytic mean curvature
K^c and K	-	estimated and analytic Gaussian curvature
Φ^c and Φ	-	estimated and exact signed distance function
$\vec{pd}_1^c, \vec{pd}_2^c$,	-	estimated principal curvature direction
$\vec{pd}_1, \vec{pd}_2$	-	analytical principal curvature direction

The signed distance function Φ is constructed in two different ways:

(1) from the implicit representation of the surface which we call *exact* signed distance and

(2) from the triangulated surface.

We study the convergence of surface parameters in both cases separately. In each of them, we show the results of least square method for 2nd, 3rd, and 4th order polynomials (denoted as LS) and the interpolated sixth order finite difference scheme (see Appendix A). Without loss of generality, this finite difference method is referred to as *FD interpolation* method hereafter. All plots are in log-log scale and the results for different orders of polynomials are shown with different colors. For each color, the discrete points are the results from our simulation and the continuous line is the best linear fit in the log-log scale. The slope of this line is taken as the approximate order of convergence in our study.

4.2.1. Exact signed distance from a torus. The exact signed distance from a torus is found from equation (2.15). The exact surface parameters can be obtained from the parameterization of the torus, given by

$$\begin{aligned} \mathbf{X}(\theta,\phi) &= ((R+r\cos\theta)\cos\phi, (R+r\cos\theta)\sin\phi, r\sin\phi), \\ &\quad \theta,\phi \in (0,2\pi], \end{aligned}$$

with the analytical normal

$$\vec{n}(\theta,\phi) = (\cos\theta\cos\phi, \cos\theta\sin\phi, \sin\phi),$$

(4.60)

and the mean and Gaussian curvature given by(see [**19**])

$$H = -\frac{R+2r\cos\theta}{2r(R+r\cos\theta)} \text{ and } K = \frac{\cos\theta}{R+r\cos\theta}.$$

As before the parameter r is the radius of the tube, and R is the distance of the center hole to the center of the tube in a local coordinate system $(\vec{t_1}, \vec{t_2}, \vec{n})$, as shown in Figure 2.14. We have,

$$\vec{t_1} = (0.5, 0.0, 0.866025),\ \vec{t_2} = (0.0, 1.0, 0.0)$$

and $\vec{n} = (-0.866025, 0, 0.500000)$, with $R = 0.6$ and $r = 0.2$.

The uniform meshwidth is given by $4/G_s$, where G_s is the grid resolution. For our test, we use $20^3, 40^3, 50^3, 100^3, 200^3, 400^3,$ and 800^3 grids. Figure 4.2 shows the plot of the infinity norm of the error of the surface parameters, with respect to different grid resolutions, taken uniformly in all three directions.

For the interface position we can see that the interpolation gives better estimations than the 2nd, 3rd, or 4th order least squares approach, but the error is also found to be more oscillatory. As expected from Theorem 3, for the position, the 2nd, 3rd and 4th order polynomials give 3rd, 4th and 5th order of convergence, respectively. It is found that 2nd, 3rd, and 4th order polynomials produce 2nd, 3rd, and 4th order of convergence for normals as expected. For the mean and Gaussian curvatures, it is found that 2nd, 3rd, and 4th order polynomials produce approximately 1st, 2nd, and 3rd order of convergence. For the principal directions, we also found that the least squares method gives better estimations than the interpolation. Here, we have not shown lower order finite difference interpolation as it is found that the rate of convergence is much lower than sixth order.

Therefore it is clear that for the estimation of normal, and curvatures, and their directions on the surface, the least squares method is accurate and has better order of convergence than higher order interpolation methods used in the literature. Further, the advantage of the least squares method is that it needs only very thin narrow band around interface. For instance, the 4th order least squares method needs only 57 points and it is enough to have narrow band width of just four times the mesh width. On the contrary, the finite difference technique requires nine times the mesh width.

Table 4.1 shows the order of convergence for surface parameters for various orders.

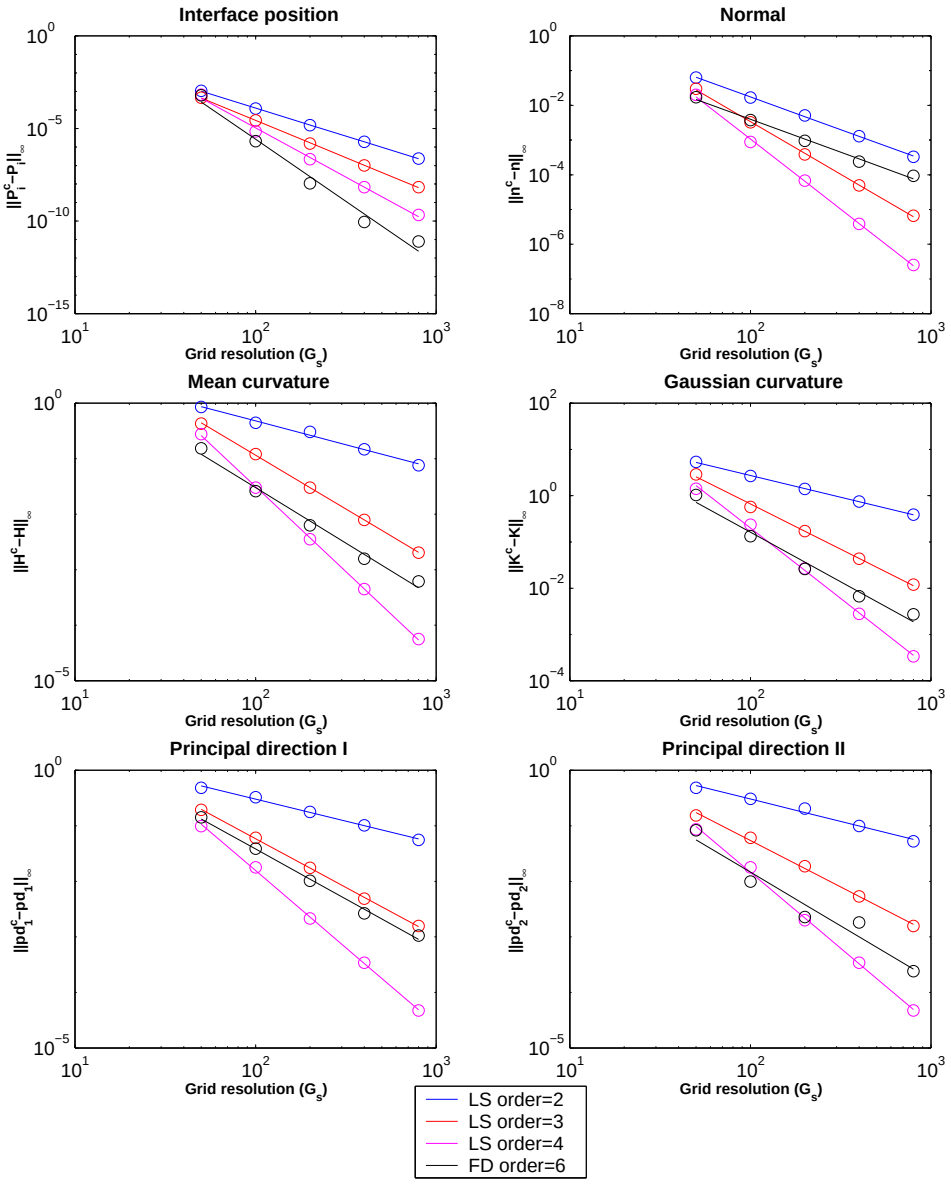


FIGURE 4.2. *Discretization error of different surface parameters with respect to grid resolutions.*

4.2.2. Signed distance function from a surface triangulation: In Chapter 2 we showed that the estimation of signed distance functions by the direct method depends chiefly on the surface resolution, i.e. triangulation. Here, we investigate two types of convergence:

(1) based on different surface resolutions for a fixed mesh width, and

(2) based on different mesh width for a fixed triangulation.

Method	Polynomial Order	Position	Normal	H	K	pd_1	pd_2
Least squares	2	3.06	1.89	0.86	0.95	0.79	0.80
Least squares	3	4.02	3.03	1.94	1.95	1.75	1.67
Least squares	4	5.32	4.04	3.06	3.04	2.78	2.74
FD Interpolation	6	6.70	1.90	2.00	2.15	1.80	1.93

TABLE 4.1. Order of convergence for surface parameters by different methods for torus where Φ is an exact signed distance. H - Mean curvature, K- Gaussian curvature, pd_1 and pd_2 are the principal directions.

Figure 4.3 show the error in the estimation of surface parameters for various surface resolutions (from 80000 to 5.12 million triangles) with fixed meshwidth of 0.01 for the same torus ($r = 0.2$ and $R = 0.6$). For the interface position, we find that the FD interpolation has the same order of convergence as the 4th order least squares method. Also, we find the fourth order least squares approach is accurate than FD method.

Thus, from the above discussion we can conclude that a good surface resolution is neceessary for estimating surface parameters accurately by higher order least squares approach. For the second order least squares or FD interpolation, increasing the surface resolution will not help in estimating surface parameters accurately.

Figures 4.4 shows the error in the estimation of surface parameters for different mesh width with fine surface resolution (around 2 million triangles). Table 4.2 shows the order of convergence for surface parameters of the torus. We observe that the order of estimation is lower than the corresponding parameters estimated with the exact signed distance case (see Table 4.1). This indicates that the results are sensitive to the signed distance function, which in turn depends on the triangulations.

4.2.3. Coarse surface and mesh resolutions. We observe, from Figure 4.3, that there is a fluctuation of the error for the first few surface resolutions. To study the effects of coarse triangulation with the fixed grid, Figure 4.5 shows the plot of surface parameters for different

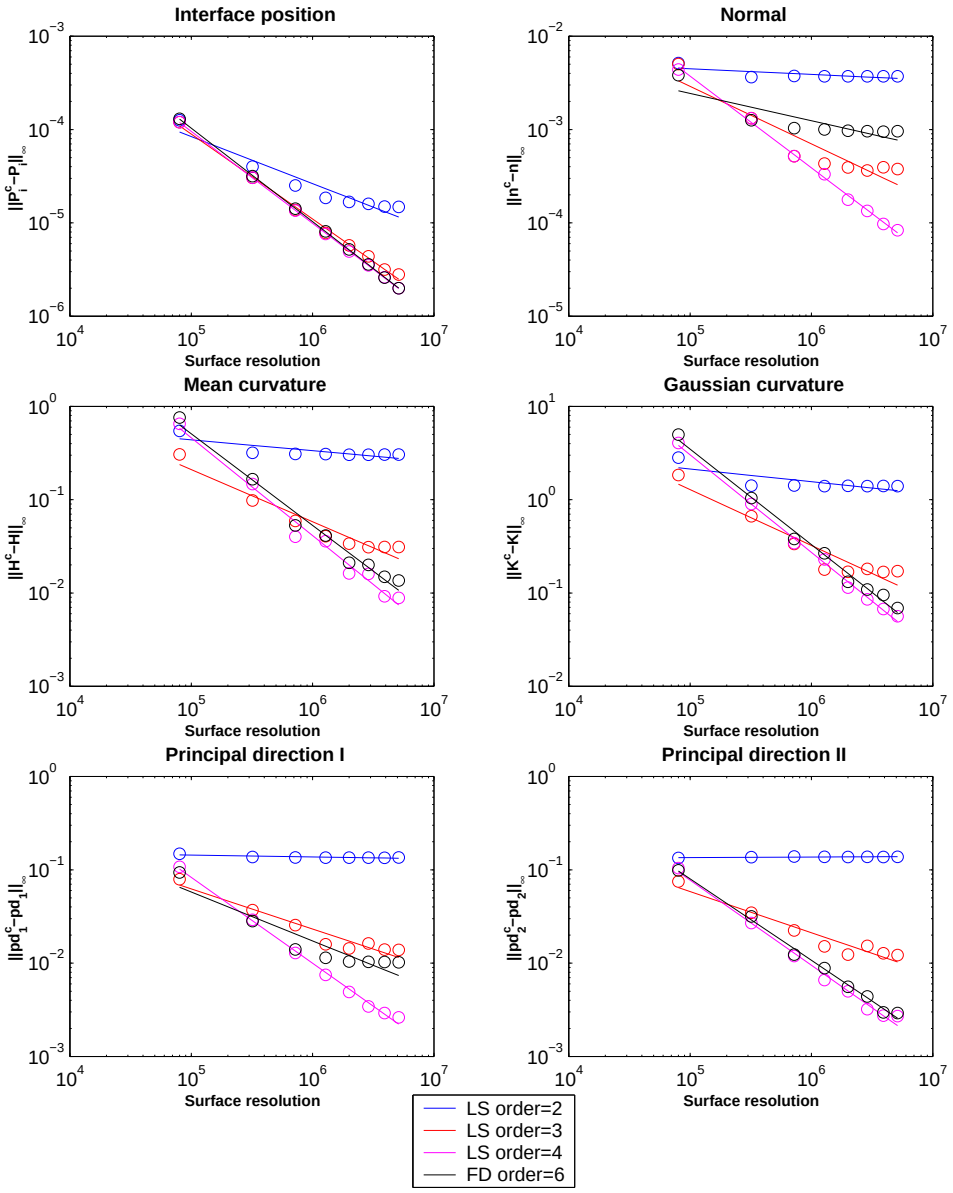


FIGURE 4.3. *Discretization error of the surface parameters with different surface resolution for* $h = 0.01$.

grid resolutions with a fixed number of surface triangles (80000 faces). In all these plots, we find that the results are highly oscillatory and therfore, we cannot ascertain the order of convergence. Similarly, to analyze the effects of coarse mesh width for various surface resolutions, we show the error plot of surface parameters in Figure 4.6 for a fixed mesh width $h = 0.08$ for different polynomial orders. In all these plots we find the results are accurate with the higher order least squares, but

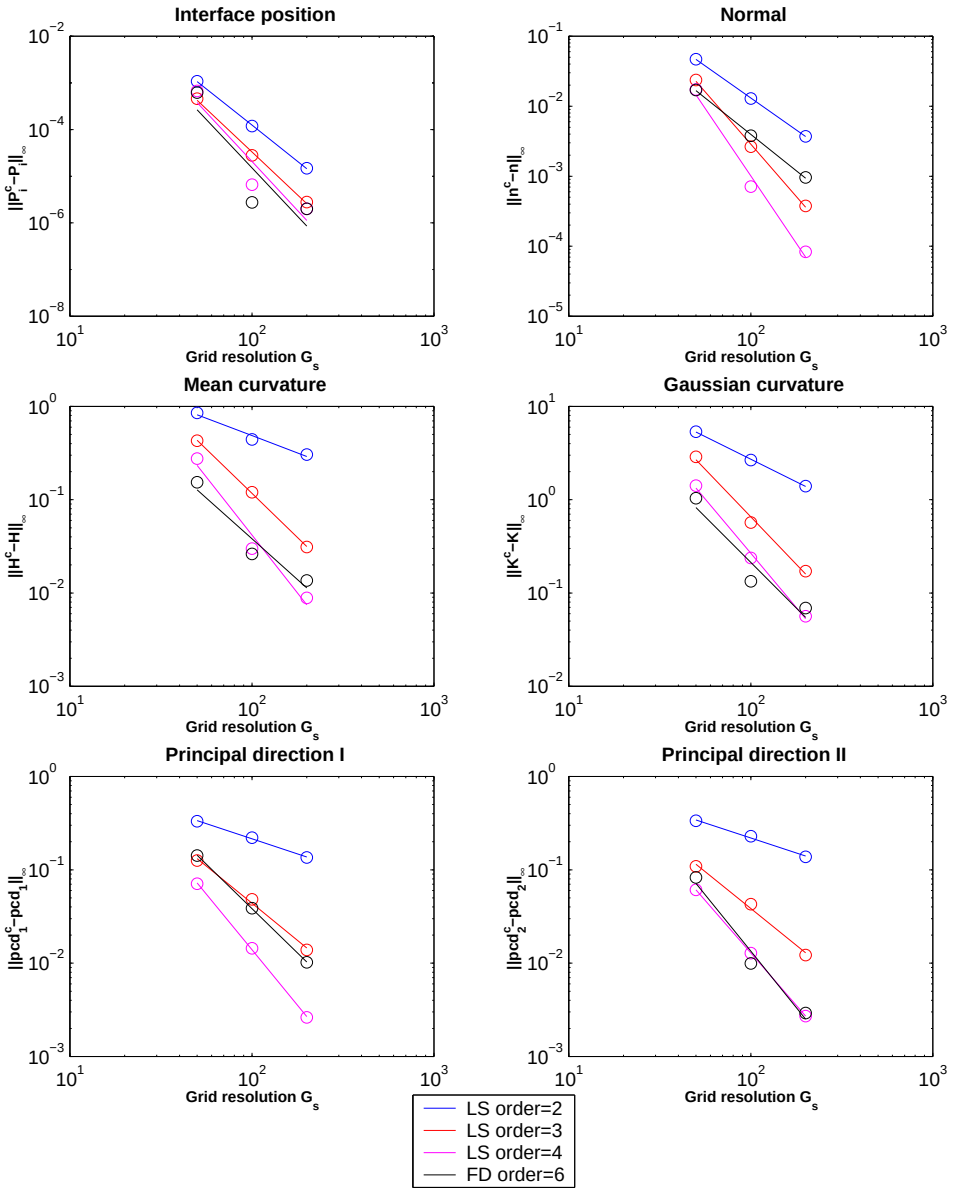


FIGURE 4.4. *Discretization error of the surface parameters for different mesh resolutions with fixed surface triangulation.*

it fails to converge even with very large number of triangles (around 5 million). Therefore, optimum mesh resolution is also a necessary condition to reach the correct rate of convergence.

Method	Polynomial order	Position	Normal	H	K	pd_1	pd_2
Least squares	2	3.09	1.83	0.74	0.97	0.64	0.64
Least squares	3	3.67	2.99	1.89	2.03	1.59	1.58
Least squares	4	4.20	3.85	2.48	2.32	2.38	2.25
FD Interpolation	6	4.13	2.07	1.74	1.95	1.90	2.41

TABLE 4.2. Order of convergence for surface parameters by different methods for torus, where Φ is calculated from triangulation. H - Mean curvature, K- Gaussian curvature, pd_1 and pd_2 are the principal directions.

4.2.4. Heuristic results. Therefore, we can reach some important *heuristic* conclusions, while dealing with the higher order approximation of surface parameters, when the surface is discretized by a triangulation:

REMARK 10. *Conditions for convergence:*

(i) *Decreasing the mesh width is not useful when the number of triangles are not sufficient to resolve the surface accurately.*

(ii) *Increasing the surface triangulation does not improve the results if the order of least squares approximation is small.*

(iii) *Increasing the surface triangulation, while using the coarse mesh, will also not help to bring the convergence of the surface parameters to the desired order. The mesh width should be sufficiently fine to get the correct order of convergence.*

Therefore, the choice of the right order of least squares approximation, a moderately fine mesh width, and a sufficiently fine surface triangulation are crucial for obtaining an accurate estimate of surface parameters.

4.2.5. Comparison of the Direct least squares method with FMM. In Chapter 2 we compared the direct method with FMM and showed that for the estimation of Φ, the former is more accurate than the latter. Here, we investigate a similar type of study for the estimation of surface parameters. To be precise, we investigate four different approaches:

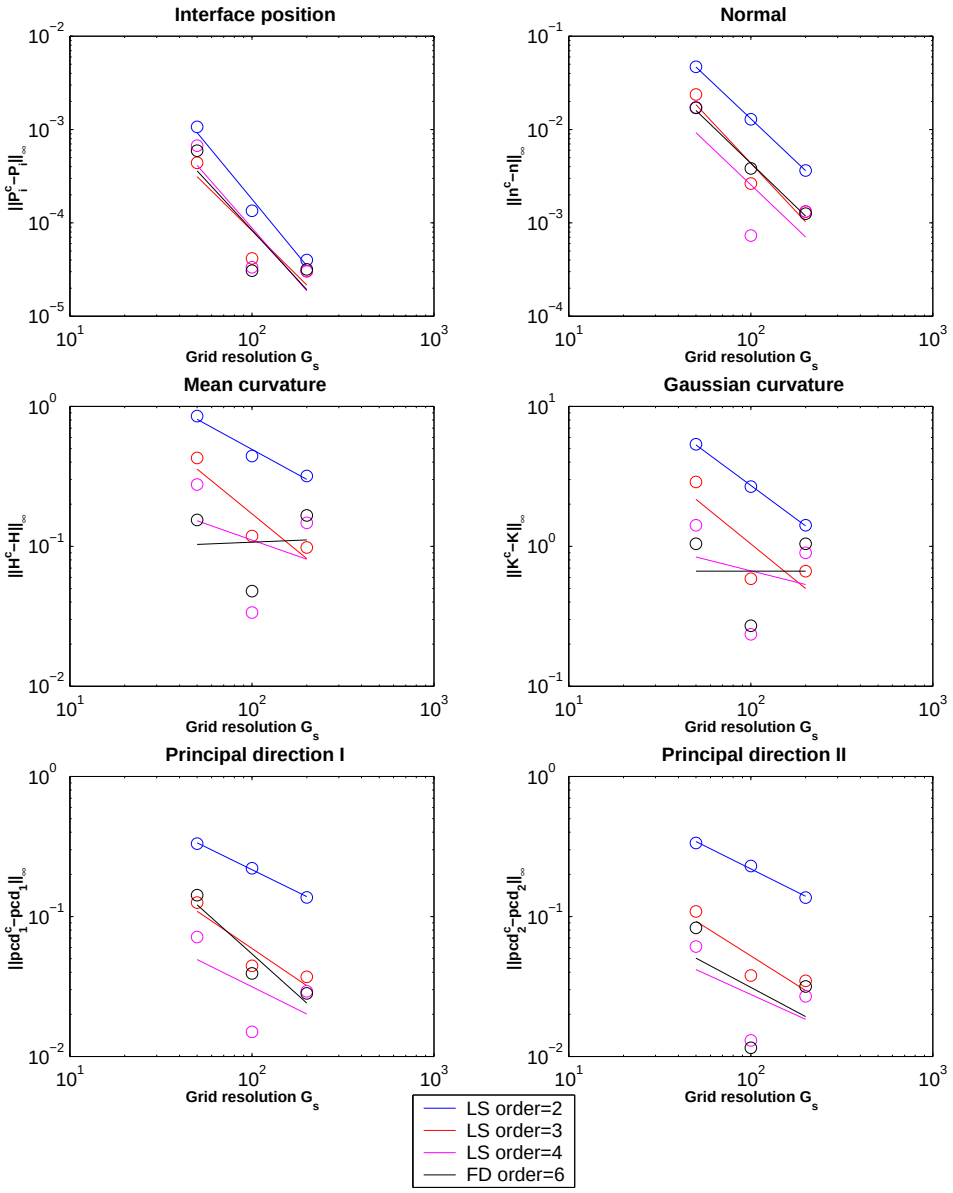


FIGURE 4.5. *Discretization error of surface parameters with different mesh width for coarse triangulations.*

(1) fourth order least squares with Φ computed by the direct method,

(2) fourth order least squares with Φ computed by first order FMM (denoted by FMM1),

(3) fourth order least squares with Φ computed by second order FMM (denoted by FMM2) and

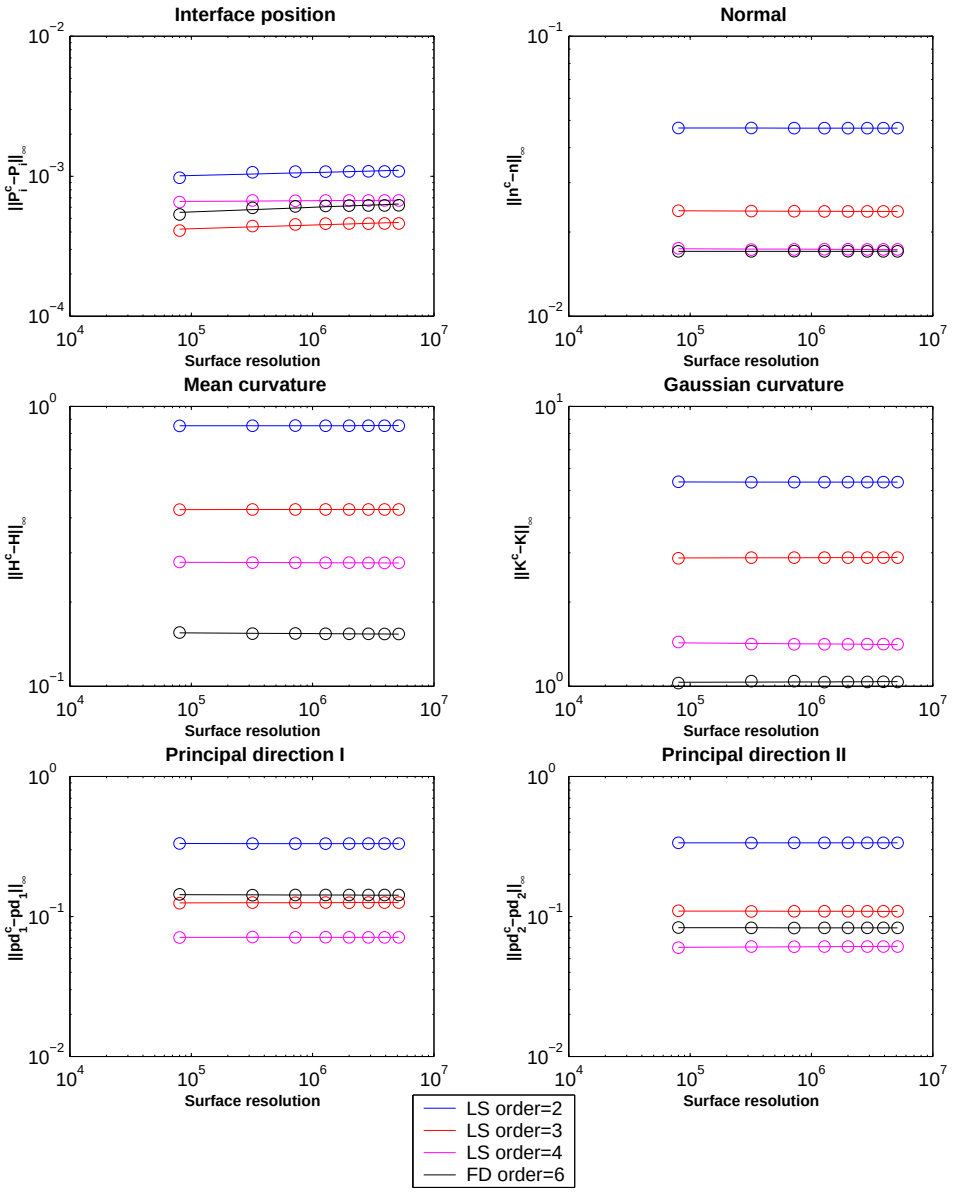


FIGURE 4.6. *Discretization error of the surface parameters for various orders of polynomial with coarse mesh width* $h = 0.08$.

(4) sixth order finite difference interpolation with Φ computed by the direct method.

We recall that in FMM, the initial Φ is computed within a distance of h from the interface by the direct method. Figure 4.7 shows the plot of surface parameters with different methods for various grid resolutions. For conciseness, we show here only the accuracy of interface

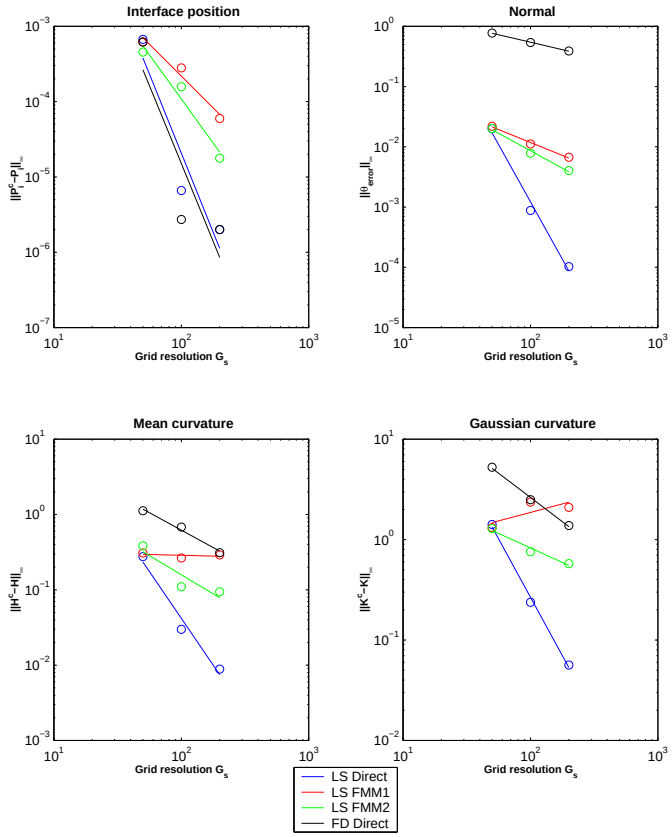


FIGURE 4.7. *Accuracy of surface parameters estimated by 4th order least squares method with Φ estimated by direct, FMM1 and FMM2 approaches and sixth order FD interpolation*

position, normal, mean and Gaussian curvatures for different grid resolutions. It is clearly seen that estimating surface parameters by least squares with Φ computed by the direct method is more accurate and has better order of convergence than FMM1 or FMM2. For estimating interface position and normal the difference in the order of convergence between the FMM1 and FMM2 is not significant. On the other hand FMM2 has a better convergence rate than FMM1 in the estimation of mean and Gaussian curvature, but still cannot match the accuracy and convergence order of the direct method. We find also that FD interpolation has a lower order of convergence than FMM2 except in the estimation of the interface position. It is clear from this investigation that one needs higher order FMM to get an accurate estimation of surface parameters. On the other hand, as we mentioned before, this may

not be efficient, as for a higher order FMM we need initial Φ values for a larger width (at a distance $\geq 2h$) from the interface, which decreases the speed of computation. Therefore, by looking into these aspects, we find that the least squares approach with Φ computed by the direct approach is more accurate and efficient than the FMM aproach for the computation of the surface parameters on the interface.

CHAPTER 5

Propagation - Application to moving interface problems

We now study the propagation of the interface by an externally generated velocity field $\vec{v}$. The original level set equation proposed by Osher and Sethian [**63**] uses a simple advection equation

$$\Phi_t + \vec{v} \cdot \nabla \Phi = 0, \tag{5.1}$$

to track implicitly the evolution of the interface,. Here it must be remembered that $\vec{n}$ and $\nabla\Phi$ point in the same direction. Therefore, it is not necessary to specify the tangential component $\vec{t}$, of the velocity as $\vec{t} \cdot \nabla\Phi = 0$. For instance, in two dimensions, if F is the normal velocity, G is the tangential velocity, and $\vec{v} = F\vec{n} + G\vec{t}$, then equation (5.1) becomes

$$\Phi_t + F\vec{n} \cdot \nabla \Phi = 0, \tag{5.2}$$

since $\vec{t} \cdot \nabla\Phi = 0$. As

$$\vec{n} \cdot \nabla \Phi = \frac{\nabla \Phi}{|\nabla \Phi|} \cdot \nabla \Phi = \frac{|\nabla \Phi|^2}{|\nabla \Phi|} = |\nabla \Phi|,$$

equation (5.2) becomes

$$\Phi_t + F|\nabla \Phi| = 0. \tag{5.3}$$

As we mentioned in Chapter 1, this also known as level set equation in the literature [**83**]. Here we use equation (5.3) for propagating the interface.

5.1. Estimation of F on grid points

The first step in the study of propagation by level set methods is the estimation of the normal velocity F on the grid point from the velocity field $\vec{v}$. In the Cartesian coordinate system if the velocity field $\vec{v} = (v_x, v_y, v_z)$ is known at *each* grid point, then the normal velocity F can be estimated by computing the normal using $\vec{n} = \frac{\nabla\Phi}{|\nabla\Phi|}$. But in

many cases, especially solvers using MAC grid, v_x, v_y and v_z are stored in a staggered way. Furthermore, in the study of free moving boundary problems, sometimes the discrete velocity field $\vec{v}$ is known only on one side of the interface. Therefore, we need an approach to extrapolate the velocity across the interface. In the literature, there are many approaches used to extrapolate velocity across the interface. To name a few, ghost fluid techniques [**25**], the method of characteristics [**6**] and the constant velocity extension [**3**] are some of the methods used in the literature.

Here, we use the constant velocity extension method to extend normal velocity from the interface to the grid point. The velocity field from the grid is first extrapolated to discrete points on the the interface by a least squares approach as discussed in the previous chapter. In our study, these discrete points are the set of projection points of grids inside the narrow band. This is schematically shown in Figure 5.1(a). We use the same equation (4.9) for finding the velocity on the interface, where f is the velocity component v_x, v_y and v_z of length N_r, the number of sample points within search radius r. If the velocity components are given in a staggered form as shown in Figure 5.1(b), then the equation (4.9) is solved (in a least square sense) for each component separately. As before, the grid matrix $\mathbf{A}$ is of length $N_r \times l$, where l depends on the order of approximation. If $(\mathbf{A^T A})$ is singular, then the search radius is increased to accommodate more points. If it is not possible to increase the number of sample points then the order of extrapolation is reduced. The normal velocity F at a grid point $\vec{P}$ is found from the velocity field $\vec{v}_s$ at the interface point $\vec{P}_p$ by

$$F = \vec{v}_s \cdot \frac{(\vec{P} - \vec{P}_p)}{\|(\vec{P} - \vec{P}_p)\|_2} \tag{5.4}$$

5.2. Applications

We investigate three different applications:

(1) *Mean curvature flow:* In this study, the interface is propagated under its mean curvature. The curvature is estimated by higher order least squares from the discrete level set function as discussed in the previous chapter.

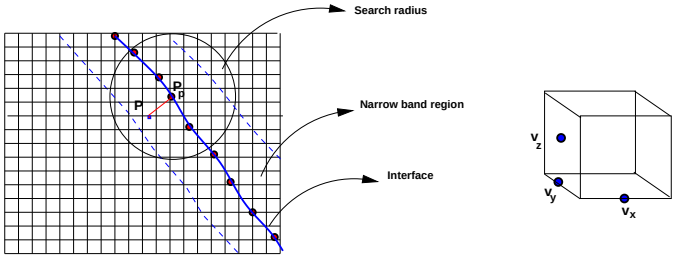


FIGURE 5.1. *(a) Least square extrapolation of velocity field and (b) staggered representation of velocity component.*

(2) *Bubble dynamics:* The level set algorithm is coupled with Lattice-Boltzmann (LB) code to investigate the movement of interface. Here, we verify the Young-Laplace law for the pressure of a spherical bubble under surface tension forces. Further, we give an example of level set LB coupling viz., the coalescence of bubbles due to surface tension effects.

(3) *VOF-coupling in the mold flow:* The level set algorithm is coupled with VOF representation in the study of an injection mold-flow using SIGMASOFT software.

5.2.1. Mean curvature flow. The evolution of a hypersurface moving according to its mean curvature has been studied in various settings. In the parametric setting, for instance Gage and Hamilton[**28**], Grayson [**34**], Huisken [**42**] investigated self similar flows due to shrinking of a closed curve. In the level set methodology, we look for $\Gamma(t)$ as the zero level set of function $\Phi : \mathbb{R}^{n+1} \times [0, \infty) \to \mathbb{R}$, i.e.

$$\Gamma(t) = \{x \in \mathbb{R}^{n+1} | \Phi(x, t) = 0\}.$$

Thereby, we use equation (5.3) with $F = -H$, where H is the mean curvature. This equation is of interest for the study of geometric curve and surface evolution in image processing [**75**] and in the modeling of dendrite growth [**83**]. Numerically, equation (5.3) is solved with a small regularizing parameter ϵ such that

$$\Phi_t - H\epsilon|\nabla\Phi| = 0. \tag{5.5}$$

The proof for the existence of a solution based on viscosity solution for equation (5.5) is well known [**23**], [**24**]. In the discretized form if

$\Phi(x_i, y_j, z_k)$ at time $t = n$ is represented by $\Phi^n_{i,j,k}$, then at time $t = n+1$ it is,

$$\Phi^{n+1}_{i,j,k} = \Phi^n_{i,j,k} - \Delta t \epsilon H^n_{i,j,k} |\nabla \Phi^n_{i,j,k}|. \tag{5.6}$$

The time step Δt satisfies the CFL condition such that $|F|\epsilon \frac{\Delta t}{h^2} < 1$. In the finite difference setting, the mean curvature H (given by equation (4.1)) is discretized by higher order central differences. The propagation equation (5.6) is then updated for each time step with simple Euler stepping and $|\nabla \Phi|$ is discretized using a first order upwind method [**83**].

If Φ is signed distance function then,

$$\nabla \cdot \frac{\nabla \Phi}{|\nabla \Phi|} = \Delta \Phi. \tag{5.7}$$

Therefore, equation (5.5) becomes a heat equation. For this type of equation, recently, a fourth-order finite difference scheme has been applied by Gibou and Fedkiw [**32**] for the study of Stefan problems. The advantage of the heat equation is that one can go for the implicit time discretization, thereby a larger time steps can be used in the computation. Unfortunately, one cannot generally substitute equation (5.7) with equation (5.5) for the study of mean curvature flow. The reason is that Φ^n may be a signed distance function, but Φ^{n+1} will generally not be a signed distance function. In other words, $\Delta \Phi^n$ may be equal to $H^n |\nabla \Phi^n|$, but $\Delta \Phi^{n+1}$ will in general not be equal to $H^{n+1} |\nabla \Phi^{n+1}|$. To solve for large time steps, recently a semi implicit method is proposed by Smereka [**85**]. In this approach Smereka used a higher order finite difference scheme for the estimation of curvature at each grid point.

Here, we follow a different approach. As we found in the last chapter, the curvature estimated by the FD method is not accurate and also has a poor rate of convergence, we estimate it *on* the surface using a higher order least squares method. Therefore, the curvature term is *not* discretized explicitly on the grid points. In our approach, as we have the normal on the interface, the velocity F at the grid points can be estimated easily from the projection point. By doing this we not only estimate the curvature accurately, but also avoid back and forth extrapolation of normal velocity on the grid.

5.2.1.1. *Validation with analytical results.* In the context of level set method, the numerical validation for the study mean curvature flow of closed contour is analyzed in detail in [**64**], [**70**], [**27**]. The important aspect investigated in all these studies is the mass loss during surface evolution. This is because the level set method has a tendency to loose mass in the under resolved regions. Further, during reinitialization of Φ, the interface may move from the zero level set making it vulnerable for mass loss/gain locally. To avoid this, several corrective measures are used during reinitialization [**89**], [**70**], [**22**]. In all these models, the curvature term is discretized by central differences. Our intention here is to investigate further the influence of curvature estimation on the mass loss. Therefore, for a fixed reinitialization step, we perform three sets of experiments.

(1) The mean curvature H, given by equation (4.1), is discretized by higher order central differences at each grid point (i, j, k) inside the narrow band. We call this a *classical approach*.
(2) The discrete mean curvature found from equation (4.1) is extrapolated to the interface by cubic splines. Then the grid points inside the narrow band inherit the normal velocity from the interface by the method of constant velocity extension. This is called the *FD interpolation approach*.
(3) The curvature H is not discretized at each grid point. We use the curvature estimation on the interface by least squares method described in Chapter 4. The grid points inside the narrow band inherits the normal velocity by the method of constant velocity extension. This we call the *least squares approach*.

We use then equation (5.5) for propagation. Here, we use explicit Euler time stepping and $|\nabla\Phi|_{i,j,k}$, is discretized by a first order upwinding method [**83**]. For the mean curvature flow, it is found that the first order upwinding is sufficient [**62**].

A first benchmark experiment is the shrinking of a sphere centered at $(0, 0, 0)$ of initial radius $r_0 = 1$, with mesh resolution $h = 0.0002$, smoothing parameter $\epsilon = 1.0e{-}2$ and $\Delta t = 0.006$. Explicit reinitialization is performed every 100 time steps by the marching cube algorithm

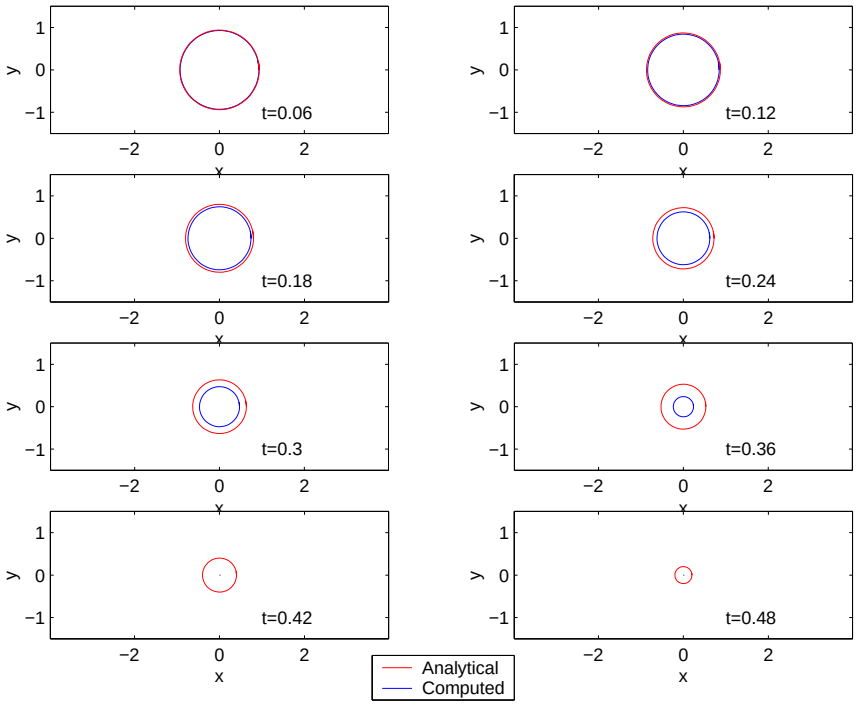


FIGURE 5.2. *Shrinking of a sphere by mean curvature flow using the classical approach.*

(see Chapter 3). Thereby Φ is estimated again from the surface triangulation (see Chapter 2). Analytically, it can be shown that the radius at time t is given by $r_t = \sqrt{(r_0 - 2\epsilon n_t \Delta t)}$, where n_t is the number of time steps.

Figure 5.2 shows the cross section of the shrinking of sphere by classical approach. It can be seen very clearly that in principle the shrinking of the surface follows self-similarity according to the theory, but the estimated mean curvature flow is faster than the analytical solution. At each step it looses mass drastically and shrinks completely well before the expected time. Figure 5.3 displays the shrinking of the sphere using the FD interpolation. The shrinking and the mass loss in this method is found to be very similar to the classical approach. Figure 5.4 shows the shrinking of the sphere with the second order least squares method with constant velocity extension. We can see here that the shrinking is better compared with the analytical solution than in the classical or FD approaches at least for the first few time steps. Table 5.1 shows the percentage of accumulative mass loss at

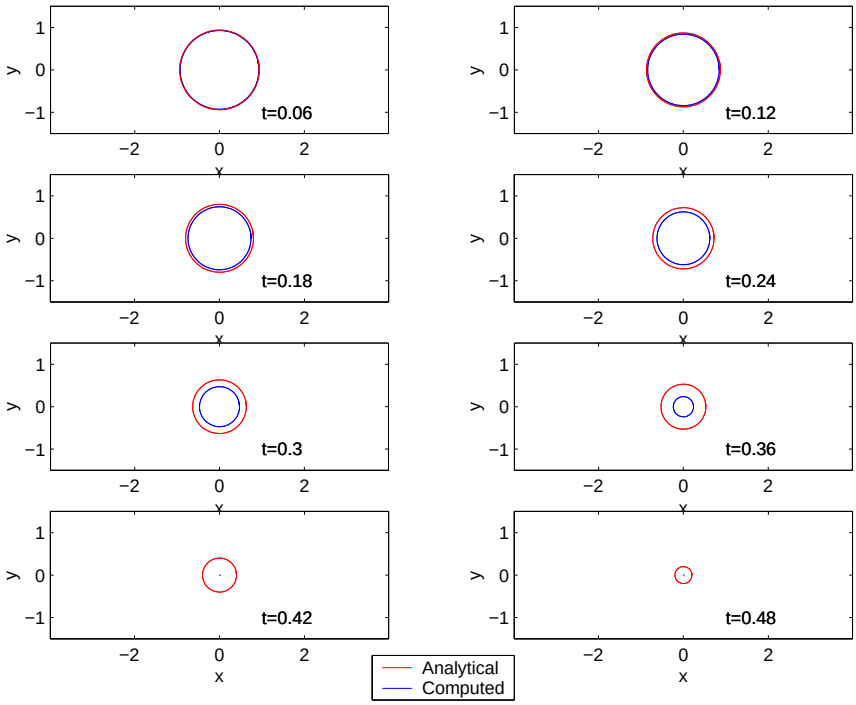


FIGURE 5.3. *Shrinking of a sphere by mean curvature flow using the FD interpolation approach.*

Time	Accumulative mass loss (in %)		
	Classical	FD interpolation	Least squares
0.06	2.87	0.24	2.6
0.12	9.442	3.90	9.3
0.18	20.13	8.56	20.67
0.24	35.74	37.24	13.16
0.30	58.54	61.53	25.78
0.36	90.87	94.113	47.37

TABLE 5.1. Accumulative mass loss for classical, FD interpolation and least squares methods using the same reinitialization step.

various time steps for the classical, FD and least squares approaches. Therefore, the accuracy of the curvature estimation is also a important in the study of mean curvature flow.

It is observed that results are not affected by using WENO schemes for $|\nabla\Phi|$ or Runge-Kutta (RK) methods for temporal discretization in

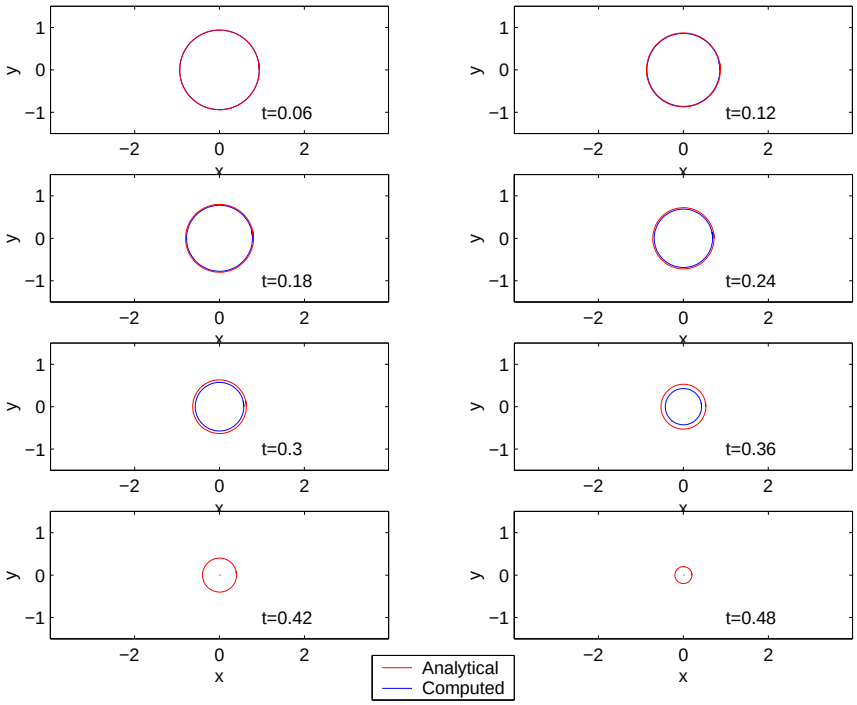


FIGURE 5.4. *Shrinking of a sphere by mean curvature flow using the higher order least squares approach.*

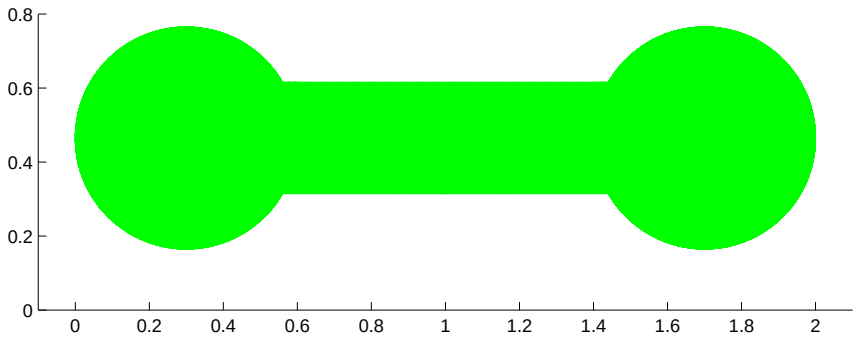


FIGURE 5.5. *Cross-section of the dumbbell geometry.*

equation (5.6). One way to reduce the mass loss, is to not to reinitialize very often and to make the narrow band broad. On the other hand, this will increase the computation time drastically as discussed in Chapter 1.

To investigate the pinching process due to mean curvature flow, we have taken a three dimensional dumbbell geometry. The cross section of this shape is shown in Fig 5.5. We have taken the dimension of the dumbell same as that of Sethian [**80**] to compare the result of our

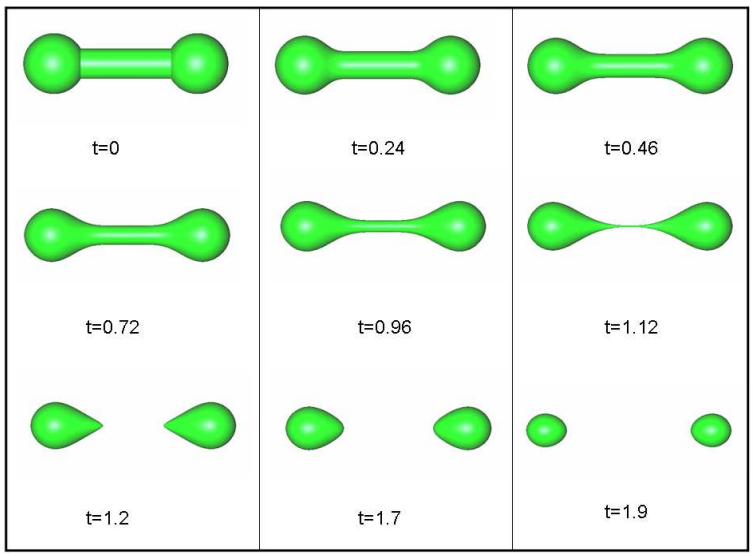


FIGURE 5.6. *Shrinking of a dumbbell by mean curvature flow using the classical approach.*

approach, with the classical method. Figures 5.6 and 5.7 show the shrinking process at various times for the classical and least squares approach, respectively. Here, also, we find that volume of shrinking by the classical method is different from the higher order method at various time steps. The main interest lies at the time during the pinch-off, where classical approach predicts rapid and sudden change, as shown in Figure 5.6 at t=1.12. On the other hand, the least squares approach (Figure 5.7(at t= 1.2)), pinches off slowly by stretching the surface and then separates into two closed surfaces.

A similar observation we found also for the mean curvature flow of the double dumbbell shape as shown in Figure 5.8. The above examples suggest that curvature estimation also plays a major role during the pinch-off process.

5.2.2. Bubble dynamics.

5.2.2.1. *Young-Laplace Law.* To study the bubble dynamics, the Lattice Boltzmann code is coupled with the level set approach for the treatment of the interface. The first test we performed here is to implement the surface tension through the curvature in the Young-Laplace experiment. The Young-Laplace law states that the pressure p inside

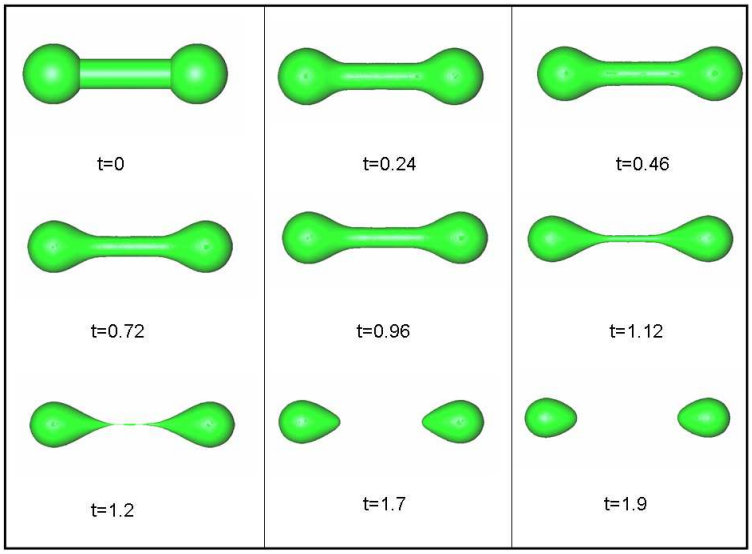


FIGURE 5.7. *Shrinking of a dumbbell by mean curvature flow using the higher order least squares approach.*

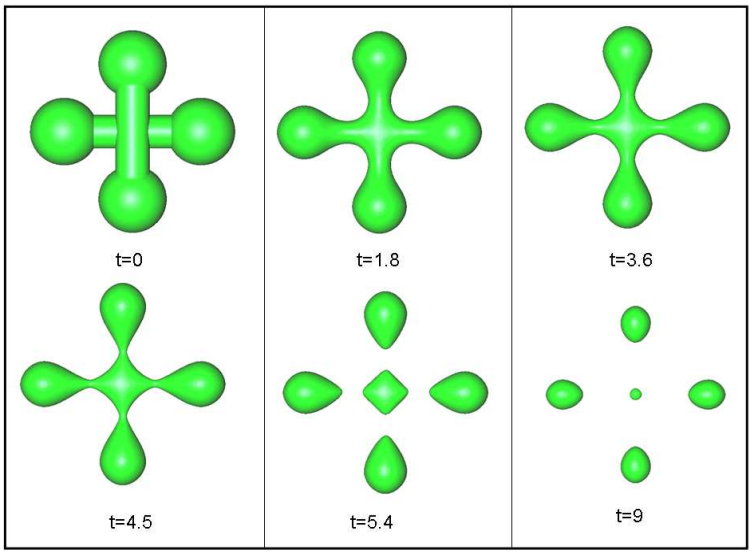


FIGURE 5.8. *Shrinking of a double-dumbell by mean curvature flow using higher order least squares*

a spherical bubble of radius r surrounded by a second fluid is proportional to the surface tension coefficient σ. Thus, according to the Young-Laplace law,

$$p = \frac{2\sigma}{r} = 2\sigma H. \tag{5.8}$$

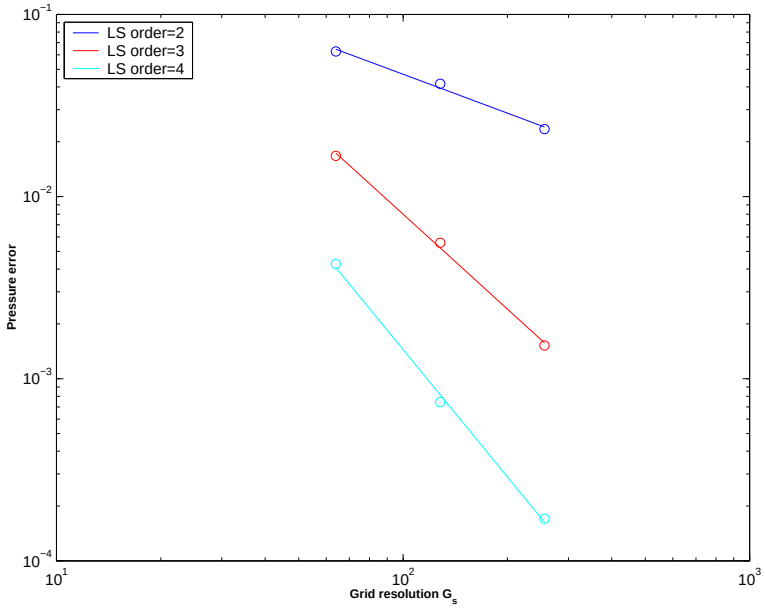


FIGURE 5.9. *Grid convergence pressure for different order of least square estimation of curvature*

A bubble of radius $r = 0.25$ is kept at the center of an unit cube $[0, 1]^3$. For the simulations we used an equidistant grid of N_g nodes in each coordinate direction and periodic boundary conditions at the sides of the cube. For the fluids, the viscosities were chosen equal $\nu_i = 1/6$ and the densities were $\rho_1 = 1$ and $\rho_2 = 10$. We started with zero pressure difference and stopped when the initial fluctuations were damped sufficiently. Here, we show only the grid convergence of the pressure error in Figure 5.9 (for other details see [**7**]). As the Young-Laplace pressure is sensitive to curvature errors, it is found that the order of convergence is influenced by the order of the curvature reconstruction. It reveals that using third and fourth order least squares estimation gives approximately a second order convergence of the pressure.

5.2.2.2. *Coalescence of bubbles.* The dynamics of coalescence of bubbles plays a major role in many engineering processes like sintering [**83**]. The numerical simulation of coalescence is a test of robustness of interface modeling as the drops can undergo stretching, tearing and folding during mixing processes. We simulated the coalescence processes coupling level set and LB method. Initially at $t = 0$, two balls of radius $r = 6$ were placed in 64^3 computational domain. The center of the domain lies at the junction between the two balls. We have used the

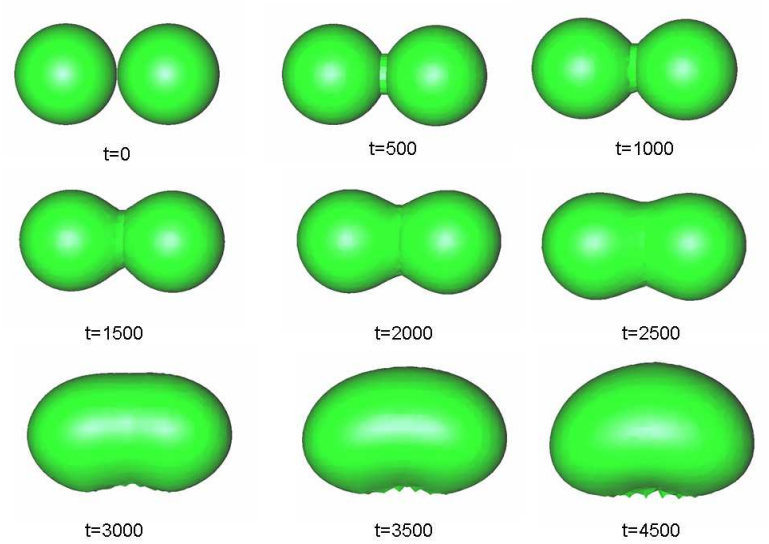


FIGURE 5.10. *Coalescence of bubbles driven by surface tension using a coarse grid.*

second order least squares estimation of curvature in this simulation. The simulation of the two bubbles is driven not only by the surface tension forces, but also has a velocity due to buoyancy. We conducted this experiment with two types of grid resolutions. Figure 5.10 shows the simulation where level set algorithm has the same grid as that of LB solver. Figure 5.11 is same as that of Figure 5.10, but the level set grid is refined by a factor of 2 along the interface.

As we see from Figure 5.10, the mixing for first few steps is in accordance with the physical processes, but later oscillations starts appearing along the base of the ball junction. In Figure 5.11, we find, due to refinement of grid there is no oscillation and the entire coalescence is much smoother than the coarser one. We also find the time of coalescence is different in the fine grid. In the coarse grid the coalescence starts much earlier. Moreover, we find a variation in the shape at various time steps when comparing two grid resolutions. Therefore, it is clear that in the study of coalescence of bubbles, the grid refinement is crucial to capture the interface correctly.

5.2.3. Injection mold-flow. An important aspect in the simulation of injection mold-flow is the representation of the free surface. The software SIGMASOFT® simulates injection mold-flow processes,

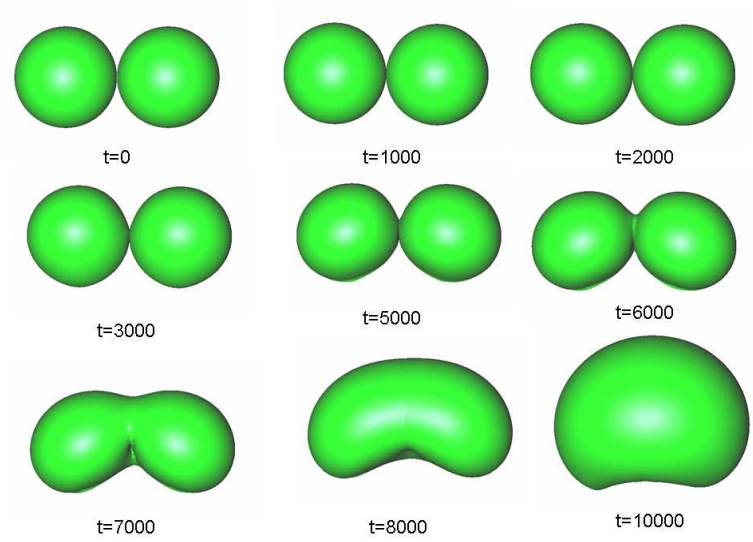
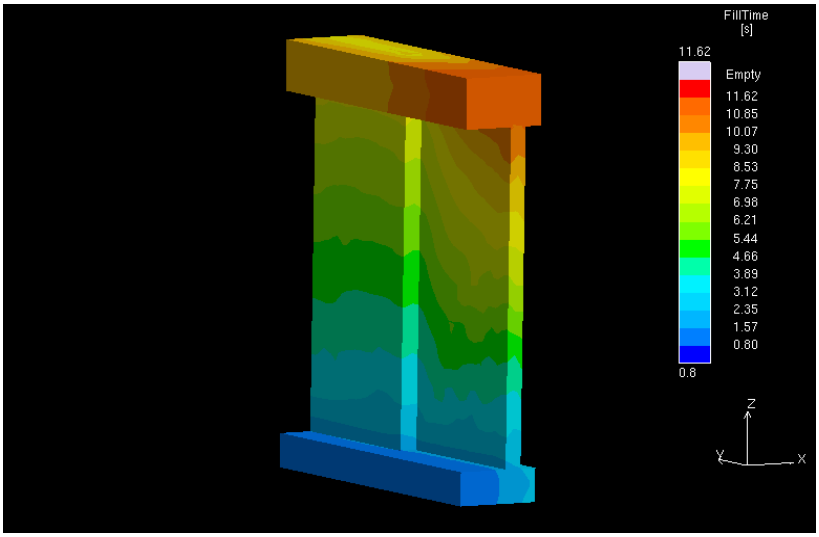


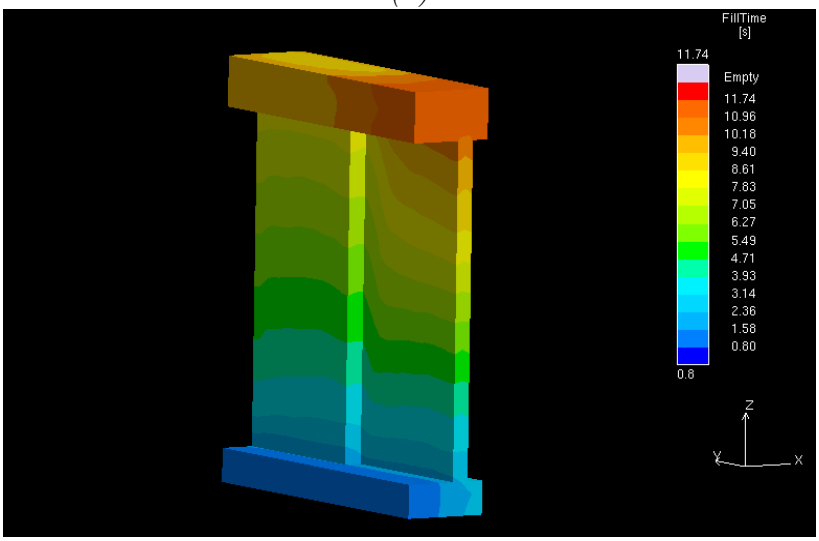
FIGURE 5.11. *Coalescence of bubbles driven by surface tension using a level set grid twice finer in the narrow band.*

including the cooling and heating phase for thermo-plastics, elastomers and thermosets (see *http://www.sigmasoft.de/* for details). This software uses 3D volume elements to solve the flow equation with the VOF method for representing the surface. To investigate the behavior of the interface with the level set method, we coupled our approach with VOF for some standard benchmark geometries. In this study, the volume of fraction computed from SIGMASOFT is corrected by the level set estimation. The reinitialization is done explicitly with Φ estimated from the triangulation. We used the least squares method for the extrapolation of the velocity field and constant velocity extension for finding the normal velocity on the grid points. We took the same coarse grid used by the SIGMASOFT flow solver (without refinement) for storing the level set data.

We simulated the mold-flow of SIGMASOFT with and without level set coupling for some standard geometries. Here, we present two examples. Figure 5.12 shows the first example where the object has a varying thickness along the xy plane. Figure 5.12(a) is the fill-up of mold flow over time without and 5.12(b) with level set coupling. We found that the evolution of the free surface at each fill-up time with the level set coupling is smooth, and has a sharp jump across different



(a)



(b)

FIGURE 5.12. *Flow fill-up over time for non-uniform thick object (a) without and (b) with level set coupling.*

thickness. On the other hand, the evolution of the free surface without level set coupling is found to be oscillatory and has a smeared jump across different thickness.

The second example is a spherically symmetric mold with inlet at the center. Figure 5.13(a) is the fill-up without and 5.13(b) with level set coupling. Similar to the first example, the free surface evolves smoothly with the the level set coupling. The interesting aspect in

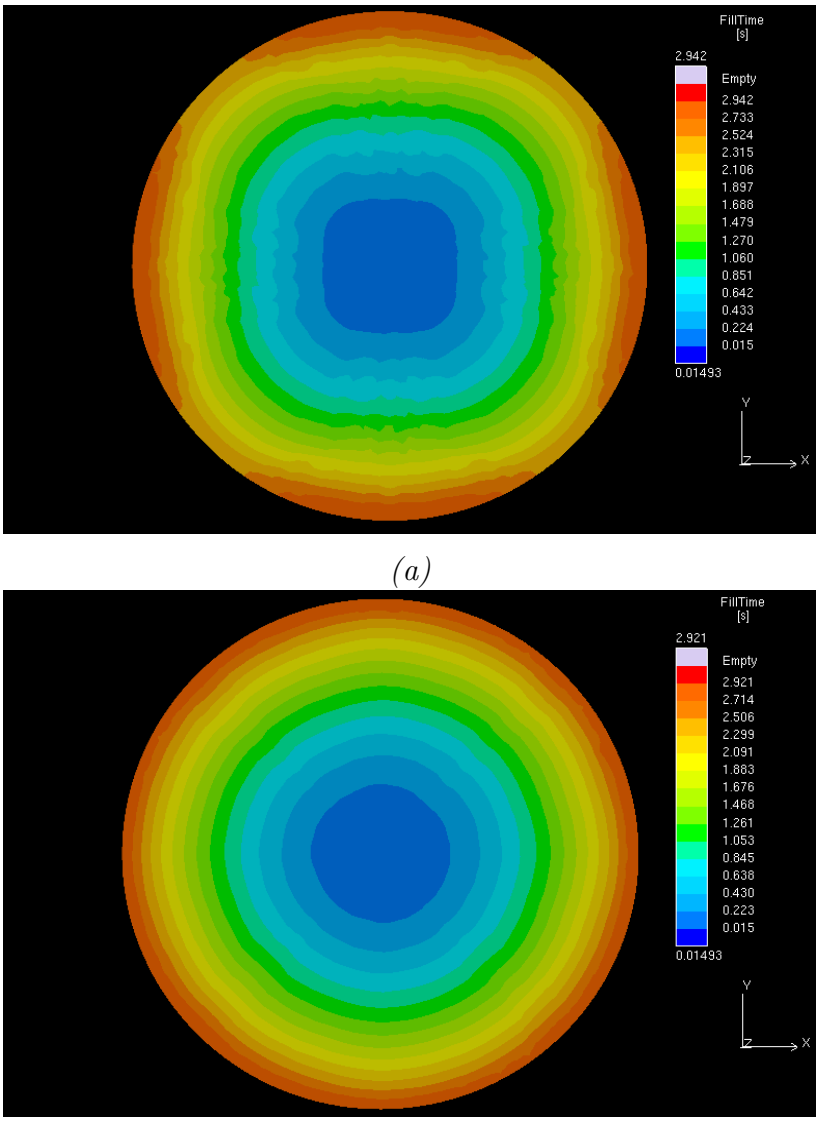


FIGURE 5.13. *Flow fill-up over time for spherically symmetric object (a) without and (b) with level set coupling.*

this study is the pattern of filling with and without level set coupling. Since the mold is uniformly thick, the filling at each time step should be spherically symmetric. It is found that the simulation with the level set coupling maintains this spherically symmetric pattern as shown in Figure 5.13(b). On the contrary, the filling pattern without level set coupling lacked spherical symmetry at each time step and the transport of flow has a preferential direction along the diagonal.

This simulation suggests that the level set method can be advantageous for the treatment of the interface for the mold-flow problems. The main disadvantage we faced in the above experiment is the "mass loss " at each time step as we have used a very coarse grid from the flow solver. Thus, the filling was found to be slower with the level set coupling than without it at each time step. This increases the computational time, especially for large geometries. For these practical applications level set methods require a refined mesh along the interface combined with robust mass correction algorithms. But incorporating the refinement and the mass conservation algorithm, without increasing the computational time is a real challenge for the treatment of the interface.

CHAPTER 6

Summary and Conclusions

A modular level set algorithm is developed to study the interface and its movement for free moving boundary problems. The algorithm is divided into three basic modules *initialization*, *propagation* and *contouring*.

A methodology is discussed to find an accurate signed distance function Φ from a closed, simply connected surface discretized by triangulation. We compute Φ using the direct method and it is stored efficiently in the neighborhood of the interface by a narrow band level set method. A novel approach is employed to determine the correct sign of the distance function at convex-concave junctions of the surface. The accuracy and convergence of the method with respect to the surface resolution is studied. It is shown that the efficient organization of surface and narrow band data structures enables the solution of large industrial problems. We compared the accuracy of Φ by the direct approach with the Fast Marching Method (FMM). It is found that the direct approach is more accurate than FMM. With respect to the speed of computation, FMM is faster than the direct method for coarse meshes. On the other hand, the direct method is faster than FMM for fine mesh width.

Contouring is performed through a variant of the marching cube algorithm used for the isosurface construction from volumetric data sets. The algorithm is designed to keep foreground and background information consistent, contrary to the neutrality principle followed for surface rendering in computer graphics. The algorithm ensures the isosurface triangulation is closed, non-degenerate and non-ambiguous. The constructed triangulation has desirable properties required for the generation of good volume meshes. These volume meshes are used in the boundary element method for the study of linear electrostatics.

For accurately estimating surface properties like interface position, normal and curvature from a discrete level set function, a method based

on higher order weighted least squares is proposed. An arbitrarily oriented torus is taken as a test object to study the accuracy for the estimation of surface parameters. It is found that the least squares approach is more accurate than finite difference approximations. Furthermore, the method of least squares requires a more compact stencil than those of finite difference schemes. If the object is given as a surface triangulation, then the accuracy and convergence of the method depends on the surface resolution and the discrete mesh width. Moreover, we find the surface parameters estimated by the direct method is accurate than FMM.

This approach is used in the propagation for the study of mean curvature flow and bubble dynamics. The advantage of this approach is that the curvature is not discretized explicitly on the grid and is estimated on the interface. The method of constant velocity extension is employed for the propagation of the interface. With the least squares approach, the mean curvature flow has a considerable reduction in mass loss compared to finite difference techniques. In the bubble dynamics, the modules are used for the study of a bubble under the influence of surface tension to validate Young-Laplace law. It is found that the order of curvature estimation plays a crucial role for calculating accurately the pressure difference between inside and outside of the bubble. Further, we study the coalescence of two bubbles under surface tension forces.

Our approach is coupled with VOF for the study of mold filling simulation using SIGMASOFT. It is found from the benchmark experiment, that the interface evolves smoothly with the level set-VOF coupling. On the other hand, filling is found to be slow due to mass loss in the under resolved regions.

Future Aspects

The future study which we would like to incorporate in our approach for various modules are as follows:

(1) Initialization:

- We showed that CPU time to estimate Φ varies cubically with the narrow band width w and linearly with the number of surface triangles. This is due to the fact that we

have chosen a local cuboid around each triangle. In 2D, we showed that we can split up the local window to sub windows to reduce the number of grid points within the narrow band. A similar result is also required in 3D. This will be useful, especially to reduce the computational time for the estimation of Φ by the direct method.

(2) Contouring:

- We showed the topological modification of the isocontour with different mesh resolution in Chapter 3. For practical applications it is desirable to have a robust algorithm, such that the change in the topology of the isosurface is not drastic. There are many strategies to do this task. One option is to look for a larger number of neighboring cubes during marching. Another approach is to go for a local refinement of grid or field values at the center of the cube to get exact surface representation [**91**] or use trilinear surface representation [**17**] within the cell.
- We found that the quality of the surface triangles is good when field values are given like a binary image. On the other hand, if the field values are real and close to the given threshold value, then the marching cube can generate a small and skewed triangle. Generating a volume from the surface mesh in this case is difficult. A good option is to change the edge connectivity for these small triangles and then quality of triangles can be improved upon without compromising the quality of isosurface.

(3) Propagation:

- It is well known that level set methods tend to lose mass during propagation. Here, we have not addressed this problem in our study. It is found that during reinitialization of Φ, the interface may move from the zero level set making it vulnerable for mass loss/gain locally. To avoid mass loss, several corrective measures are used in the literature during reinitialization [**89**], [**70**], [**22**]. In the context of level set coupling with Lattice-Boltzmann

code, we are using the principle of dilation locally to conserve mass. We need to investigate further this aspect so that it can be suitable for variety of applications.

Apart from these modules, we are working to parallelize the modules to be able to to solve over larger problems.

APPENDIX A

Higher order finite difference scheme

Let discrete Φ_{ijk} is given on each grid node for a uniform mesh width h. We give here the sixth order finite difference formulas for the derivatives at (i, j, k). For Φ_x we have

$$\begin{aligned} \Phi_x &\approx \frac{1}{60h}\left[-\Phi_{i-3,j,k} + \Phi_{i+3,j,k} + 9\Phi_{i-2,j,k} - \right. \\ &\left. 9\Phi_{i+2,j,k} - 45\Phi_{i-1,j,k} + 45\Phi_{i+1,j,k}\right], \end{aligned} \tag{A.1a}$$

and a similar relation holds for Φ_y and Φ_z along j and k direction respectively. For Φ_{xx} we have

$$\begin{aligned} \Phi_{xx} &\approx \frac{1}{90h^2}\left[\Phi_{i-3,j,k} + \Phi_{i+3,j,k} - \frac{27}{2}\Phi_{i-2,j,k} - \right. \\ &\left. \frac{27}{2}\Phi_{i+2,j,k} + 135\Phi_{i-1,j,k} + 135\Phi_{i+1,j,k} - 245\Phi_{i,j,k}\right], \end{aligned} \tag{A.1b}$$

and a similar relation holds for Φ_{yy} and Φ_{zz} along j and k direction respectively. For the mixed derivatives Φ_{xy} we have

$$\begin{aligned} \Phi_{xy} &\approx \frac{1}{360h^2}\left[\Phi_{i+3,j+3,k} - \Phi_{i+3,j-3,k} - \Phi_{i-3,j+3,k} + \right. \\ &-\Phi_{i-3,j-3,k} + 27\Phi_{i+2,j+2,k} + 27\Phi_{i+2,j-2,k} + \\ &\left. 27\Phi_{i-2,j+2,k} - 27\Phi_{i-2,j-2,k}\right], \end{aligned} \tag{A.1c}$$

and a similar relation holds for Φ_{xz} and Φ_{yz} along (i, k) and (j, k) direction respectively.

Bibliography

1. Adalsteinsson, D., and Sethian, J.A. , *A level set approach to a unified model for etching deposition and lithography, I: Two dimensional simulation*, J. Comput. Phys. **120** (1995), 128–144.
2. ———, *A level set approach to a unified model for etching deposition and lithography, II: Three dimensional simulation*, J. Comput. Phys. **122** (1995), 348–366.
3. ———, *The fast level set methodfor propagating interfaces*, J. Comput. Phys. **118** (1995), 269–277.
4. ———, *A level set approach to a unified model for etching deposition and lithography, III: Re-deposition, re-meission, surface diffusion and complex simulations*, J. Comput. Phys. **120** (1997), 128–144.
5. ———, *The fast construction of extension velocities in level set methods*, J. Comput. Phys. **148** (1999), 2–22.
6. Aslam, T., *A partial differential equation approach to multidimensional extrapolation*, J. Comput. Phys. **193** (2004), 349–355.
7. Becker, J., Junk, M., Kherwald, D., Klar, A., Steiner, K., Thömmes, G., Vaikuntam, A.K., and Wiegmann, A., *A coupled Lattice-Boltzmann and level set method for immiscible two-phase flows*, (Manuscript under preparation).
8. Bern, M., Eppstein, D., and Gilbert, J., *Provably good mesh generation*, IEEE Symp. Found. Comput. Sc. **1** (1990), 231–241.
9. Bourke, P., *http://local.wasp.uwa.edu.au/ pbourke/dataformats/ply/.*
10. Bridson, R., *Computational Aspects of Dynamic Surfaces*, Ph.D dissertation, Stanford University, Stanford California, 2003.
11. Chang, Y.C., Hou, T.Y., Merriman, B., and Osher, S., *A level set formulation of Eulerian interface capturing methods for incompressible flows.*, J.Comput.Phys. **124** (1996), 449–464.
12. Chen, S., Merriman, B., Osher, S., and Smereka, P., *A simple level set method for solving Stefan problems*, J.Comput.Phys **135** (1997), 8–29.
13. Chernyaev, E.V., *Marching Cubes 33*, Tech. Report Tech report CN/95-15, CERN, 1995.
14. Chopp, D.L., *Some improvement of the fast marching method*, SIAM J. Sci. Comput. **23** (2001), 230–244.
15. Chopp, D.L., and Sethian, J.A. , *Motion by intrinsic Laplacian of curvature*, Interfaces and free boundaries **1** (1999), 1–18.

16. Chorin, A.J., *Flame advection and propagation algorithm*, J. Comput. Phys. **35** (1980), 1–11.
17. Cignoni P., Ganoveli, F., Montani, C., Scopigno, R. , *Reconstruction of topologically correct and adaptive trilinear surfaces*, Computers and graphics **24** (2000), 399–418.
18. Deckelnick, K., and Dzùik, G., *Mean curvature flow and related topics*, Frontiers in Numerical Analysis Durham 2002, Springer Verlag, Heidelberg, 2003, pp. 63–108.
19. Do Carmo, M.P., *Differential geometry of curves and surfaces*, Prentice-Hall, Englewood cliffs, New jersey, 1976.
20. Dürst, *Additional references to marching cubes*, Computer Graphics **22** (1988), 72–73.
21. Dzùik, G., *An algorithm for evolutionary surfaces*, Numer. Math **58** (1991), 603–611.
22. Enright, D., Fedkiw, R., Ferziger, J., and Mitchell, I., *A hybrid particle level set method for improved interface capturing*, J. Comput. Phys. **183** (2002), 83–116.
23. Evans, L.C., and Spruck, J. , *Motion of level sets by mean curvature I*, J. Diff.Geom **33** (1991), 635–681.
24. _____, *Motion of level sets by mean curvature II*, Trans. Amer. Math **330** (1992), 321–332.
25. Fedkiw, R., Aslam, T., and Xu, S., *The ghost fluid method for deflagaration and detonation discontinuities*, J. Comput. Phys. **154** (1999), 393–424.
26. Fedkiw, R., Sapiro, G., and Shu, C-W., *Shock capturing, level sets and PDE based methods in computer vision and image processing: A review on Osher's contribution*, J. Comput. Phys. **185** (2003), 309–341.
27. Frolkovic, P., and Mikula, K., *Flux-based level set method: A finite volume method for evolving interface*, Tech. Report Preprint 15, Interdisziplinäres Zentrum fur Wissenschaftliches Rechnen, Heidelberg, September 2003.
28. Gage, M.E., and Hamilton, R.S., *The heat equation shrinking convex plane curves*, J. Diff. Geom. **23** (1986), 69–96.
29. Gallier, J., *Geometric Methods and applications - For computer science and engineering*, Springer verlag, 2001.
30. Gao, J., Zhang, J, and Suri, J.S., *Motion imagery segmentation via PDE*, Kluwer Academic Newyork, 2002.
31. GEOMVIEW, *http://www.geomview.org/*.
32. Gibou, F., and Fedkiw, R., *A fourth order accurate discretization for the Laplace equations on arbitrary domains, with applications to the Stefan problem*, J. Comput. Phys **202** (2005), 577–601.
33. Gray, A., *Modern differential geometry of curves and surfaces with MATHEMATICA*, CRC press, 1998.

34. Grayson, M.A.,, *The heat equation shrinks embedded plane curves to round points*, J. Diff. Geom **26** (1987), 285–314.
35. Gross, S., Reichelt, V., Reusken, A., *A finite element based level set method for two-phase incompressible flows*, IGPM report Nt:243, RWTH, Aachen, Germany, 2004.
36. Harlow, F. H., and Welch, J.E., *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, Phys. Fluids **8** (1965), 2182–2189.
37. Harten, A., Enquist, B., Osher, S., and Chakravarthy, S., *Uniformly higher order accurate essentially non oscillatory schemes III*, J. Comput. Phys. **71** (1987), 231–303.
38. Hirt, C.W., and Nicholas, B.D., *Volume of Fluid (VOF) method for dynamics of free boundaries*, J. Comput. Phys. **39** (1981), 201–225.
39. Houston, B., Nielson, M.B., Batty, C., Nielsson, O., and Museth, K., *Hierarchical RLE level set: A compact and versatile deformable surface representation*, ACM Trans. Graph **25** (2006), 151–175.
40. Houston, B., Wieby, M., and Batty, C., *RLE sparse level set*, Proceedings of the SIGGRAPH '04 on sketches and applications, ACM Press, New York, 2006.
41. Huang, W.Z., Ren, Y., and Russel, R.D., *Moving mesh methods based on moving mesh partial differential equation*, J. Comput. Phys. **113** (1994), 279–290.
42. Huisken, G., *The volume preserving mean curvature flow*, J. Reine Angew. Math. **382** (1987), 35–48.
43. Leveque, R., *Finite difference methods for differential equations*, Lecture notes, University of Washington, Seattle, USA,
Link: http://www.amath.washington.edu/ rjl/pubs, 2006.
44. Lingrand, D., *http://www.essi.fr/ lingrand/MarchingCubes/algo.html*.
45. Liu, X-D., Osher, S., and Chan, T., *Weighted essentially non oscillatory schemes*, J. Comput. Phys. **126** (1996), 202–212.
46. Lorensen, W.E., and Cline, H.E., *Marching Cubes: A high resolution 3D surface construction algorithm*, SIGGRAPH 87 Conference Proceedings, Computer Graphics **21** (1987), 163–169.
47. Lorentz, A. R., *Multivariate Birkhoff interpolation*, Lecture Notes in Mathematics ed., Springer Verlag, Heidelberg, 1992.
48. Losasso, F., Fedkiw, R., and Osher, S. , *Spatially adaptive techniques for level set methods and incompressible flow*, Computers and Fluids **35** (2006), 995–1010.
49. Losasso, F., Gibou, F. and Fedkiw, R., *Simulating water and smoke with an octree data structure*, ACM SIGGRAPH - 2004, ACM Press, New York, 2004, pp. 457–462.

50. Malladi, R., and Sethian, J. , *Image processing via level set curvature flow*, Proceedings of the National Academy of Sciences **92** (1995), 7046–7050.
51. Malladi, R., Sethian, J., and Vemuri, B. , *Shape modeling with front propagation: A level set approach*, IEEE Trans. on Pattern Analysis and machine inteligence **17** (1995), 158–175.
52. Martin, A.D, *http://www.varlog.com/products/admesh/*.
53. Mauch, S, *Efficient algorithms for solving static Hamiltonian-Jacobi equations*, Ph.D dissertation, California Institute of Technology, Pasadena California, 2003, pp. 1–237.
54. Monaghan, J.J., *Simulating free surface flows with SPH*, J.Comput.Phys **110** (1994), 399–406.
55. Museth, K., Breen, D.E., Whitaker, R.T., and Barr, A.H. , *Level set surface editing operators*, ACM Trans. Graphics **21** (2002), 330–338.
56. Nakayama, T., and Mori, M., *An Eulerian finite element method for time-dependendent free surface problems in hydrodynamics*, Int. J. Num. Meth. Fluids **22** (1996), 175–194.
57. Natarajan, B., *On generating topologically consistent iso-surfaces from uniform samples*, The Visual Computer **11** (1994), 52–62.
58. Nielson, M.B., and Museth, K. , *Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets*, J. Sci. Comput. **26** (2006), 1–39.
59. Nielson, N.M., and Hamann, B., *The asymptotic decider: Resolving the ambiguity in marching cubes*, IEEE Visualization - Proceeding of 2nd Conference on Visualization '91 (1991), 83–91.
60. Noh, W.F., and Woodward, P.R., *SLIC - A simple line interface calculation*, Lecture notes in Physics 59 - Fifth conference on fluid dynamics 2002, Springer Verlag, Heidelberg, 1976, pp. 330–340.
61. Of G., Steinbach, O., and Wendland, W.L., *Application of fast multi-pole Galerkin boundary element method in linear electrostatics*, Comput. Visual Sci. **8** (2005), 201–209.
62. Osher, S., and Fedkiw, R., *Level set methods and dynamic implicit surfaces*, Springer Verlag New York, 2003.
63. Osher, S., and Sethian, J., *Fronts propagating with curvature-ependent speed: Algorithms based on Hamilton-Jacobu formulations*, J. Comput. Phys. **79** (1988), 12–49.
64. Peng, D., Merriman, B., Osher, S., Zhao, H-K., and Kang, M. , *A PDE based fast local level set method* , J. Comput. Phys. **120** (1999), 278–304.
65. Persson, P-O., *Mesh generation of implicit geometries*, Ph.D Dissertation, Massachusetts Institute of technology, USA, 2005.
66. Pilliod, J.E., *An analysis of piecewise linear interface algorithm for volume-of-fluid dynamics*, Master's thesis, University of California at Davis, 1992.

67. Pozrikidis, M.P., *Boundary integral and singularity methods for linearized viscous flow*, Cambridge University Press, 1992.
68. Pucket, E.G., Almgreen, A.S., Bell, J.B., Marcus, D.L., and Rider, W.J., *A high-order projection methods for tracking fluid interfaces in variable density incompressible flows*, J. Comput. Phys. **130** (1997), 269–282.
69. Rider, W.J., and Kothe, D.B., *Reconstructing volume tracking*, J. Comput. Phys. **141** (1998), 112–152.
70. Russo, G., and Smereka, P. , *A Remark on computing distance function*, J. Comput. Phys. **163** (2000), 51–67.
71. Rutka, V., *Immersed Interface Methods for Elliptic Boundary Value Problems*, Ph.D Dissertation, TU Kaiserslautern, Germany, 2005, pp. 1–145.
72. Rutka, V., and Wiegmann, A., *Explicit Jump Immersed Interface Method for virtual material design of the effective elastic moduli of composite materials*, Tech. Report 73, Fraunhofer ITWM Kaiserslautern, 2005.
73. ———, *Explicit Jump Immersed Interface Method for virtual material design of the effective elastic moduli of composite materials*, Numerical Algorithms (2007), (to appear).
74. Rutka, V., Wiegmann, A., and Andrä, H., *EJIIM for calculation of effective elastic moduli in 3D linear elasticity*, Tech. Report 93, Fraunhofer ITWM Kaiserslautern, 2006.
75. Sapiro, G., *Geometric partial differential equation and image analysis*, Cambridge University press, 2001.
76. Scardovelli, S., and Zaleski, S., *Direct numerical simulation of free-surface and interfacial flow*, Ann. Rev. Fluid Mech. **31** (1999), 567–603.
77. ———, *Interface reconstruction with least-square fit and split Eulerian-Lagrangian advection*, Int. J. Num. Meth. Fluids **41** (2003), 251–274.
78. Sedgewick, R., *Algorithms in C++*, Addison-Wesley Longman Inc, 1998.
79. Sethian, J., and Wiegmann, A., *Structural boundary design via level set and immersed interface method*, J. Comp. Phys. **163** (2000), 489–528.
80. Sethian, J.A., *Numerical algorithm for propagating interfaces: Hamilton-Jacobi equations and conservation laws*, J. Diff. Geom. **31** (1990), 131–161.
81. ———, *Curvature flow and entropy conditions applied to grid generation*, J. Comput. Phys. **115** (1994), 440–454.
82. ———, *Fast marching methods*, SIAM Rev. **41** (1999), 199–235.
83. ———, *Level set methods and fast marching methods - Evolving interfaces in Computational geometry, Fluid mechanics, Computer vision and Materials science*, Cambridge Univ. Press, U.K., 1999.
84. Shin, S., and Juric, D., *Using level contour reconstruction method for front tracking without connectivity*, J. Comp. Phys. **180** (2002), 427–470.
85. Smereka, P., *Semi implicit level set methods for curvature and surface diffusion motion*, J. Sci. Comput. **19** (2003), 439–456.
86. Spanier, E.H., *Algebraic Topology*, McGraw-Hill New York, 1966.

87. Strain, J.A., *Tree methods for moving interfaces*, J. Comp. Phys. **151** (1999), 616–648.
88. Suri, J.S., Singh, S., and Laxminarayan, S., *Medical image segmentation using level sets*, Kluwer Academic, New york, 2002.
89. Sussmann, M., Smereka, P., and Osher, S. , *A level set approach for computing solutions to incompressible two phase flow*, J. Comput. Phys. **114** (1994), 146–159.
90. Unverdi, S.O., and Tryggvason, G., *Computation of multi-fluid flows*, Physica D **60** (1992), 70–83.
91. Van Gelder, A. and Wilhelms, J., *Topological considerations of iso-surface generation*, ACM Transaction on Graphics **13** (1994), 337–375.
92. Whitaker, R.T, *Iso-surface and level set surface models*, Technical report UUCS-02-010, School of Computing, University of Utah UT84112, USA, 2002.
93. Wiegmann, A., *http://www.geodict.com/*.
94. _____, *The explicit jump immersed interface method and interface problems of differential equation*, Ph.D Dissertation, University of Washington, Seattle, USA, 1998.
95. Wiegmann A., and Bube, K., *The explicit jump immersed interface method: Finite difference methods for pde with piecewise smooth solutions*, SIAM J. Num. Anal. **37** (2000), 827–862.
96. Wiegmann, A., Vaikuntam, A. K., Teichmann, E., Andrä, H., and Becker, J., *Some industrial challenges to level set methods data sets*, Workshop on level set methods for direct and inverse problems (University of Linz, Austria), Sept14-16 2005.
97. Wilhelms, J., and Van Gelder, A., *Topological considerations of iso-surface generation extended abstract*, Computer Graphics **24** (1990), 79–86.
98. Youngs, D.L., *Time-dependent multi-material flow with large fluid distortion*, Academic-Press, New York, 1982.
99. _____, *An interface tracking method for a 3D-Eulerian hydrodynamics code*, Technical report 44/92/35, AWRE, 1984.
100. Zhang, J, and Liu, W., *Motion image segmentation using deformable models*, Kluwer Academic Newyork, 2002.

CURRICULUM VITAE

17 March, 1964	Born in Madurai, India
1982	All India Higher Secondary School Leaving Certificate Examination.
1982 – 1985	Bachelor of Physics, Madurai-Kamaraj University, Madurai, India.
1985 – 1987	Master of Physics, Madurai-Kamaraj University, Madurai, India.
1988- Jan 1990	Master of Technology (Atmospheric Science & Technology), Department of Physics & Meteorology, Indian Institute of Technology, Kharagpur, India.
1990 - 1996	Teaching Assistant, Department of Physics & Meteorology, Indian Institute of Technology, Kharagpur, India.
1996 - 1998	Master of Science (Industrial Mathematics), AG- Technomathematics University of Kaiserslautern, Germany
1998-2001	Project Associate, Joint Advanced Technology Project(JATP) Indian Institute of Science, Bangalore, India.

From 2001	Scientific staff, Dept: Flow and Material Simulation (Previously: Flow and Complex structure), Fraunhofer ITWM, Kaiserslautern, Germany.
Sept 2003	Registered for Doctoral studies, University of Kaiserslautern, Department of Mathematics and Fraunhofer Institute Techno- und Wirtschaftsmathematik (ITWM) Kaiserslautern, Germany.