Title
Regular Grid Based Methods for Fluid Simulation and Meshing Self-Intersecting Surfaces

Author
Gagniere, Steven William

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Regular Grid Based Methods for Fluid Simulation and Meshing Self-Intersecting Surfaces

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

by

Steven William Gagniere

2022

ABSTRACT OF THE DISSERTATION

Regular Grid Based Methods for Fluid Simulation
and Meshing Self-Intersecting Surfaces

by

Steven William Gagniere
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2022
Professor Joseph M. Teran, Chair

Regular grids are ubiquitous across computation mathematics, appearing in areas such as the numerical solution of differential equations, rendering in computer graphics, and mesh generation. In this dissertation we present two methods utilizing a regular background grid. We also discuss miscellaneous smaller contributions.

The first is a hybrid Lagrangian/Eulerian advection and projection method for fluid simulations which employs a regular grid both for Eulerian advection and for sub-grid-cell representation of irregular computational domains using a variation of the Marching Cubes algorithm. This method uses a Chorin splitting of the advection and pressure terms in the (incompressible) Euler equations. We present a novel backward semi-Lagrangian method using quadratic B-splines for velocity interpolation during the advection step. We additionally use B-spline interpolation over a regular grid in a variational technique for the pressure projection step.

The second is a method for creating volumetric meshes to represent the interior of self-intersecting input surfaces with emphasis on efficiency and the minimization of costly exact

or adaptive arithmetic. Standard approaches assume that the input surface is free of self-intersection, but in practice surface meshes have some amount of self-intersection. Our approach generates an embedded hexahedron mesh where each hexahedral element is a copy of a background grid cell. Regions of self-intersection are resolved by using multiple copies of the grid cells, with connectivity determined by the behavior of the input surface. While sufficiently high resolution is occasionally required to correctly resolve self-intersections, we present a topology preserving coarsening method to reach the desired lower resolution.

The first of the smaller contributions is a computation of the eigenstructure of the Hessian for a surface tension energy density model used in an updated-Lagrangian method for simulating mid-to-extreme surface tension forces. The second is a proof using B-spline techniques of the simplified form of the inertia tensor from the Affine Particle-in-Cell (APIC) method.

The dissertation of Steven William Gagniere is approved.

Jeffrey D. Eldredge

Marcus Leigh Roper

Luminita Aura Vese

Joseph M. Teran, Committee Chair

University of California, Los Angeles

2022

To my parents, David Gagniere and Pan Kueifeng

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Professor Joseph Teran, for his support, instruction, and time throughout the years of research. I would also like to thank the members of my doctoral committee: Professors Luminita Vese, Marcus Roper, and Jeffrey Eldredge.

I'm extremely grateful to fellow lab member David Hyde, who has provided invaluable assistance in the form of brainstorming, coding, technical support, and system maintenance for me and all of my fellow lab members. I also want to thank the rest of my fellow lab members whom I've had the pleasure of working with throughout my years in the program: Alan Marquez-Razon, Victoria Kala, Ayano Kaneda, Jingyu Chen, Elias Gueidon, Yizhou Chen, Yushan Han, Stephanie Wang, Mengyuan Ding, Xuchen Han, Qi Guo, and Ziheng Ge.

I would also like to thank my good friends Nicholas Tammadge, Caleb Choban, and Camille Bernal, as well as all of my other good friends for their support.

And last, but certainly not least, I want to express my deepest gratitude to my parents for their constant love, support, and encouragement.

The contents of Chapter 2 are a version of [GHM20a]; an expanded description of cut cells has been added along with various style and grammar updates. The contents of Section 4.1 are a version of material contained in the supplementary material for [HGM20]. These works were supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort between the U.S. DOE Office of Science and DOE National Nuclear Security Administration. These works were additionally supported by DOE Oak Ridge National Laboratory contract 4000171342. Chapter 3 and Section 4.2 are adapted from manuscripts currently submitted and under review.

VITA

2015	B.S. (Mathematics) and B.S. (Physics w/ specialization in Astrophysics), UCSD.
2016-2021	Teaching Assistant, Mathematics Department, UCLA.

PUBLICATIONS

Gagniere, S., Smith, S. G. L., & Yeh, H. D. (2018). Excess pore water pressure due to ground surface erosion. Applied Mathematical Modelling, 61, 72-82.
https://doi.org/10.1016/j.apm.2018.03.041

Wang, S., Ding, M., Gast, T. F., Zhu, L., Gagniere, S., Jiang, C., & Teran, J. M. (2019). Simulation and visualization of ductile fracture with the material point method. Proceedings of the ACM on Computer Graphics and Interactive Techniques, 2, 1-20.
https://doi.org/10.1145/3340259

Gagniere, S., Hyde, D., Marquez-Razon, A., Jiang, C., Ge, Z., Han, X., Guo, Q. & Teran, J. (2020). A Hybrid Lagrangian/Eulerian Collocated Velocity Advection and Projection Method for Fluid Simulation. Computer Graphics Forum, 39, 1-14.
https://doi.org/10.1111/cgf.14096

Hyde, D. A., Gagniere, S. W., Marquez-Razon, A., & Teran, J. (2020). An implicit updated lagrangian formulation for liquids with large surface energy. ACM Transactions on Graphics (TOG), 39, 1-13. https://doi.org/10.1145/3414685.3417845

CHAPTER 1

Introduction

1.1 Overview

In this dissertation we present meshing and simulation methods based on a regular background grid. First, we introduce a hybrid Lagrangian/Eulerian method for the simulation of incompressible fluids which naturally handles irregular flow domains on a regular grid through a cut-cell approach utilizing a variation on the marching cubes algorithm; our variant is an extension of the approach by Bhaniramka et al. [BWC04]. The incompressible Euler equations are discretized using a B-spline mixed FEM method with a Chorin splitting of the governing equations, where velocity degrees of freedom are collocated at the centers of a regular grid and pressure degrees of freedom are located on cell vertices. We also introduce a Backward semi-Lagrangian quadratic B-spline (BSLQB) technique for velocity interpolation which is second order accurate in space and time. We discuss this method in detail Chapter 2.

Second, we introduce an algorithm for generating an embedding hexahedron volume mesh from a triangle surface mesh with self-intersections to accurately represent the self-intersecting volume bounded by the triangle surface mesh; the steps in this algorithm are designed with efficiency in mind by minimizing the use of exact and/or adaptive arithmetic. The hexahedral elements in the output mesh of this algorithm are formed as copies of regular background grid cells which are duplicated an appropriate number of times and connected in a manner which respects the surface mesh. As the algorithm requires sufficient resolution

often finer than the target resolution of the user in order to resolve regions of high curvature and other small-scale features, we also introduce a coarsening method to reach the desired resolution while preserving the correct topological features. Additionally, we present a hexahedron to tetrahedron conversion method which again respects the topology. We cover this algorithm in Chapter 3.

We further present the following minor contributions: a computation of the eigenstructure for the Hessian of the surface tension energy defined in [HGM20] and a proof of the inertia tensor expression for the APIC method introduced in [JSS15]. These contributions are the contents of Chapter 4.

The remainder of Chapter 1 covers some relevant mathematical and physical background: continuum mechanics, the marching cubes algorithm, and graph traversal algorithms.

1.2 Background

1.2.1 Continuum Mechanics

In continuum mechanics, any material under consideration (solid or fluid) follows the continuum assumption: the material is continuous instead of discrete (even though every such material is fundamentally comprised of atoms, molecules, etc.). The contents of this section are based on [GS08].

1.2.1.1 Mass & Force

Let Ω denote the region of space occupied by the material. In accordance with the continuum assumption, we assume the existence of a mass density field denoted by $\rho : \Omega \to \mathbb{R}$. The mass of any open subset $U \subseteq \Omega$ is then given by the integral of ρ over that subset:

$$\text{mass}(U) = \int_U \rho \, d\mathbf{x}. \tag{1.1}$$

The volume of U is similarly defined by integrating 1 over U.

Forces can be classified as either external body forces (e.g. gravity) or surface forces resulting from contact (both internal and external). The only body force we shall consider is gravity. Letting $\mathbf{g}$ denote the acceleration due to gravity, the force due to gravity on $U \subseteq \Omega$ is given by the integral

$$\mathbf{f}_g(U) := \int_U \rho \mathbf{g} \, d\mathbf{x}. \tag{1.2}$$

To describe the surface force, first let Γ denote an oriented surface in Ω which is not necessarily a subset of $\partial\Omega$. The force per unit area exerted on material on one side of this surface by material on the other side is given by a traction field $\mathbf{t}$, with the force exerted over the whole of Γ given by

$$\mathbf{f}_s(\Gamma) := \int_\Gamma \mathbf{t} \, ds. \tag{1.3}$$

This traction field depends only on position and normal vector, and can be written in terms of a second order tensor field as

$$\mathbf{t} = \boldsymbol{\sigma}\mathbf{n}, \tag{1.4}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor field and $\mathbf{n}$ is a normal vector. The total force on $U \subseteq \Omega$ is the sum

$$\mathbf{f}(U) = \mathbf{f}_g(U) + \mathbf{f}_s(\partial U). \tag{1.5}$$

1.2.1.2 Flow

Denote the initial region occupied by the material (called the reference configuration) by Ω^0 and the region occupied at a later time t by Ω^t. Let $\mathbf{X} \in \Omega^0$ be a point in the reference configuration, and let $\mathbf{x} \in \Omega^t$ be the corresponding point at time t. We assume the existence of a bijective deformation map (or flow map) $\boldsymbol{\phi}(\cdot, t) : \Omega^0 \to \Omega^t$, and we express the correspondence between $\mathbf{x}$ and $\mathbf{X}$ using the flow map as $\mathbf{x} = \boldsymbol{\phi}(\mathbf{X}, t)$. We further assume that the flow map is locally orientation preserving: it satisfies the inequality

$$\det \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} > 0, \tag{1.6}$$

where $\mathbf{F} := \partial\boldsymbol{\phi}/\partial\mathbf{X}$ denotes the deformation gradient.

We thus have two coordinate systems with which to describe the system: Lagrangian coordinates $\mathbf{X}$ corresponding to specific particles in the reference configuration and Eulerian coordinates $\mathbf{x}$ corresponding to spatial points. Any scalar, vector, or tensor field can be described using either coordinate system. For example, consider velocity: the velocity of a particle $\mathbf{X}$ at time t is defined by

$$\mathbf{V}(\mathbf{X}, t) := \frac{\partial \boldsymbol{\phi}}{\partial t}(\mathbf{X}, t). \tag{1.7}$$

We use the flow map inverse $\boldsymbol{\phi}^{-1}(\cdot, t) : \Omega^t \to \Omega^0$ to find the velocity $\mathbf{v}$ at a spatial point $\mathbf{x}$:

$$\mathbf{v}(\mathbf{x}, t) = \mathbf{V}(\boldsymbol{\phi}^{-1}(\mathbf{x}, t), t). \tag{1.8}$$

Likewise, any field expressed in Eulerian coordinates has an associated Lagrangian description. For example, consider the time t spatial density $\rho(\cdot, t) : \Omega^t \to \mathbb{R}$. The corresponding field $R(\cdot, t) : \Omega^0 \to \mathbb{R}$ in Lagrangian coordinates is then given by

$$R(\mathbf{X}, t) = \rho(\boldsymbol{\phi}(\mathbf{X}, t), t). \tag{1.9}$$

1.2.1.3 Balance Laws

The conservation of mass and the balance of momentum lead to localized Lagrangian and Eulerian balance laws. Given an open subset $U^t \subseteq \Omega^t$, there is a corresponding set U^0 such that $U^t = \boldsymbol{\phi}(U^0, t)$. Conservation of mass is then the statement

$$\text{mass}(U^t) = \text{mass}(U^0). \tag{1.10}$$

Using definition (1.1) of the mass in terms of the mass density ρ on both sides and using a change of variable to Lagrangian coordinates on the resulting LHS integral leads to the localized Lagrangian mass balance

$$R(\mathbf{X}, 0) = R(\mathbf{X}, t) \det \mathbf{F}(\mathbf{X}, t), \tag{1.11}$$

where R (defined in Equation (1.9)) is the Lagrangian counterpart of ρ. Here, $\mathbf{F}$ is the deformation gradient from Inequality (1.6). Taking the derivative of (1.11) and simplifying

eventually leads to the localized Eulerian form of the mass balance

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0. \tag{1.12}$$

The balance of momentum in integral form is

$$\frac{d}{dt} \int_{U^t} \rho \mathbf{v} \, d\mathbf{x} = \int_{\partial U^t} \boldsymbol{\sigma} \mathbf{n} \, ds + \int_{U^t} \rho \mathbf{g} \, d\mathbf{x}. \tag{1.13}$$

Let $\frac{D}{Dt}$ denote the material derivative. Using the property

$$\frac{d}{dt} \int_{U^t} \rho \mathbf{v} \, d\mathbf{x} = \int_{U^t} \rho \frac{D\mathbf{v}}{Dt} \, d\mathbf{x}, \tag{1.14}$$

which follows from Result 5.6 of [GS08], together with the divergence theorem on the surface integral leads to the localized Eulerian momentum balance equation

$$\rho(\mathbf{x}, t) \frac{D\mathbf{v}}{Dt}(\mathbf{x}, t) = \nabla \cdot \boldsymbol{\sigma}(\mathbf{x}, t) + \rho(\mathbf{x}, t)\mathbf{g}. \tag{1.15}$$

By employing a change of variable, the divergence theorem, and conservation of mass, we may derive the localized Lagrangian form:

$$R(\mathbf{X}, t) \frac{\partial \mathbf{V}}{\partial t}(\mathbf{X}, t) = \nabla \cdot \mathbf{P}(\mathbf{X}, t) + R(\mathbf{X}, 0)\mathbf{g}, \tag{1.16}$$

where $\mathbf{P}(\mathbf{X}, t) = \det \mathbf{F}(\mathbf{X}, t) \boldsymbol{\sigma}(\boldsymbol{\phi}(\mathbf{X}, t), t) \mathbf{F}(\mathbf{X}, t)^{-T}$ is the first Piola-Kirchhoff stress.

1.2.1.4 Incompressible Euler Equations

If we assume that a fluid satisfies the incompressibility condition

$$\nabla \cdot \mathbf{v} = 0 \tag{1.17}$$

and that the Cauchy stress is of the form $\boldsymbol{\sigma} = -p\mathbf{I}$ where p is a pressure field, we obtain the incompressible Euler equations from equation (1.15):

$$\rho \frac{D\mathbf{v}}{Dt} = \rho \left(\frac{\partial \mathbf{v}}{\partial t} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}} \mathbf{v} \right) = -\nabla p + \rho \mathbf{g}, \tag{1.18}$$

$$\nabla \cdot \mathbf{v} = 0. \tag{1.19}$$

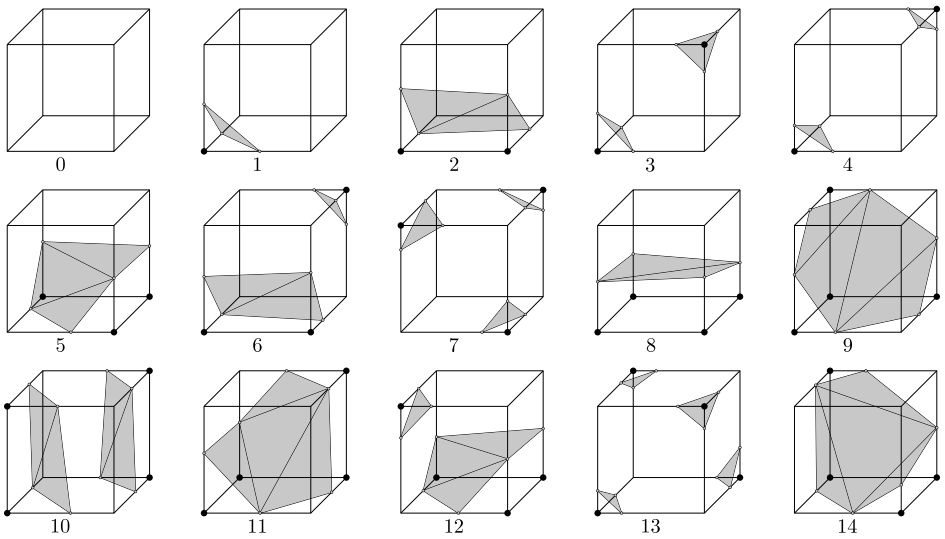


Figure 1.1: **Original Marching Cubes Cases.** The 15 marching cubes cases from [LC87] using the authors' original indexing. Each of the 256 possible choices of positive or negative values on the cell vertices can be associated to one of these cases through the use of various symmetries and reductions. Adapted from Figure 3 of [LC87].

In solving these equations, the pressure is also an unknown quantity in addition to the velocity $\mathbf{v}$. We solve these governing equations with appropriate boundary conditions in Chapter 2.

1.2.1.5 Lagrangian and Eulerian Methods

1.2.2 Marching Cubes

The marching cubes algorithm (also known as the marching squares algorithm in 2D) was first introduced in 1987 by Lorensen and Cline [LC87] to construct surface meshes from medical data. The basic concept of the algorithm is to generate a triangle mesh from a scalar field stored at the vertices of a regular grid by approximating the zero-isocontour defined by linear interpolation on these scalar field values. Each grid cell is associated with the corresponding 8 scalars stored at its vertices. For each edge whose scalar values at

its 2 incident vertices differ in sign, linear interpolation is used to find the position where the zero-isocontour intersects the edge. A local triangle mesh is then generated by using appropriate polygons to connect the intersection points on the edges. When a cell has been processed, the algorithm continues, or 'marches,' onto other cells. The global triangle mesh is then simply the union of the local grid cell polygon meshes. While in principle there are 256 different sign permutations on grid cell vertices, Lorensen and Cline use rotational symmetry and sign flips (on the scalar values) to reduce this number to 15 unique cases. Thus, the original version of the algorithm stores 15 polygon configurations in a lookup table and used the above symmetries to determine the correct configuration. We show these 15 configurations in Figure 1.1.

This original version contains significant drawbacks, however. First, it is possible for holes to develop in the global mesh (i.e. to generate a mesh which is not watertight). We demonstrate this issue using an example from Chernyaev [Che95] in Figure 1.2. Less seriously, the surface can fail to capture the topology of the zero-isocontour even when the resulting mesh is watertight. Both of these issues result from ambiguities on the faces and centers of some of the original 15 cases. Consider case 3 of Figure 1.1 (one of the two cases from Figure 1.2 which together form a hole). For this case there exists another local mesh which is not topologically equivalent but is still consistent with the scalar values on the vertices. In Figure 1.2 we show this case and demonstrate that its use resolves the hole shown in the same figure.

Many authors have since modified the original algorithm and case table to resolve these shortcomings. Chernyaev [Che95] expanded the original lookup table from 15 to 33 cases by adding sub-cases to those original cases which contained face or center ambiguities. He also provided tests to determine the correct sub-cases. Lewiner et al. [LLV03] further identified and resolved missing tests from [Che95] and provided an implementation. Other approaches expand the lookup table by discarding sign flips and using only rotations and/or reflections [RH99]; such lookup tables additionally consider cases with 5 to 8 positive scalar values.

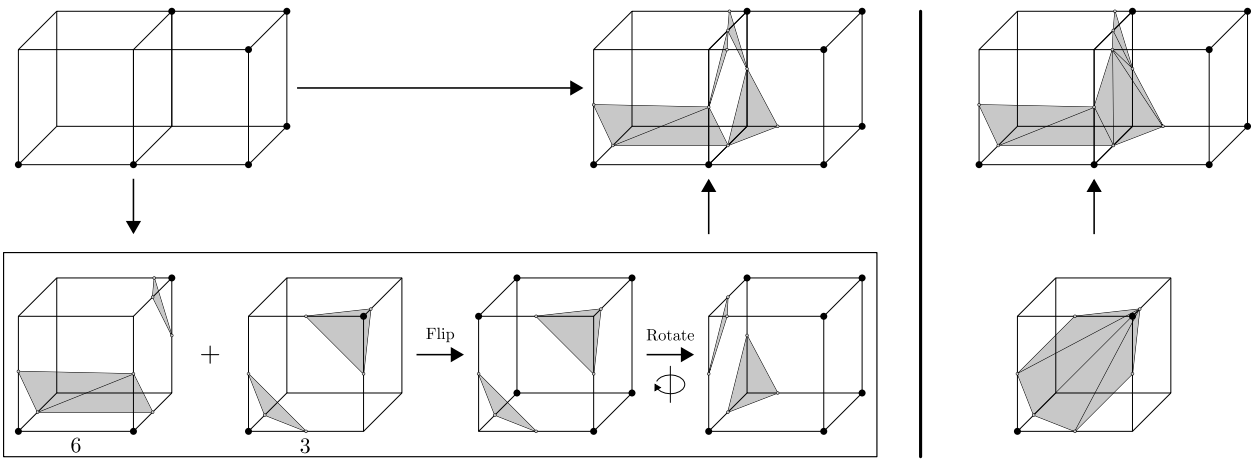


Figure 1.2: **Topological Holes in Marching Cubes. Left:** The original marching cubes algorithm applied to the adjacent cells in the upper left result in a hole in the surface. The left cell is the same as case 6 from Figure 1.1, while the right cell is reduced to case 3 by a sign flip and a rotation. This figure is adapted from Figure 2 of [Che95]. **Right:** Using an alternate choice of local triangle mesh for case 3, the same process generates a consistent mesh across the adjacent cells.

Banks and Linton [BL03] used computational group theory methods to count and verify the number of reduced cases from the full 256 based on whether sign flips are used and on the types of cube symmetries allowed; they do not, however, count the number of sub-cases required for topological correctness. In particular, they verify that the 15 cases from the original paper are indeed the minimal number required to generate 256 cases through sign flips and rotations.

Bhaniramka et al. [BWC04] take this even further by constructing the full 256 lookup table without considering symmetries or sign-flips. They generate each case in the lookup table by first taking the convex hull of the interior vertices (those vertices whose stored scalar value is negative) together with the sample intersection points at midpoints of intersected edges. Then they store the collection of boundary triangles on the convex hull which do not lie on cell boundaries; i.e. the triangles on the boundary which are interior to the

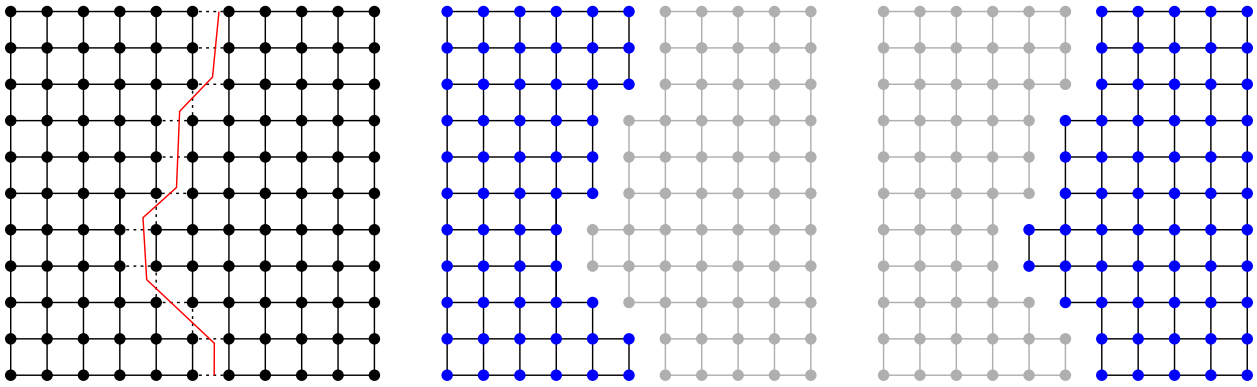


Figure 1.3: **Connected Components in a Graph. Left:** We may consider this regular grid as a graph where the vertices are grid vertices and the edges are the solid grid edges. In Chapter 3 we will 'cut' edges of the grid using polygonal lines such as the red line shown here. **Middle:** One of the connected components is shown in blue. **Right:** The other connected component is shown.

cell. We make use of their approach and extend it by also building a lookup table of the tetrahedra comprising the convex hull. In other words, we use their marching cubes approach to additionally generate a tetrahedralization of the volume bounded by the input surface mesh.

For a detailed accounting of the marching cubes literature, including performance advances and further algorithmic variants, see the survey by Newman and Yi [NY06].

1.2.3 Graph Traversal Algorithms

We now describe two common graph traversal algorithms, depth-first search (DFS) and breadth-first search (BFS), and their application to finding connected components (which we use extensively in Chapter 3). We first define a few of the basic notions related to graphs (note: we only consider undirected graphs here).

- A *graph* is a structure consisting of a set of vertices V and edges E which connect them; each edge is given by a pair of vertices (the endpoints of the edge).

Algorithm 1: DFS

```
Input: Adjacency list L (list of lists), input vertex v, list K
begin
    set visited[v] to true;
    add v to K;
    forall u in L[v] do
        if not visited[u] then
            DFS(L, u, K);
        end
    end
end
```

- An edge is *incident* to a vertex if that vertex is one of the two vertices in the pair.
- Two vertices are *adjacent* if they are the endpoints of some edge.
- A *path* is a sequence of alternating vertices and edges such that each edge is incident to the vertex before and after it in the sequence and no vertex is repeated.
- Two vertices are *connected* if there exists a path between them.
- A graph is *connected* if every pair of vertices are connected.
- Finally, a *connected component* is a maximal connected subgraph.

We illustrate connected components in Figure 1.3.

The DFS and BFS algorithms both begin at an input vertex and systematically visit all of the vertices connected to it, differing only in the manner in which the traversal is done.

The two most common data structures for representing a graph are the adjacency matrix and the adjacency list. Many variations exist for the specific implementations of these data structures. We shall use the following characterization of the adjacency list: a list of lists, where the i-th list contains all of the vertices which are adjacent to vertex i.

Algorithm 2: Connected Components

```
Input: Adjacency list L (list of lists)
Output: Connected components C (list of lists)
begin
    for v = 1 to length(L) do
        if not visited[v] then
            initialize new list K;
            DFS(L, v, K);
        end
        add K to C;
    end
end
```

1.2.3.1 DFS

We begin this algorithm by selecting one vertex and marking it as visited. We then select an unvisited vertex from the adjacency list and repeat the process recursively. This process of moving to adjacent unvisited vertices continues until we reach a vertex whose adjacent vertices have all been visited. At this point, we return until we have reached a vertex which still has unvisited neighbors, and repeat until all connected vertices have been visited.

This describes the basic process of DFS. With a slight modification to this process we can compute the connected components. Note that by the end of the algorithm, every vertex in the component containing the input will have been visited. So the idea is to call the algorithm repeatedly on remaining unvisited vertices until none remain. Our input to this algorithm will be the adjacency list. Our output will similarly be a list of lists: the i-th list contains all of the vertices in connected component i. This connected components DFS algorithm is presented in more detail in Algorithms 1 and 2. For simplicity we have shown the recursive version of DFS in Algorithm 1. In practice, however, stack overflow issues arise for large graphs and it is preferable to switch to an iterative formulation.

Algorithm 3: BFS

```
Input: Adjacency list L (list of lists), input vertex v, list K
begin
    set visited[v] ← true;
    add v to K;
    initialize new list R and add v to R;
    while R is not empty do
        initialize new list R_next;
        forall w in R do
            forall u in L[w] do
                if not visited[u] then
                    set visited[u] ← true;
                    add u to K;
                    add u to R_next;
                end
            end
        end
        set R ← R_next;
    end
end
```

1.2.3.2 BFS

Whereas the DFS algorithm proceeds by traversing as far as possible in a single direction, the BFS algorithm gradually spreads outwards from the input vertex. We do so by keeping track of the current "ring" of vertices, and successively building new rings out of the unvisited neighbors of vertices in the current ring. As before, we mark the starting vertex as visited. We additionally add this vertex to the starting ring. Then, for each vertex in the current ring we add all unvisited neighbors to the next ring and mark them as visited. This process continues until the next ring is empty. We show this in more detail in Algorithm 3.

The extension from the core BFS traversal to the connected components algorithm is essentially identical to the case of DFS; the only difference is which traversal algorithm is called. Hence, the complete algorithm consists of Algorithms 3 and 2 (substituting the call to DFS with a call to BFS in the latter).

For additional graph theory definitions and concepts, see [Cha77]. For a more in-depth explanation of DFS and BFS, complexity results, and additional applications, see the text by Goodrich et al. [GTG14].

CHAPTER 2

A Hybrid Lagrangian/Eulerian Collocated Velocity Advection and Projection Method for Fluid Simulation

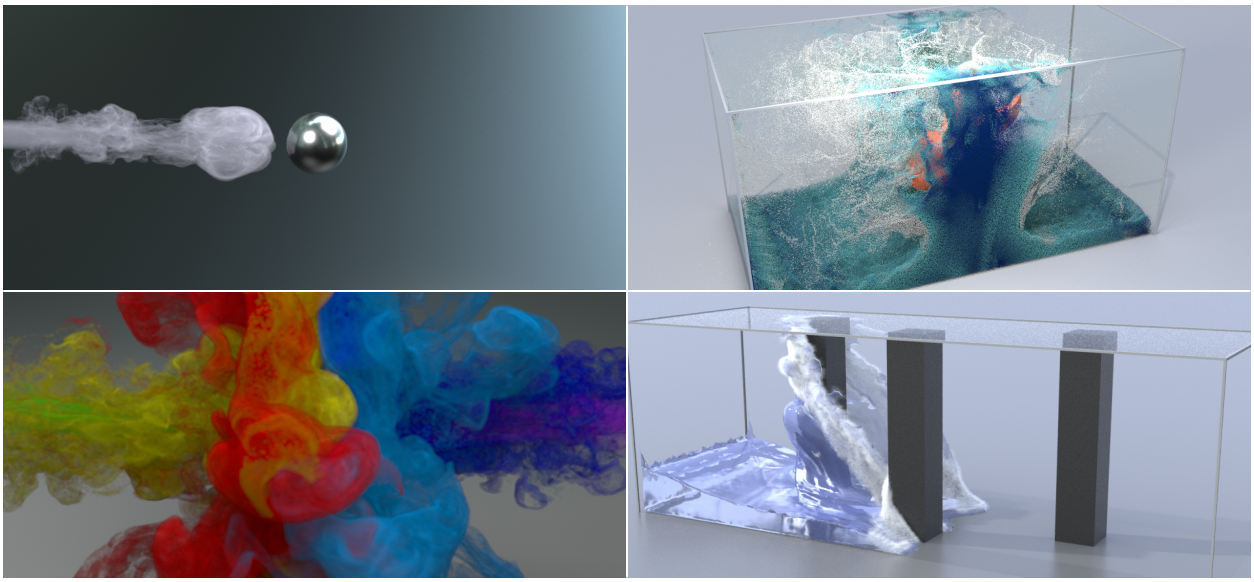


Figure 2.1: **Banner Image.** We simulate detailed incompressible flows with free surfaces and irregular domains using our novel hybrid particle/grid simulation approach. Our numerical method yields intricate flow details with little dissipation, even at modest spatial resolution. Furthermore, we use collocated velocity grids rather than staggered MAC grids.

2.1 Introduction

Incompressible flow simulation is an integral part of modern computer graphics toolkits. Chorin splitting of the advective and pressure projection terms in the incompressible equations has been standard in graphics since the works of Foster and Metaxas [FM96], Stam [Sta99], and Fedkiw et al. [FSJ01, FF01]. Most such methods employ regular Marker-And-Cell (MAC) grids with pressure degrees of freedom at cell centers and velocity degrees of freedom at cell faces. This staggered pressure/velocity grid has the advantages of preventing pressure null modes and allow for straightforward second order central differences when discretizing the gradient and divergence operators. However, MAC grids can add undesirable complexity to the algorithms which employ them as the pressure and each component of velocity exist on essentially distinct regular grids.

The advection step of the splitting is typically handled with semi-Lagrangian (SL) techniques originating from the atmospheric and oceanic sciences literature [Rob81]. One of the major advantages of SL is that it allows for large time steps, which can reduce overall computation cost. The trade-off for this increased stability is often significant dissipation, which can be quite detrimental for graphics applications. Additionally, when combined with MAC grids, the staggered locations of the velocity components means that SL must compute an upwind location for each of these staggered locations, as opposed to a single upwind location for collocated velocity grids.

Other non-MAC grid approaches with collocated velocities exist in the literature, including mixed Finite Element Methods (FEM) [Hug12]. One example is an FEM method using Taylor-Hood elements [TH73]. These elements have also been combined with B-spline interpolation [Bre10]. We build on this approach with a method using multi-quadratic and multi-linear B-spline interpolation for velocity and pressure degrees of freedom, respectively. However, while our velocity components are collocated, the velocities and pressure are on cell centers and cell nodes, respectively, as in [NSB18]. We use regular grids for velocity and

pressure but allow for non-voxelized, irregular domains with a variational approach with cut cells. This is similar to XFEM [BGV09, KBT17] and virtual node techniques [SSH14], where integrals in the variational formulation are performed over the intersection of the grid with the physical domain.

For the advection step, we introduce a novel backward semi-Lagrangian (BSL) approach using quadratic B-splines, which we call BSLQB, which is second order in space and time. Both SL and BSL methods are based on the implicit relation

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{u}(\mathbf{x} - (t - s)\mathbf{u}(\mathbf{x}, t), s) \tag{2.1}$$

for solutions of the inviscid Burgers' Equation [Eva10]; here $s \leq t$. Semi-Lagrangian advection performs the velocity update via

$$\mathbf{u}_\mathbf{i}^{n+1} = \mathbf{u}(\mathbf{x}_\mathbf{i} - \Delta t \mathbf{u}_\mathbf{i}^n, t^n), \tag{2.2}$$

where $\mathbf{x}_\mathbf{i}$ is the location of grid node $\mathbf{i}$, $\mathbf{u}_\mathbf{i}^n$ and $\mathbf{u}_\mathbf{i}^{n+1}$ are the velocities at times t^n and t^{n+1}, respectively, and $\mathbf{u}(\mathbf{x}_\mathbf{i} - \Delta t \mathbf{u}_\mathbf{i}^n, t^n)$ is estimated at non-grid locations using interpolation. In contrast, BSL solves the implicit equation

$$\mathbf{u}_\mathbf{i}^{n+1} = \mathbf{u}(\mathbf{x}_\mathbf{i} - \Delta t \mathbf{u}_\mathbf{i}^{n+1}, t^n), \tag{2.3}$$

which is more directly analogous to Equation (2.1). This approach is at least as stable as SL, likewise allowing for larger than CFL time steps.

Finally, we develop a hybrid particle/BSLQB advection method which uses PolyPIC [FGG17] in the parts of the computational domain containing particles and BSLQB in the parts which do not. As areas sparsely populated by particles develop from turbulent flows, BSLQB takes over to maintain high quality results.

Our contributions are summarized below:

- A novel cut-cell collocated velocity B-spline mixed FEM method for Chorin [Cho67] splitting discretization of the incompressible Euler equations.

- BSLQB: a novel BSL technique designed for collocated multiquadratic B-spline velocity interpolation that achieves second order accuracy in space and time for the advection step.
- A hybrid BSLQB/PolyPIC method for narrow band free-surface flow simulations and concentrated-detail smoke simulations.

2.2 Previous work

2.2.1 Advection

Stam [Sta99] first demonstrated the efficacy of semi-Lagrangian techniques for graphics applications and they have since become the standard, largely due to the large time steps they engender and their simple interpolatory nature. Many modifications to the original approach of Stam have been developed, often inspired by approaches in the engineering literature. Fedkiw et al. [FSJ01] use vorticity confinement [SU94] to counterbalance vorticity lost to dissipation and cubic grid interpolation. Kim et al. [KLL06, KLL05] and Selle et al. [SFK08] combine forward and backward semi-Lagrangian steps to estimate and remove dissipative errors. Constrained Interpolation Profile [KSK08, YXU01, SKK09] techniques additionally advect function derivatives to reduce dissipation. Molemaker et al. [MCP08] use the QUICK technique of Leonhard [Leo79] which is essentially upwinding with quadratic interpolation and Adams-Bashforth temporal discretization, although this does not have the favorable stability properties of semi-Lagrangian. Backward Difference Formula techniques are useful because they use an implicit multistep formulation for higher-order semi-Lagrangian advection yet still only require one projection per time step [XK01, SSH14].

The main idea in semi-Lagrangian techniques is to interpolate data from a characteristic point. This idea goes back to the Courant-Isaacson-Rees [CIR52] method. However, as noted in [FSJ01], semi-Lagrangian advection is very popular in atmospheric science simulation and

the variants used in graphics that account for characteristics traveling beyond the local cell in one time step go back to Sawyer [Saw63]. The first BSL approach utilizing Equation (2.3) was done by Robert [Rob81] in which they use fixed point iteration to solve the nonlinear equation. They fit a bi-cubic function to their data over 4×4 grid patches, then use that function in the fixed point iteration. If the upwind point leaves the grid, they clamp it to the boundary of the 4×4 patch. This clamping will degrade accuracy for larger time steps. In this case, more general interpolation is typically used (see [SC91, FF98] for useful reviews). Pudykiewicz and Staniforth [PS84] investigate the effects of BSL versus explicit semi-Lagrangian. Specifically, they compare Bates and McDonald [BM82] (explicit) versus Robert [Rob81] (BSL). They show that keeping all things equal, the choice of Equation (2.2) (explicit) instead of Equation (2.3) (BSL) leads to more dissipation and mass loss. This is consistent with our observations with BSLQB.

Interestingly, multiquadratic B-splines have not been adopted by the semi-Lagrangian community, despite their natural regularity. Hermite splines, multicubic splines and even Lagrange polynomials are commonly used [SC91]. Preference for Hermite splines and Lagrange polynomials is likely due to their local nature (they do not require solution of a global system for coefficients) and preference for multicubic splines (over multi-quadratic) is possibly due to the requirement of odd degree for natural splines (odd degree splines behave like low pass filters and tend to be smoother than even degree splines [CWB01, CK12]). Cubic splines are considered to be more accurate than Hermite splines and Lagrange interpolation [SC91, MK96]. Interestingly, Riishøjgaard et al. [RCL98] found that cubic spline interpolation gave rise to a noisier solution than cubic Lagrange interpolation with a technique analogous to that of Makar and Karpik [MK96]. However, they also note that addition of a selective scale diffusion term helps reduce noise associated with cubic splines. Wang and Layton [WL10] use linear B-splines with BSL but only consider one space dimension which makes Equation (2.3) linear and easily solvable.

Dissipation with explicit semi-Lagrangian advection is so severe that many graphics re-

searchers have resorted to alternative methods to avoid it. Mullen et al. [MCP09] develop energy preserving integration to prevent the need for correcting dissipative behavior. Some authors [QZG19, TP11, SIB17, SBI18] resolve the flow map characteristics for periods longer than a single time step (as opposed to one step with semi-Lagrangian) to reduce dissipation. Hybrid Lagrange/Eulerian techniques like PIC (and related approaches) [Bri08, JSS15, FGG17, ZB05] explicitly track motion of particles in the fluid, which is nearly dissipation-free, but can suffer from distortion in particle sampling quality. Vorticity formulations are also typically less dissipative, but can have issues with boundary conditions enforcement [SRF05, AN05, CKP16, STK07, PK05, WP10]. Zehnder et al., Zhang et al. and Mullen et al. [MCP09, ZNT18, NZT19, ZBG15] have noted that the Chorin projection itself causes dissipation. Zhang et al. [ZBG15] reduced artificial dissipation caused by the projection step by estimating lost vorticity and adding it back into the fluid. Zehnder et al. [ZNT18, NZT19] propose a simple, but very effective modification to the splitting scheme that is similar to midpoint rule integration to reduce the projection error.

2.2.2 Pressure projection

Graphics techniques utilizing pressure projection typically use voxelized MAC grids with boundary conditions enforced at cell centers and faces. However, many methods improve this by taking into account sub-cell geometric detail. Enright et al. [ENG03] showed that enforcing the pressure free surface boundary condition at MAC grid edge crossings (rather than at cell centers) dramatically improved the look of water surface waves and ripples. Batty, Bridson, and colleagues developed variational weighted finite difference approaches to enforce velocity boundary conditions with MAC grids on edge crossings and improved pressure boundary conditions at the free surface in the case of viscous stress [BBB07, BB08, LBB17]. XFEM [BGV09, KBT17] and virtual node (VNA) [SSH14] techniques also use cut cell geometry with variational techniques. Schroeder et al. [SSH14] use cut cells with MAC grids, but their technique is limited to moderate Reynolds numbers. Recently, Nielsen et

al. [NSB18] have shown that collocated velocities with only staggered pressures can be used effectively for projection with turbulent detailed flow simulations.

There is a vast literature on enforcing incompressibility in the FEM community [Hug12]. Our approach is most similar to the B-spline Taylor-Hood element of Bressan [Bre10]. Adoption of B-spline interpolation in FEM is part of the isogeometric movement [HCB05, RC12]. Originally motivated by the desire to streamline the transition from computer-aided design (CAD) to FEM simulation, isogeometric analysis explores the use of CAD-based interpolation (e.g. B-splines and nonuniform rational B-splines (NURBS)) with FEM methodologies. Hughes et al. [HCB05] show that in addition to simplifying the transition from CAD to simulation, the higher regularity and spectral-like properties exhibited by these splines makes them more accurate than traditionally used interpolation. We enforce Dirichlet boundary conditions weakly as in XFEM and VNA approaches [BGV09, KBT17, SSH14]. Bazilevs et al. [BH07] show that weak Dirichlet enforcement with isogeometric analysis can be more accurate than strong enforcement. Edwards and Bridson [EB14] use a discontinuous Galerkin FEM approach to simulate free surface flows over adaptive grids. Ferstl et al. [FWD14] also use an FEM based approach for discretization of pressure projections over adaptive hexahedral grids. Schneider et al. [SDG19] recently developed a third order accurate FEM approach for solving Poisson and other problems on predominantly hexahedral meshes.

Graphics applications are typically concerned with turbulent, high-Reynolds numbers flows. Interestingly, B-splines have proven effective for these flows by researchers in the Large Eddy Simulation (LES) community [Kim98, KMS99]. Kravchenko et al. [KMS99] use a variational weighted residuals approach with B-splines for turbulent LES and show that the increased regularity significantly reduces computational costs. Botella [Bot02] use a similar approach, but apply a collocation technique where the strong form of the div-grad formulation of incompressibility is enforced pointwise. They show that their B-spline approach attains optimal order of accuracy with accurate resolution of quadratic flow invariants. Botella [Bot02] also introduce a notion of sparse approximation to the inverse mass

matrix to avoid dense systems of equations in the pressure solve.

2.3 Governing Equations and Operator Splitting

We solve the incompressible Euler equations that describe the evolution of a fluid in terms of its mass density ρ, velocity $\mathbf{u}$, pressure p and gravitational constant $\mathbf{g}$ as

$$\rho \frac{D\mathbf{u}}{Dt} = \rho \left(\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \mathbf{u} \right) = -\nabla p + \rho \mathbf{g}, \ \mathbf{x} \in \Omega \tag{2.4}$$

$$\nabla \cdot \mathbf{u} = 0, \ \mathbf{x} \in \Omega \tag{2.5}$$

$$\mathbf{u} \cdot \mathbf{n} = a, \ \mathbf{x} \in \partial\Omega_S \tag{2.6}$$

$$p = 0, \ \mathbf{x} \in \partial\Omega_{FS} \tag{2.7}$$

where Equation (2.4) is balance of linear momentum, Equation (2.5) is the incompressibility constraint, Equation (2.6) is the boundary condition for the normal component of the velocity, and Equation (2.7) is the free surface boundary condition. We use Ω to denote the region occupied by the fluid, $\partial\Omega_S$ to denote the portion of the boundary of the fluid domain on which velocity is prescribed to be a (which may vary over the boundary) and $\partial\Omega_{FS}$ is the surface of the water where the pressure is zero (see Figure 2.2).

In a Chorin [Cho67] operator splitting of the advective and pressure terms, velocity is first updated to an intermediate field $\mathbf{w}$ under the convective term $\rho \frac{D\mathbf{u}}{Dt} = \mathbf{0}$, followed by an update from the pressure and gravitational body forcing under $\rho \frac{\partial \mathbf{u}}{\partial t} = -\nabla p + \rho \mathbf{g}$, where the pressure is determined to enforce $\nabla \cdot \mathbf{u} = 0$. Dividing by the mass density, the convective step is seen to be an update under Burgers' equation

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \mathbf{u} = \mathbf{0}. \tag{2.8}$$

Burgers' equation governs temporally constant Lagrangian velocity (zero Lagrangian acceleration). The characteristic curves for flows of this type are straight lines (since the Lagrangian acceleration is zero), on which the velocity is constant (see Figure 2.3). This gives rise to the

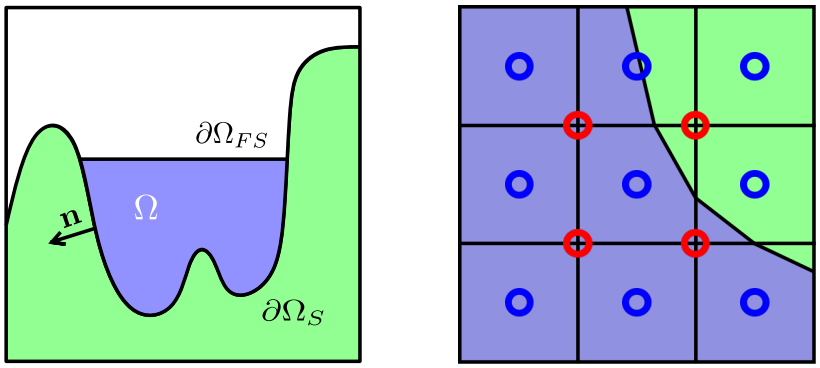


Figure 2.2: **Flow Domain and Grid. Left:** we use Ω to denote the fluid domain, with $\partial\Omega_S$ used to indicate the portion of the fluid domain subject to velocity boundary conditions and $\partial\Omega_{FS}$ to indicate the free-surface portion of the boundary with pressure condition $p = 0$. **Right:** We use multiquadratic interpolation for velocity ($\bar{\mathbf{u}}_\mathbf{i}$ at cell centers, blue) and multilinear for pressure ($p_\mathbf{c}$ at nodes, red). The fluid domain is defined with sub-grid-cell accuracy.

implicit relation (2.1). Intuitively, if we want to know the velocity $\mathbf{u}(\mathbf{x}, t)$ at point $\mathbf{x}$ at time t, we look back along the characteristic passing through $\mathbf{x}$ at time t to any previous time s; however, the characteristic is the straight line defined by the velocity $\mathbf{u}(\mathbf{x}, t)$ that we want to know. Hence, we take an implicit approach to the solution of this equation, which when combined with the operator splitting amounts to

$$\frac{\mathbf{w} - \tilde{\mathbf{u}}^n}{\Delta t} = \mathbf{0} \tag{2.9}$$

$$\rho\frac{\mathbf{u}^{n+1} - \mathbf{w}}{\Delta t} = -\nabla p^{n+1} + \rho\mathbf{g} \tag{2.10}$$

$$\nabla \cdot \mathbf{u}^{n+1} = \mathbf{0}, \tag{2.11}$$

where we use the notation $\mathbf{u}^{n+\alpha}(\mathbf{x}) = \mathbf{u}(\mathbf{x}, t^{n+\alpha})$, $\alpha = 0, 1$, to denote the time $t^{n+\alpha}$ velocities. Furthermore, the intermediate velocity $\mathbf{w}$ is related to $\tilde{\mathbf{u}}^n$ through $\tilde{\mathbf{u}}^n(\mathbf{x}) = \mathbf{u}(\mathbf{x} - \Delta t\mathbf{w}(\mathbf{x}), t^n)$.

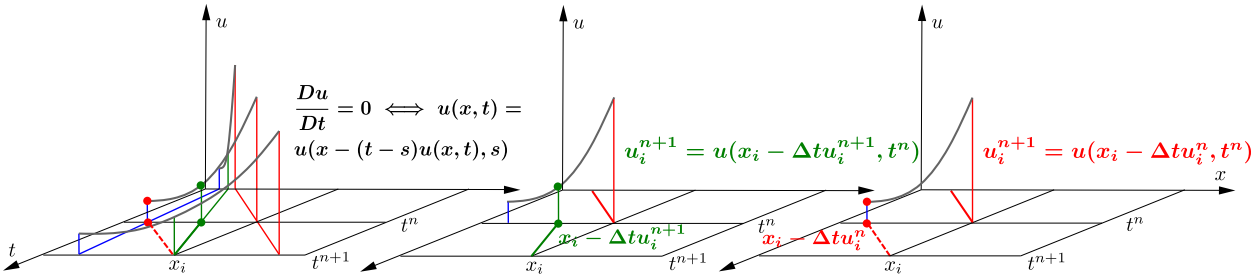


Figure 2.3: **BSL vs. SL.** We illustrate the difference between explicit semi-Lagrangian and BSL in 1D. **Left:** The exact solution of Burgers' equation has straight line characteristics shown in blue, green and red on which velocity (plotted above the plane in gray) is constant. **Middle:** BSL (green) uses Newton's method to solve for the exact characteristic going through x_i at time t^{n+1} to determine u_i^{n+1}. **Right:** explicit semi-Lagrangian (red) uses a stale, time t^n approximation of the characteristic which overshoots, resulting in an underestimate of the velocity and energy loss.

2.4 Spatial Discretization

We discretize in space by first representing velocity and pressure in terms of multiquadratic and multilinear B-splines for velocity and pressure, respectively. We use a regular grid with spacing Δx and define pressure degrees of freedom at grid vertices and velocity degrees of freedom at grid cell centers as in [ATW13] (see Figure 2.2). This efficiently aligns the support of the multiquadratic and multilinear interpolating functions which naturally allows for a grid-cell-wise definition of the flow domain (see Figure 2.11). We use $N_{\mathbf{i}}(\mathbf{x})$ to represent the multiquadratic B-spline basis function associated with velocity degree of freedom $\bar{\mathbf{u}}_{\mathbf{i}}$ at grid cell center $\mathbf{x_i}$ and $\chi_{\mathbf{c}}(\mathbf{x})$ for the multilinear basis function associated with pressure $p_{\mathbf{c}}$ at grid node $\mathbf{x_c}$. These are defined as

$$N_{\mathbf{i}}(\mathbf{x}) = \prod_{\alpha} \hat{N}\left(\frac{x_\alpha - x_{\alpha \mathbf{i}}}{\Delta x}\right), \ \chi_{\mathbf{c}}(\mathbf{x}) = \prod_{\alpha} \hat{\chi}\left(\frac{x_\alpha - x_{\alpha \mathbf{c}}}{\Delta x}\right) \tag{2.12}$$

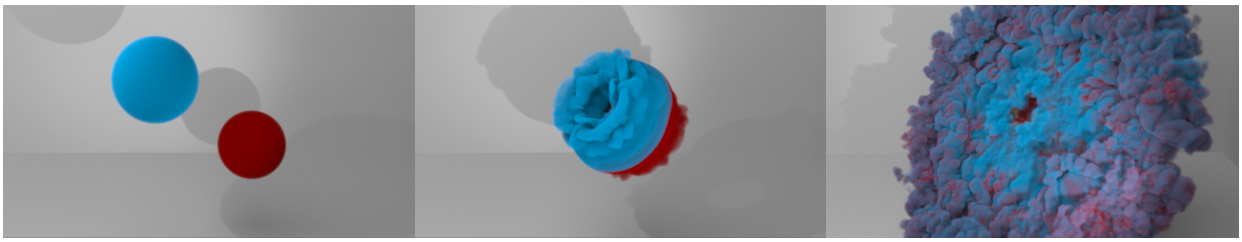


Figure 2.4: **High-Resolution Smoke.** Two spheres of smoke collide in a high-resolution 3D simulation ($\Delta x = 1/255$). BSLQB accurately resolves vorticial flow detail.

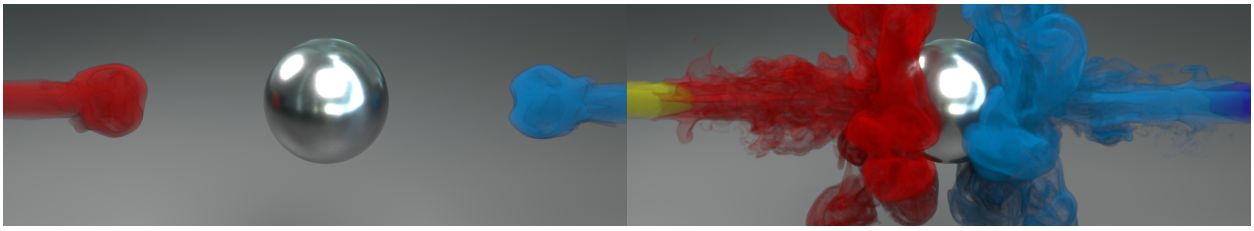


Figure 2.5: **Colorful Smoke Jets.** Multicolored jets of smoke are simulated with BSLQB. Intricate mixing is induced as the flows collide at the spherical boundary.

$$\hat{N}(\eta) = \begin{cases} \frac{1}{2}\left(\frac{3}{2} - |\eta|\right)^2, & |\eta| \in (\frac{1}{2}, \frac{3}{2}) \\ -\eta^2 + \frac{3}{4}, & |\eta| \in [0, \frac{1}{2}] \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

$$\hat{\chi}(\nu) = \begin{cases} 1 - |\nu|, & |\nu| \in (0, 1) \\ 0, & \text{otherwise} \end{cases} \tag{2.14}$$

where we use Greek indices α to indicate components of the vectors $\mathbf{x}$, $\mathbf{x_i}$, and $\mathbf{x_c}$. With this convention we interpolate to define the velocity and pressure fields:

$$\mathbf{u}(\mathbf{x}) = \sum_{\mathbf{i}} \bar{\mathbf{u}}_{\mathbf{i}} N_{\mathbf{i}}(\mathbf{x}), \; p(\mathbf{x}) = \sum_{\mathbf{c}} p_{\mathbf{c}} \chi_{\mathbf{c}}(\mathbf{x}). \tag{2.15}$$

We use the notation $\bar{\mathbf{u}}_{\mathbf{i}}$ to distinguish it from the grid node velocity $\mathbf{u}(\mathbf{x_i}) = \sum_{\mathbf{j}} \bar{\mathbf{u}}_{\mathbf{j}} N_{\mathbf{j}}(\mathbf{x_i})$ since the multiquadratic B-splines are not interpolatory and these will in general be different. Note that multilinear interpolation is interpolatory and $p_{\mathbf{c}} = \sum_{\mathbf{d}} p_{\mathbf{d}} \chi_{\mathbf{d}}(\mathbf{x_c})$.

2.4.1 BSLQB Advection

With this interpolation choice, we first solve for intermediate grid node velocity values $\mathbf{w}(\mathbf{x_i})$ from Equation (2.9) as

$$\mathbf{w}(\mathbf{x_i}) = \sum_{\mathbf{j}} \bar{\mathbf{u}}_{\mathbf{j}}^n N_{\mathbf{j}} \left(\mathbf{x_i} - \Delta t \mathbf{w}(\mathbf{x_i})\right). \tag{2.16}$$

We can solve this equation using Newton's method since the multiquadratic B-splines are C^1. We use $\mathbf{w}_{\mathbf{i}}^k$ to denote the k^{th} Newton approximation to $\mathbf{w}(\mathbf{x_i})$. Explicit semi-Lagrangian is used as an initial guess with $\mathbf{w}_{\mathbf{i}}^0 = \sum_{\mathbf{j}} \bar{\mathbf{u}}_{\mathbf{j}}^n N_{\mathbf{j}} \left(\mathbf{x_i} - \Delta t \sum_{\mathbf{l}} \bar{\mathbf{u}}_{\mathbf{l}}^n N_{\mathbf{l}}(\mathbf{x_i})\right)$ and then we update iteratively via $\mathbf{w}_{\mathbf{i}}^k \mathrel{+}= \boldsymbol{\delta}\mathbf{u}^k$ with Newton increment $\boldsymbol{\delta}\mathbf{u}^k$ satisfying

$$\boldsymbol{\delta}\mathbf{u}^k = \left(\mathbf{I} + \Delta t \frac{\partial \mathbf{u}^n}{\partial \mathbf{x}}\left(\mathbf{x_i} - \Delta t \mathbf{w}_{\mathbf{i}}^k\right)\right)^{-1} \left(\sum_{\mathbf{j}} \bar{\mathbf{u}}_{\mathbf{j}}^n N_{\mathbf{j}}\left(\mathbf{x_i} - \Delta t \mathbf{w}_{\mathbf{i}}^k\right) - \mathbf{w}_{\mathbf{i}}^k\right), \tag{2.17}$$

where $\frac{\partial \mathbf{u}^n}{\partial \mathbf{x}}\left(\mathbf{x_i} - \Delta t \mathbf{w}_{\mathbf{i}}^k\right) = \sum_{\mathbf{j}} \bar{\mathbf{u}}_{\mathbf{j}}^n \frac{\partial N_{\mathbf{j}}}{\partial \mathbf{x}}\left(\mathbf{x_i} - \Delta t \mathbf{w}_{\mathbf{i}}^k\right)$. It is generally observed [KW90, PS84] that with BSL approaches of this type, this iteration will converge as long as

$$\mathbf{I} + \Delta t \sum_{\mathbf{j}} \bar{\mathbf{u}}_{\mathbf{j}}^n \frac{\partial N_{\mathbf{j}}}{\partial \mathbf{x}}\left(\mathbf{x_i} - \Delta t \mathbf{w}_{\mathbf{i}}^k\right) \tag{2.18}$$

is non-singular. We note that this condition holds as long as no shocks form under Burgers' equation [Eva10] (forward from time t^n). This is a safe assumption since we are modeling incompressible flow with which shock formation does not occur, but it may be a problem for compressible flows. In practice, this iteration converges in 3 or 4 iterations, even with CFL numbers larger than 4 (see Section 2.7.1). When it does fail (which occurs less than one percent of the time in the examples we run), it is usually for points near the boundary with characteristics that leave the domain (since we cannot estimate $\frac{\partial \mathbf{u}^n}{\partial \mathbf{x}}$ using grid interpolation if the upwind estimate leaves the grid). In this case we use explicit semi-Lagrangian and interpolate from the boundary conditions if the characteristic point is off the domain.

Once we have obtained the grid node values of the intermediate velocity $\mathbf{w}(\mathbf{x_i})$, we must determine interpolation coefficients $\bar{\mathbf{w}}_{\mathbf{j}}$ such that $\mathbf{w}(\mathbf{x_i}) = \sum_{\mathbf{j}} \bar{\mathbf{w}}_{\mathbf{j}} N_{\mathbf{j}}(\mathbf{x_i})$. On the boundary

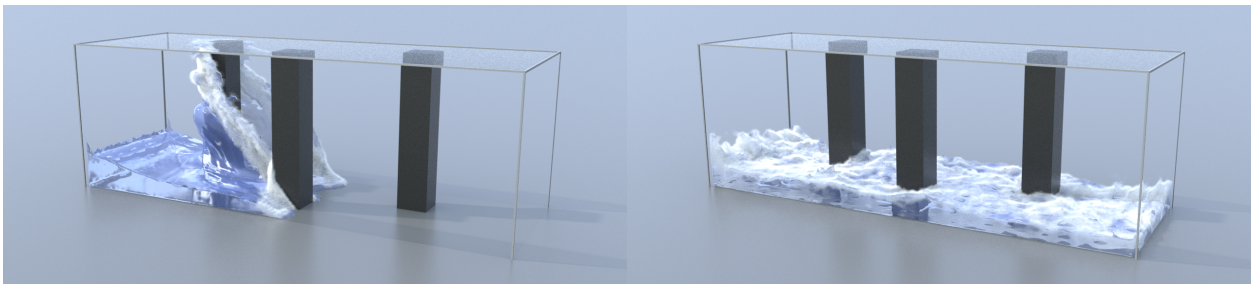


Figure 2.6: **Dam break.** A block of water falls in a rectangular domain with obstacles. Dynamic splashing behavior is followed by settling of the water in the tank. White water rendering effects are added based on [IAA12].

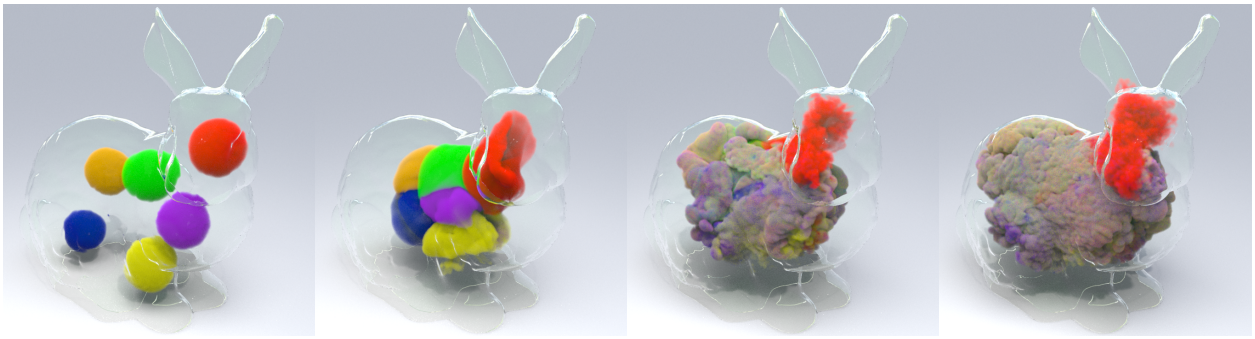


Figure 2.7: **Smoke in an Irregular Domain.** Multicolored spheres of smoke with non-zero initial velocity conditions flow and collide inside the Stanford bunny. Zero normal velocity is enforced with our cut cell formulation.

of the grid, we set $\bar{\mathbf{w}}_\mathbf{j} = \mathbf{w}(\mathbf{x}_\mathbf{j})$ since we can only interpolate to $\mathbf{x}_\mathbf{i}$ if all of its neighbors have data. This yields a square, symmetric positive definite system of equations for the remaining $\bar{\mathbf{w}}_\mathbf{j}$. The system is very well conditioned with sparse, symmetric matrix $N_\mathbf{j}(\mathbf{x}_\mathbf{i})$ consisting of non-negative entries and rows that sum to one. The sparsity and symmetry of the system arises from the compact support and geometric symmetry, respectively, of the B-spline basis functions $N_\mathbf{j}$. In practice, the system can be solved to machine precision in tens of unpreconditioned CG iterations. We have noticed that for some flows, determining the coefficients $\bar{\mathbf{w}}_\mathbf{j}$ can lead to increasingly oscillatory velocity fields. This is perhaps due to the unfavorable filtering properties of even order B-splines [CWB01, CK12]. However, we

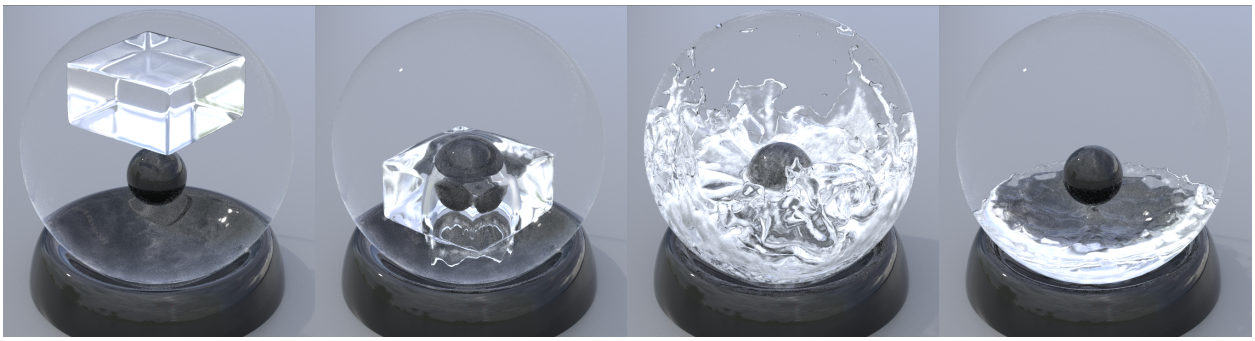


Figure 2.8: **Water in a Globe.** A block of water splashes and naturally slides along cut cell boundaries in an irregular domain interior to one large sphere and exterior to one small sphere.

found that a simple stabilization strategy can be obtained as

$$\sum_{\mathbf{j}} \left(\lambda N_{\mathbf{j}}(\mathbf{x_i}) + (1-\lambda)\delta_{\mathbf{ij}}\right) \bar{\mathbf{w}}_{\mathbf{j}} = \mathbf{w}(\mathbf{x_i}), \tag{2.19}$$

where $\lambda \in [0, 1]$ and $\delta_{\mathbf{ij}}$ is the Kronecker delta. A value of $\lambda = 0$ is very stable, but extremely dissipative. Stable yet energetic behavior is achieved by decreasing the value of λ under grid refinement. In practice we found that $\lambda \in (.95, 1]$ with $\lambda = c\Delta x$ for constant c provided a good balance without compromising second order accuracy of the method (see Section 2.7.1). We note that Riishøjgaard et al. [RCL98] also added diffusion to cubic spline interpolation based semi-Lagrangian to reduce noise.

2.4.2 Hybrid BSLQB-PolyPIC Advection

In some portions of the domain, we store particles with positions $\mathbf{x}_p^n$ and PolyPIC [FGG17] velocity coefficients $\mathbf{c}_p^n$. In the vicinity of the particles, we use PolyPIC [FGG17] to update the intermediate velocity field $\bar{\mathbf{w}}_{\mathbf{j}}$. First we update particle positions as $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^n$ (where the velocity $\mathbf{v}_p^n$ is determined from $\mathbf{c}_p^n$ following [FGG17]). Then the components $\bar{w}_{\mathbf{j}\alpha}$

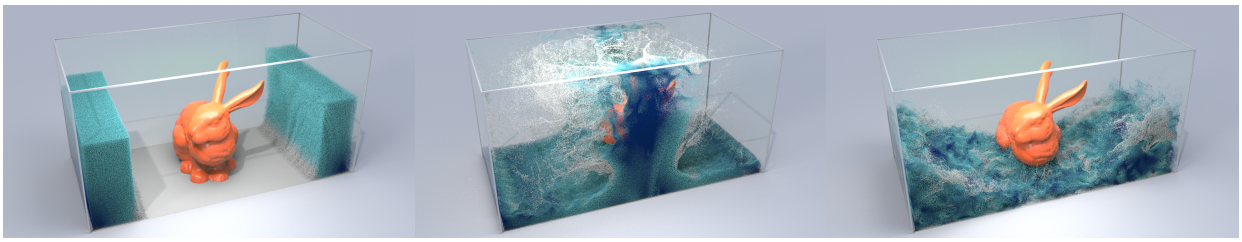


Figure 2.9: **Dam Break with Bunny.** Opposing blocks of water collapse in a tank and flow around the irregular domain boundary placed in the middle of the tank. Particles are colored from slow (blue) to fast (white) speed.

of the coefficients $\bar{\mathbf{w}}_{\mathbf{j}}$ are determined as

$$\bar{w}_{\mathbf{j}\alpha} = \frac{\sum_p m_p N_{\mathbf{j}}(\mathbf{x}_p^{n+1}) \left(\sum_{r=1}^{N_r} s_r(\mathbf{x}_{\mathbf{j}} - \mathbf{x}_p^{n+1}) c_{pr\alpha}^n \right)}{\sum_p m_p N_{\mathbf{j}}(\mathbf{x}_p^{n+1})}, \tag{2.20}$$

where N_r is the number of polynomial modes $s_r(\mathbf{x})$, as in Fu et al. [FGG17]. To create our hybrid approach, we update $\bar{w}_{\mathbf{j}\alpha}$ from Equation (2.20) whenever the denominator is greater than a threshold $\sum_p m_p N_{\mathbf{j}}(\mathbf{x}_p^{n+1}) > \tau^m$; Otherwise, we use the BSLQB update from Equation (2.19). We use this threshold because the grid node update in Equation(2.20) loses accuracy when the denominator is near zero and in this case the BSLQB approximation is likely more accurate. Note that the polynomial mode coefficients for the next time step $\mathbf{c}_p^{n+1}$ are determined from the grid velocities at the end of the time step (using particle positions $\mathbf{x}_p^{n+1}$ and after pressure projection).

2.5 Pressure Projection

We solve Equations (2.10)-(2.11) and boundary condition Equations (2.6)-(2.7) in a variational way. To do this, we require that the dot products of Equations (2.10), (2.11), and Equations (2.6) with arbitrary test functions $\mathbf{r}$, q, and μ, respectively, integrated over the domain are always equal to zero. The free surface boundary condition in Equation (2.7) is

naturally satisfied by our treatment of Equation (2.10). We summarize this as

$$\int_\Omega \mathbf{r}\cdot\rho\left(\frac{\mathbf{u}^{n+1}-\mathbf{w}}{\Delta t}\right)d\mathbf{x} = \int_\Omega p^{n+1}\nabla\cdot\mathbf{r}+\rho\mathbf{r}\cdot\mathbf{g}d\mathbf{x} - \int_{\partial\Omega} p^{n+1}\mathbf{r}\cdot\mathbf{n}ds(\mathbf{x}) \tag{2.21}$$

$$\int_\Omega q\nabla\cdot\mathbf{u}^{n+1}d\mathbf{x} = 0 \tag{2.22}$$

$$\int_{\partial\Omega_S}\mu\left(\mathbf{u}^{n+1}\cdot\mathbf{n}-a\right)ds(\mathbf{x}) = 0. \tag{2.23}$$

Here we integrate by parts in the integral associated with Equation (2.10). Furthermore, we modify the expression $\int_{\partial\Omega} p^{n+1}\mathbf{r}\cdot\mathbf{n}ds(\mathbf{x})$ in Equation (2.21) in accordance with the boundary conditions. We know that the pressure is zero on $\partial\Omega_{FS}$, however, we do not know its value on $\partial\Omega_S$. We introduce the pressure on this portion of the domain as a Lagrange multiplier λ^{n+1} associated with satisfaction of the velocity boundary condition in Equation (2.23). Physically, this is the external pressure we would need to apply on $\partial\Omega_S$ to ensure that $\mathbf{u}^{n+1}\cdot\mathbf{n} = a$. With this convention, we have $\int_{\partial\Omega} p^{n+1}\mathbf{r}\cdot\mathbf{n}ds(\mathbf{x}) = \int_{\partial\Omega_S}\lambda^{n+1}\mathbf{r}\cdot\mathbf{n}ds(\mathbf{x})$. We note that unlike Equation (2.23) (and its strong form counterpart (2.6)) which requires introduction of a Lagrange multiplier, Equation (2.7) is naturally enforced through the weak form simply by setting $p^{n+1}=0$ in the integral over $\partial\Omega_{FS}$ in Equation (2.21).

To discretize in space, we introduce interpolation for the test functions $\mathbf{r}$, q, and μ. We use the same spaces as in Equation (2.15) for velocity and pressure for $\mathbf{r} = \sum_{\mathbf{i}}\bar{\mathbf{r}}_{\mathbf{i}}N_{\mathbf{i}}$ and $q = \sum_{\mathbf{d}} q_{\mathbf{d}}\chi_{\mathbf{d}}$. For the test functions μ, we choose the same space as q, p, but with functions restricted to $\partial\Omega_S$: $\mu = \sum_{\mathbf{b}}\mu_{\mathbf{b}}\chi_{\mathbf{b}}$ for $\mathbf{b}$ with grid cell $\Omega_{\mathbf{b}}\cap\partial\Omega_S\neq\emptyset$ (see Figure 2.11). We choose the same space for $\lambda^{n+1} = \sum_{\mathbf{b}}\lambda_{\mathbf{b}}^{n+1}\chi_{\mathbf{b}}$ to close the system. We note that this choice of interpolating functions is necessary for preserving a standing pool since the pressure and λ interpolation functions need to cancel out to prevent artificial currents (see proof in [GHM20b]). With these choices for the test functions, the variational problem is projected to a finite dimensional problem defined by the interpolation degrees of freedom. This is expressed as a linear system for velocities $\bar{\mathbf{u}}_{\mathbf{j}}^{n+1}$, internal pressures $p_{\mathbf{c}}^{n+1}$, and external pressures

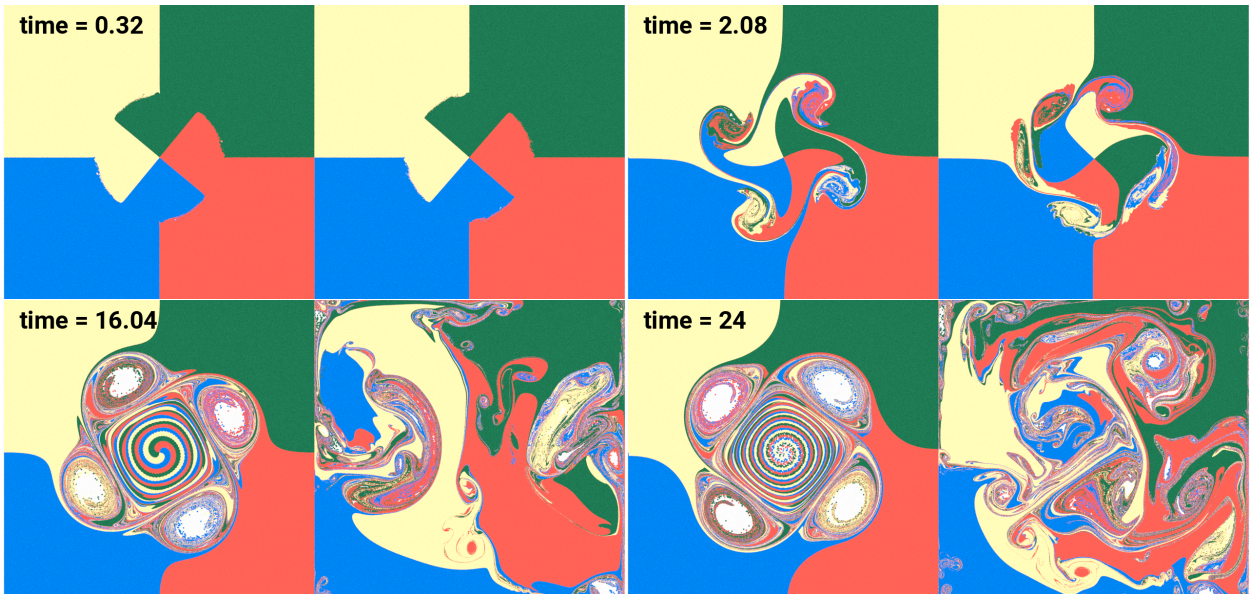


Figure 2.10: **SL vs. BSLQB.** We compare semi-Lagrangian (left) and BSLQB (right) in a vorticity-intensive example. BSLQB breaks symmetry and exhibits a more turbulent flow pattern. Note we only use particles for flow visualization and not for PolyPIC advection in this example.

$\lambda_{\mathbf{b}}^{n+1}$ which is equivalent to

$$\begin{pmatrix} \mathbf{M} & -\mathbf{D}^T & \mathbf{B}^T \\ -\mathbf{D} & & \\ \mathbf{B} & & \end{pmatrix} \begin{pmatrix} \mathbf{U}^{n+1} \\ \mathbf{P}^{n+1} \\ \mathbf{\Lambda}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{MW} + \hat{\mathbf{g}} \\ \mathbf{0} \\ \mathbf{A} \end{pmatrix}. \tag{2.24}$$

Here $\mathbf{U}^{n+1}$, $\mathbf{P}^{n+1}$, and $\mathbf{\Lambda}^{n+1}$ are the vectors of all unknown $\bar{\mathbf{u}}_{\mathbf{j}}^{n+1}$, $p_{\mathbf{c}}^{n+1}$, and $\lambda_{\mathbf{b}}^{n+1}$, respectively. Furthermore $\mathbf{M}$ is the mass matrix, $\mathbf{B}$ defines the velocity boundary conditions, and $\mathbf{D}$ defines the discrete divergence condition. Lastly, $\mathbf{W}$ is the vector of all $\bar{\mathbf{w}}_{\mathbf{i}}$ that define the intermediate velocity, $\hat{\mathbf{g}}$ is from gravity, and $\mathbf{A}$ is the variational boundary condition. Using the convention that Greek indices α, β range from 1–3, these matrices and vectors have

entries

$$M_{\alpha\mathbf{i}\beta\mathbf{j}} = \delta_{\alpha\beta} \int_\Omega \frac{\rho}{\Delta t} N_\mathbf{i} N_\mathbf{j} d\mathbf{x}, \tag{2.25}$$

$$D_{\mathbf{d}\beta\mathbf{j}} = \int_\Omega \chi_\mathbf{d} \frac{\partial N_\mathbf{j}}{\partial x_\beta} d\mathbf{x}, \;\; \hat{g}_{\alpha\mathbf{i}} = \int_\Omega \rho g_\alpha N_\mathbf{i} d\mathbf{x}, \tag{2.26}$$

$$B_{\mathbf{b}\beta\mathbf{j}} = \int_{\Omega_S} \chi_\mathbf{b} N_\mathbf{j} n_\beta ds(\mathbf{x}), \;\; A_\mathbf{b} = \int_\Omega a \chi_\mathbf{b} ds(\mathbf{x}). \tag{2.27}$$

If we define $\mathbf{G} = [-\mathbf{D}^T, \mathbf{B}^T]$, we can convert this system into a symmetric positive definite one for $\mathbf{P}^{n+1}$ and $\mathbf{\Lambda}^{n+1}$ followed by a velocity correction for $\mathbf{U}^{n+1}$:

$$\begin{pmatrix} \mathbf{P}^{n+1} \\ \mathbf{\Lambda}^{n+1} \end{pmatrix} = \left(\mathbf{G}^T \mathbf{M}^{-1} \mathbf{G}\right)^{-1} \left(\mathbf{G}^T \left(\mathbf{W} + \mathbf{M}^{-1} \hat{\mathbf{g}} \right) - \begin{pmatrix} \mathbf{0} \\ \mathbf{A} \end{pmatrix} \right) \tag{2.28}$$

$$\mathbf{U}^{n+1} = -\mathbf{M}^{-1} \mathbf{G} \begin{pmatrix} \mathbf{P}^{n+1} \\ \mathbf{\Lambda}^{n+1} \end{pmatrix} + \mathbf{W} + \mathbf{M}^{-1} \hat{\mathbf{g}}. \tag{2.29}$$

Unfortunately, this system will be dense in the current formulation since the full mass matrix $M_{\alpha\mathbf{i}\beta\mathbf{j}}$ is non-diagonal with dense inverse [Bot02]. However, a simple lumped mass approximation

$$M^l_{\alpha\mathbf{i}\beta\mathbf{j}} = \begin{cases} \delta_{\alpha\beta} \int_\Omega \frac{\rho}{\Delta t} N_\mathbf{i} d\mathbf{x}, & \mathbf{i} = \mathbf{j} \\ 0, & \text{otherwise} \end{cases} \tag{2.30}$$

gives rise to a sparse matrix in Equation (2.28).

2.5.1 Cut Cells

As in XFEM and VNA approaches [BGV09, KBT17, SSH14], we resolve sub-grid-cell geometry by simply performing the integrations in Equations (2.26)-(2.27) over the geometry of the fluid domain. We use a level set to define solid boundaries (green in Figure 2.11) on which velocity boundary conditions are defined. We triangulate the zero isocontour using marching cubes (MC) [Che95] (see Figure 2.13).

As noted in Section 1.2.2, the MC algorithm creates the zero isocontour in a cell-by-cell manner, with the triangulation in any given cell dependent on the level set values (and their

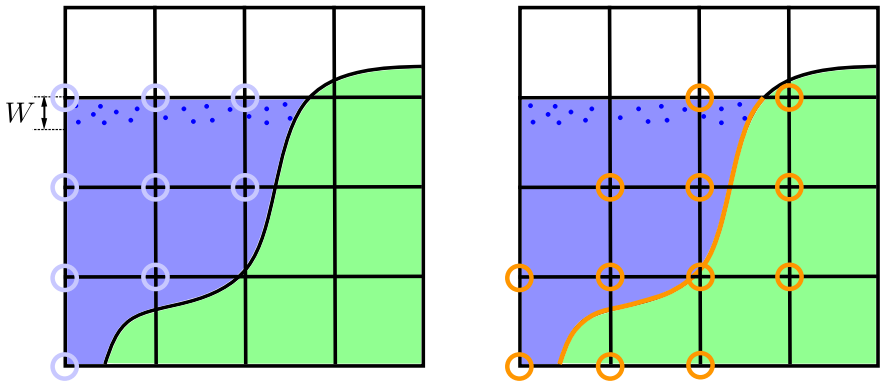


Figure 2.11: **Discrete Free Surface Fluid Domain. Left:** We define the fluid domain to consist of cells that either have (1) a particle (dark blue) in it or (2) a node with non-positive level set value (light blue). **Right:** Boundary Lagrange multiplier external pressure $\lambda_{\mathbf{b}}$ (orange circles) are like the interior pressures $p_{\mathbf{c}}$ except only defined on fluid domain cells that intersect $\partial\Omega_S$.

signs) at the 8 nodes of the cell. In most variants of the MC algorithm, the topology of a cell's triangulation is determined by the pattern of signs at the nodes, which leads to 256 total cases. Our particular implementation of MC is based on the approach of Bhaniramka et al. [BWC04], who create a full 256 case lookup table instead of exploiting various symmetries like other MC algorithms. They maintain topological consistency of the surface, in the sense that it is free from topological holes, by generating each case based on the following method. Sample points are placed at the midpoints of cell edges incident to both a positive and negative node. Then, they take the convex hull of the sample points together with negatively signed nodes. The convex hull is then tetrahedralized, and the boundary triangles which are not on cell-boundaries are extracted to form an entry in the lookup table. The actual isocontour is then obtained from the lookup table by replacing the sample points with their actual locations on the edges.

Since we need to integrate over both the surface and the volume, we extend this approach by also extracting the tetrahedrons for each case into a second 256 case lookup table.

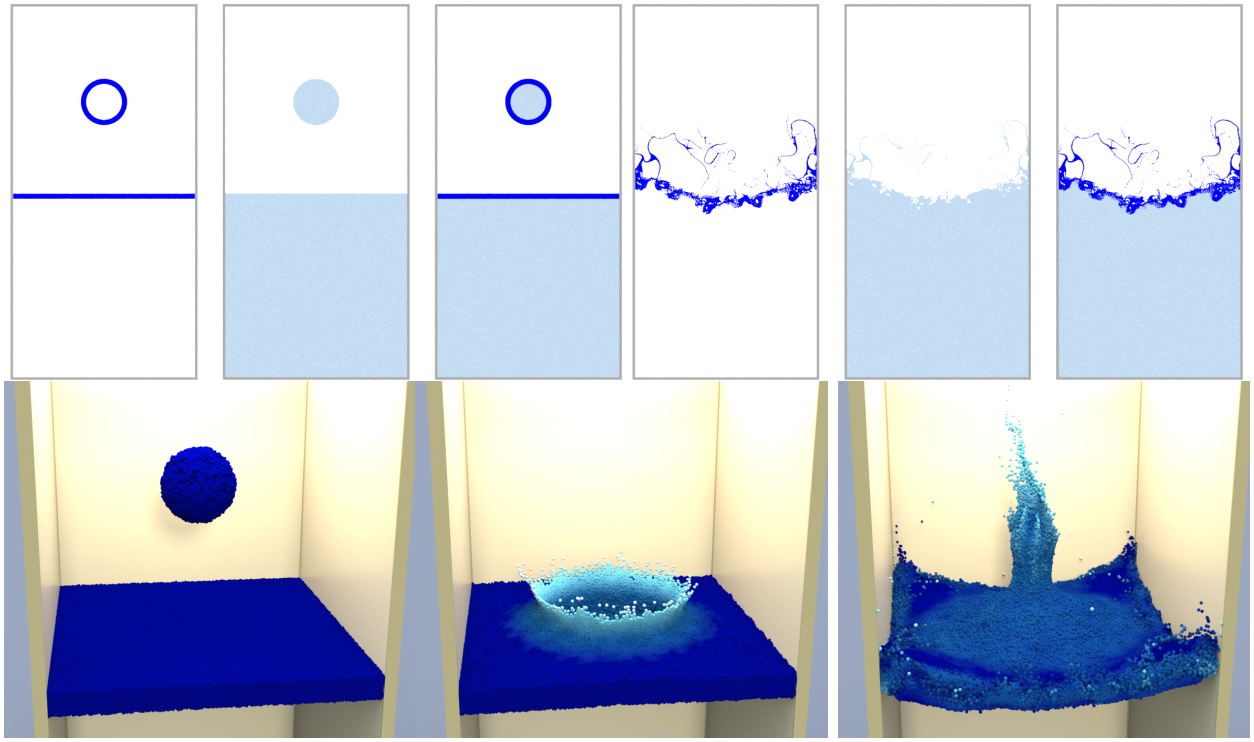


Figure 2.12: **Narrow Band Free Surface.** A circle/sphere falls in a tank of water under gravity. Using only a narrow band of particles saves computational cost and enables increased resolution of the free surface. **Top:** In 2D we illustrate the hybrid particle (dark blue)/level set (light blue) representation. **Bottom:** Particles are colored based on velocity magnitude.

Thus, the integrals in Equations (2.26)-(2.27) are over volumetric polyhedra (Equations (2.26), blue in Figure 2.13) or surface polygons (Equations (2.27), green in Figure 2.13). Furthermore, the integrands are all polynomials, so we use Gauss quadrature of sufficient order to compute these integrals with no error (see [GHM20b]). For free surface flows, we use particles (and additionally a level set function in the case of narrow banding, see Section (2.6)) to denote grid cells with fluid in them. Cells near the solid boundary are clipped by the marching cubes geometry, but otherwise the free surface boundary is voxelized. The fluid domain Ω is defined as the union of all clipped and full fluid cells (see Figure 2.11).

Notably, taking a cut cell approach with our variational formulation allows us to prove

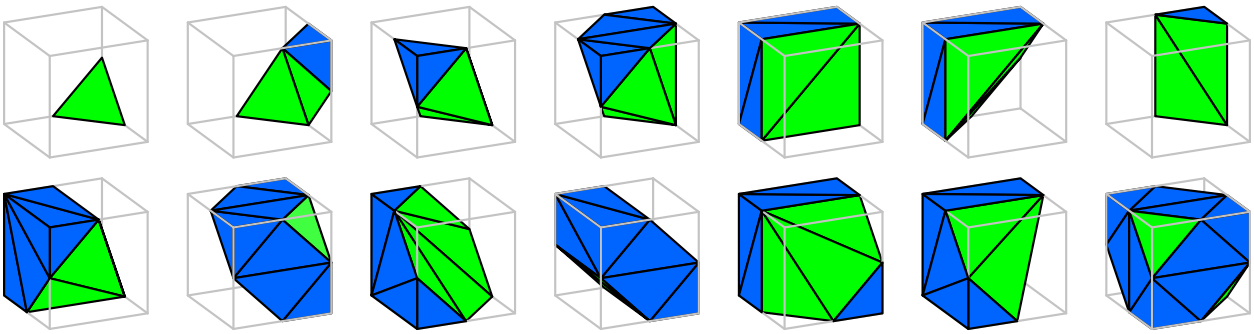


Figure 2.13: **Cut Cells.** We show the 14 essential cases used in determining the cut cell fluid domain geometry. Blue faces indicate the intersection of the grid cell with the fluid domain. Green faces indicate the velocity boundary condition faces on $\partial\Omega_S$.

that our method can resolve a standing pool of water exactly without producing numerical currents. We know that with gravitational force $\rho\mathbf{g}$ (e.g. with $\mathbf{g}$ pointing in the y direction with magnitude g), steady state is maintained if the pressure increases with depth as $p = \rho g\,(y_0 - y)$ where y_0 is the height of the water surface at rest, since $-\nabla p + \rho\mathbf{g} = \mathbf{0}$. Since we use multilinear interpolating functions for p, the exact solution is representable in our discrete space and with a short proof we show (see [GHM20b]) that this means our method will choose it to maintain a standing pool of water, independent of fluid domain boundary geometry.

2.6 Narrow band free surface

For free surface flows, we develop a narrow band approach as in [CMK15, FAW16, SWT18]. We represent the fluid domain with a level set and seed particles in a band of width W from the zero isocontour (see Figure 2.11). Particles are advected and used to augment BSLQB advection as detailed in Section 2.4.2. We also advect the level set by interpolating its value at the previous step from the upwind location $\mathbf{x_i} - \Delta t\mathbf{w}(\mathbf{x_i})$ determined in Equation (2.16). We then use the updated particle locations to compute a narrow band level set from the particles based on the method of Boyd and Bridson [BB12]. We update the level set to be

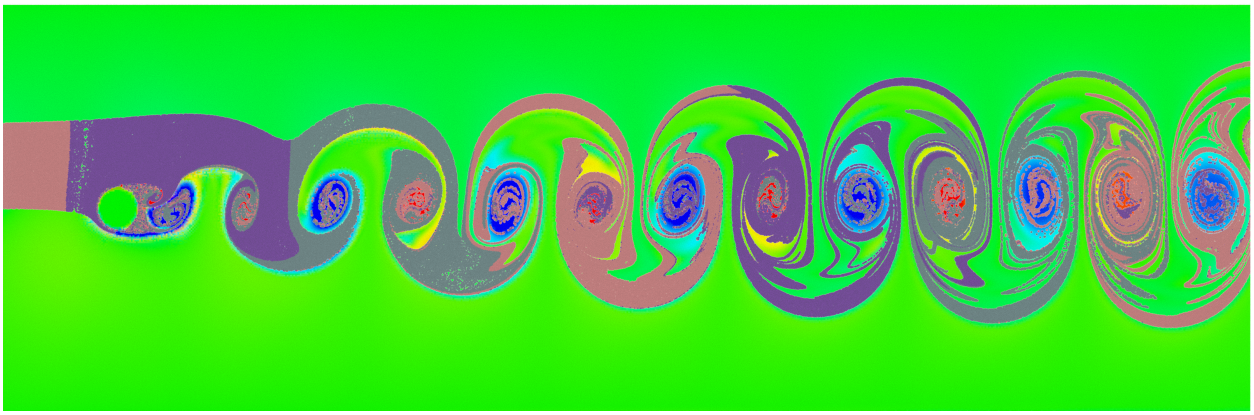


Figure 2.14: **Von Karman Vortex Shedding.** We demonstrate the accuracy of our Hybrid BSLQB/PolyPIC with vortex shedding past a disk in 2D. Note the smooth transition between regions with particles (PolyPIC) and those without (BSLQB).

the union of that defined by the narrow band and that from advection. This is done by taking the minimum of the two level set values and then redistancing with the method of Zhao [Zha05].

2.7 Examples

2.7.1 Hybrid BSLQB/PolyPIC

We demonstrate our hybrid BSLQB/PolyPIC advection with water simulation. We prevent excessive run times by utilizing a narrow band of particles near the free surface and a level set (with BSLQB advection) in deeper levels. Figure 2.12 top shows a disc of water splashing in a rectangular tank with dimension 1×2 and grid cell size $\Delta x = 1/255$. The time step Δt is restricted to be in the range $[0.005, 0.01]$. 20 particles are initialized in every cell that is initially in a narrow band of $7\Delta x$ below the zero isocontour of the level set. Figure 2.12 Bottom shows an analogous 3D example where a sphere of water splashes in a tank. A cell size of $\Delta x = \frac{1}{63}$ is used in a domain with dimensions $1 \times 2 \times 1$. We take a fixed time step of $\Delta t = 0.01$ and demonstrate that narrow banding does not prevent larger-than-CFL time

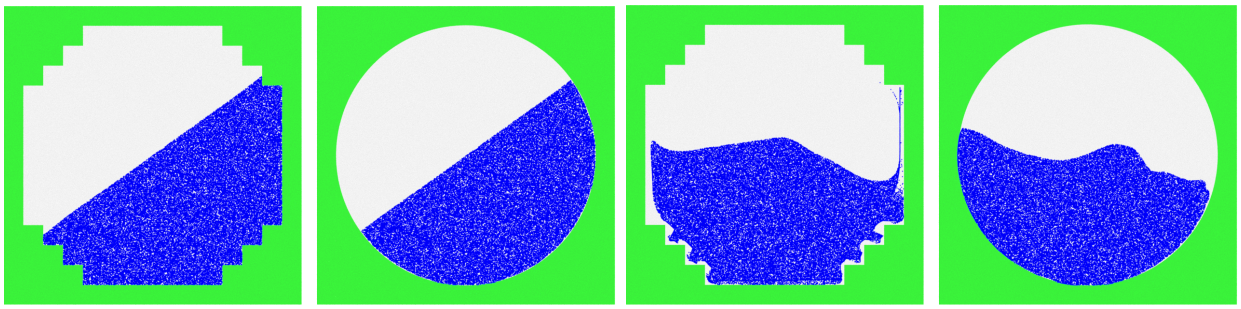


Figure 2.15: **Cut Cell vs. Voxelized Domain.** Using a cut cell domain (right) instead of a voxelized domain (left) yields marked improvements in simulation quality.

steps. 1,008,187 particles are used to resolve the free surface in a narrow band of width $5\Delta x$. As in 2D, the particles capture highly dynamic behavior of the free surface while the level set is sufficient to represent the bulk fluid in the bottom half of the domain.

We also demonstrate our hybrid advection with a vortex shedding example (see Figure 2.14). The flow domain Ω is a 3×1 rectangle with circle of radius 0.05. We seed a band of particles of width .2 above the midline $y = .5$ for PolyPIC advection. Advection in the rest of the domain is done with BSLQB. The vorticity plot illustrates a seamless transition between the two advection schemes. The simulation was run with a grid resolution of $\Delta x = \frac{1}{255}$, CFL number of 4 (i.e. $\Delta t = \frac{4\Delta x}{v_{\max}}$), and inlet speed of 1.5.

2.7.2 BSLQB Comparisons

We demonstrate improved resolution of flow detail with BSLQB compared to explicit semi-Lagrangian in a 2D example of smoke flowing past a circle (see Figure 2.16) and with a 2D spinning circle example (see Figure 2.10). Note that particles are only used for flow visualization and not for PolyPIC advection in these examples. BSLQB exhibits more energetic, turbulent flows than semi-Lagrangian advection. Notably, the BSLQB result breaks symmetry sooner. In Figure 2.16 we also examine the effect of extremal values of the λ parameter described in Equation (2.19). A zero value of λ is quite dissipative compared to a full value

of $\lambda = 1$ for both semi-Lagrangian and BSLQB. As mentioned in Section 2.4.1, we generally found that keeping λ close to 1 provided the least dissipative behavior, while setting the value slightly less than 1 helped restore stability when necessary (one can also dynamically adjust this value over the course of a simulation, e.g. setting λ closer to 1 when vorticity is high to better resolve desirable details.). In Table 2.1 we examine the efficiency of semi-Lagrangian and BSLQB for various grid resolutions and values of λ. We see that BSLQB takes more time to run than semi-Lagrangian, and that time also increases slightly with higher values of λ. Similarly, in Table 2.2 we look at the stability of semi-Lagrangian and BSLQB for different values of λ and Δt. We observe that for $\lambda = 1$, both semi-Lagrangian and BSLQB are unstable when the time step is sufficiently small, though the instability vanishes when λ is reduced to 0.9. We illustrate this instability in Figure 2.21. In Figure 2.10, we initially set the angular velocity to 4 radians per second in a circle of radius .2 (with $\Omega = [0, 1] \times [0, 1]$). The simulation is run with $\Delta x = \frac{1}{511}$ and a $\Delta t = .02$ (CFL number of 3).

We also compare BSLQB with APIC and advection-reflection [ZNT18] in Figure 2.19. We again set the angular velocity of each circle to 4 radians per second in a circle of radius .2, and the simulation is run with $\Delta x = \frac{1}{127}$ and $\Delta t = .02$. Even at a lower resolution, both BSLQB and APIC exhibit more energetic flows than advection-reflection. BSLQB also shows more turbulent behavior compared to APIC.

In addition, in Figure 2.20 we compare our approach with Houdini's smoke tool using basic settings. We do this with the smoke past a sphere example from Figure 2.18. Both simulations used a grid resolution of $\Delta x = \frac{1}{127}$. Houdini's simulation is much faster per frame (see Table 2.3 for our Smoke Jet's average time per frame). However, our method exhibits finer detail at similar resolution.

We examine the convergence behavior of BSLQB for the 2D Burgers' equation $\frac{D\mathbf{u}}{Dt} = \mathbf{0}$ with initial data $\mathbf{u}(\mathbf{x}) = \mathbf{x} \cdot (\mathbf{A}\mathbf{x})$ for $\mathbf{A} = \mathbf{R}\mathbf{\Lambda}\mathbf{R}^T$ for diagonal $\mathbf{\Lambda}$ with entries 1 and .25 and rotation (of .1 radians) $\mathbf{R}$ (see Figure 2.17). We examine the convergence behavior under refinement in space and time with $\Delta t = \Delta x$. We compute the best fit line to the plot of

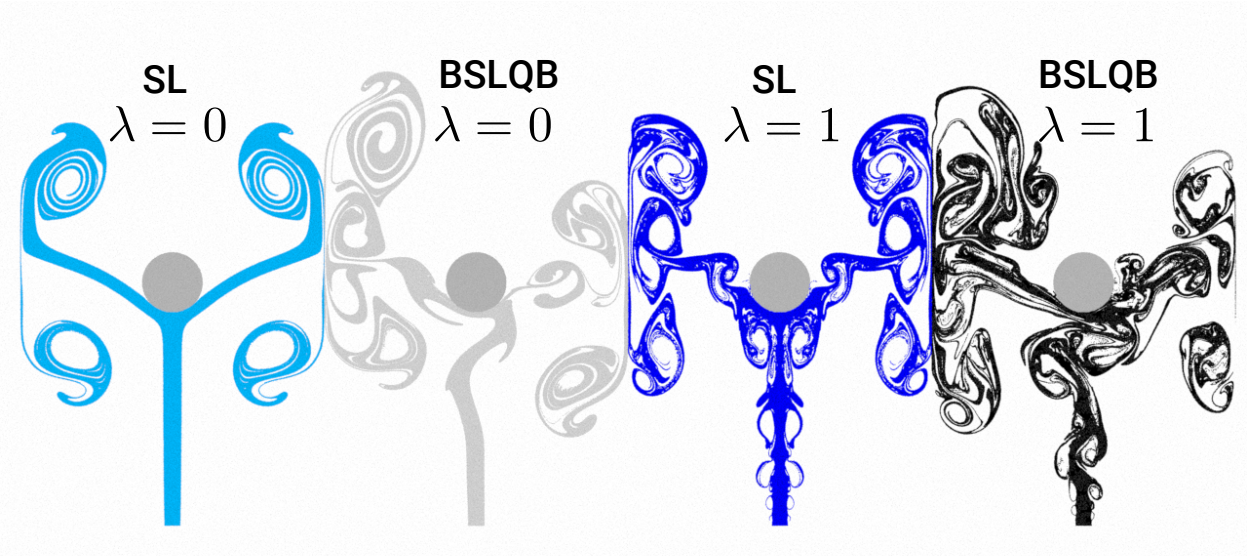


Figure 2.16: **Interpolation Correction.** BSLQB exhibits more fine-scale flow detail and vorticity than semi-Lagrangian for extremal values of interpolation parameter λ (Equation (2.19)). From left to right: semi-Lagrangian with $\lambda = 0$, BSLQB with $\lambda = 0$, semi-Lagrangian with $\lambda = 1$, BSLQB with $\lambda = 1$.

the logarithm of the L^∞ norm of the error versus the logarithm of Δx for a number of grid resolutions. We observe slopes of approximately 2 for BSLQB with interpolation parameter $\lambda = 1$ and $\lambda = 1 - c\Delta x$ (with $c = 2.95$), indicating second order accuracy in space and time under refinement. We observe slopes of approximately 1 for explicit semi-Lagrangian, indicating first order.

2.7.3 Cut Cell Examples

We demonstrate the ability of our cut cell method to produce detailed flows in complicated irregular domains for smoke and free surface water examples. Figure 2.5 demonstrates the subtle and visually interesting behavior that arises as two plumes of multicolored smoke flow to the center of a cubic domain colliding with a spherical boundary. We use $\Delta x = 1/63$ and $\Delta t = .02$. We demonstrate a more complex domain in Figure 2.7. Puffs of colored smoke

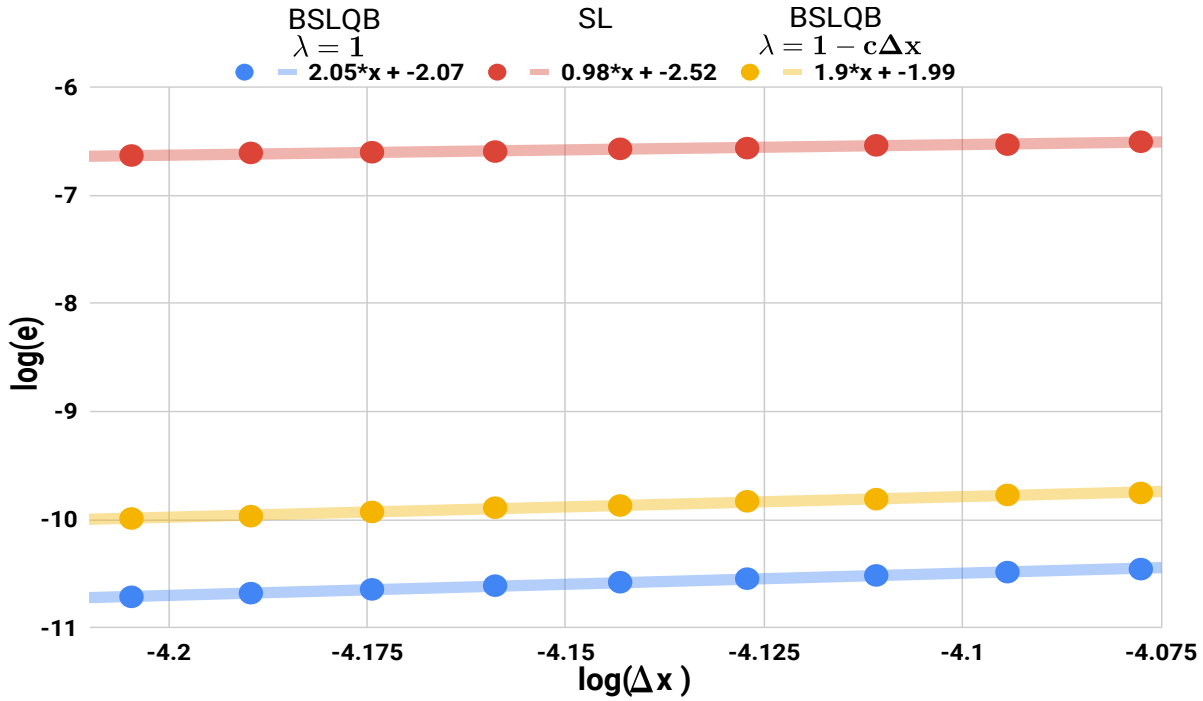


Figure 2.17: **Convergence.** We compare explicit semi-Lagrangian (SL, red), with BSLQB (blue) and interpolation coefficient $\lambda = 1$ (Equation (2.19)) and BSLQB with interpolation coefficient $\lambda = 1 - c\Delta x$ (orange) to a final time of 1. We plot $\log(\Delta x)$ versus $\log(e)$ (where e is the infinity norm of the error) for a variety of grid resolutions Δx and compute the best fit lines. The slope of the line provides empirical evidence for the convergence rate of the method.

with converging initial velocities are placed in a bunny shaped clear domain. We use a grid cell size of $1/127$ and a fixed time step of $\Delta t = 0.01$ (CFL number > 1). In Figure 2.8, we demonstrate water splashing, while accurately conforming to the walls of an irregular domain defined as the interior of a large sphere and exterior of a small inner sphere. The spatial resolution of the domain is $\Delta x = 1/127$, and 30 particles per cell are seeded in the initial fluid shape. A minimum time step of $\Delta t = 0.001$ is enforced, which is often larger than the CFL condition. We also consider dam break simulations in rectangular domains with column obstacles (Figure 2.6) and a bunny obstacle (Figure 2.9). Both examples use a

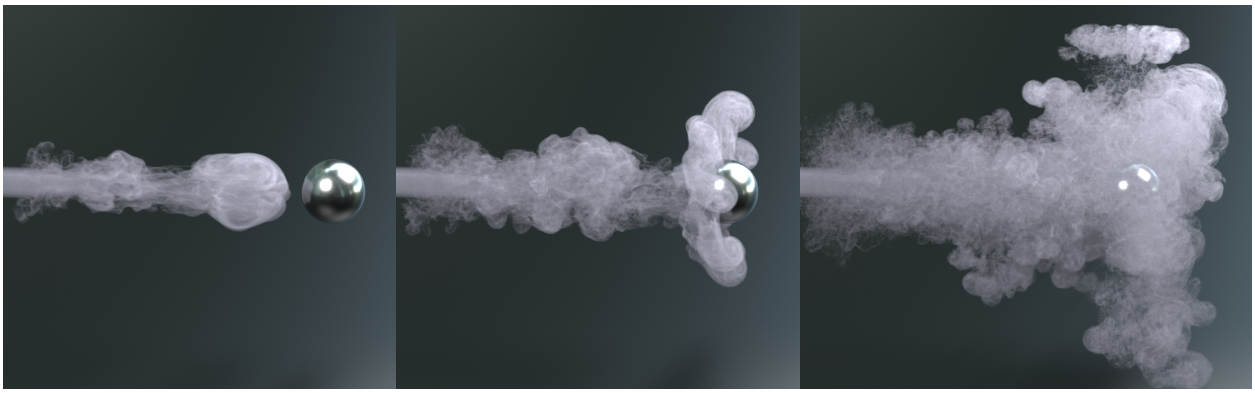


Figure 2.18: **Smoke Jet.** A plume of smoke is simulated with BSLQB. Zero normal velocity boundary conditions are enforced on the irregular boundary of the sphere inducing intricate flow patterns as the smoke approaches it.

$\Delta x =$	1/31	1/63	1/127	1/255	1/511
SL ($\lambda = 1$)	3.16	11.26	48.25	260.26	1329.88
SL ($\lambda = 0.5$)	2.62	11.05	48.07	252.68	1263.91
SL ($\lambda = 0$)	2.31	10.66	44.35	238.84	1193.98
BSL ($\lambda = 1$)	4.92	19.33	79.86	393.47	1838.78
BSL ($\lambda = 0.5$)	4.64	18.53	77.36	378.95	1777.42
BSL ($\lambda = 0$)	4.49	18.19	74.75	365.98	1707.63

Table 2.1: **SL/BSL 2D Run Time Comparison.** Comparison of run times (in seconds) for SL and BSL for three values of the interpolation parameter λ in 2D. The example is that of Figure 2.16 with a fixed time step of $\Delta t = 0.2$ at various grid resolutions out to a total time of 4.

grid cell size of $\Delta x = 1/127$, 8 particles per cell and a fixed time step of $\Delta t = 0.003$. Lastly, we demonstrate the benefits of our cut cell formulation over a more simplified, voxelized approach in Figure 2.15. Notice the water naturally sliding in the cut cell domain compared with the jagged flow in the voxelized domain.

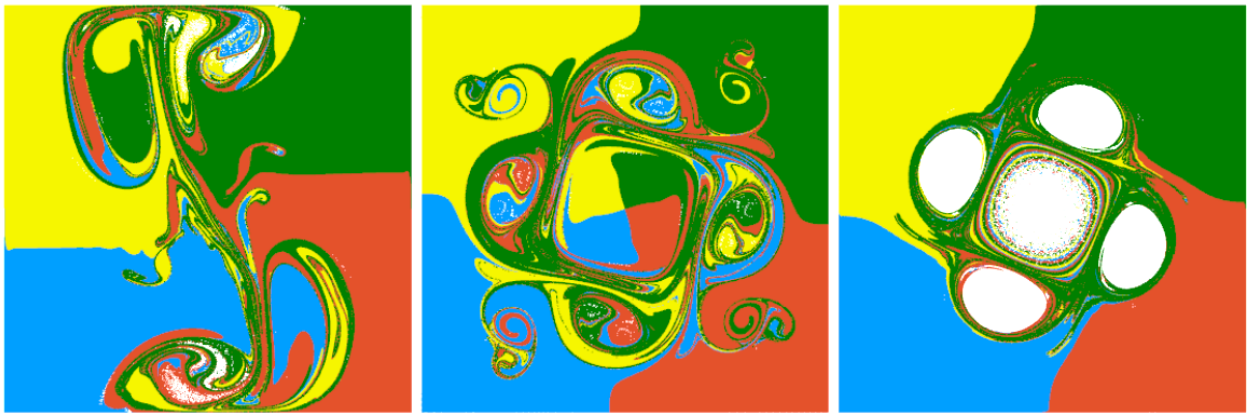


Figure 2.19: **BSLQB Compared to Other Advection Schemes.** From left to right: BSLQB, APIC, and Advection-Reflection at time = 6.

$\Delta t =$	.02	.001	.0005	.00025	.0001
SL ($\lambda = 1$)	✓	✓	✓	✓	×
SL ($\lambda = 0.9$)	✓	✓	✓	✓	✓
SL ($\lambda = 0$)	✓	✓	✓	✓	✓
BSL ($\lambda = 1$)	✓	✓	×	×	×
BSL ($\lambda = 0.9$)	✓	✓	✓	✓	✓
BSL ($\lambda = 0$)	✓	✓	✓	✓	✓

Table 2.2: **SL/BSL 2D Stability Comparison**. Comparison of stability for SL and BSL at three values of the interpolation parameter λ in 2D. The example is that of Figure 2.16 with a fixed resolution of $\Delta x = 1/127$ at various time steps Δt out to a total time of 2. Stable simulations are marked with a check mark, while unstable simulations are marked with an x.

2.7.4 Performance Considerations

The implementation of our method takes advantage of hybrid parallelism (MPI, OpenMP, and CUDA/OpenCL) on heterogeneous compute architectures in order to achieve practical

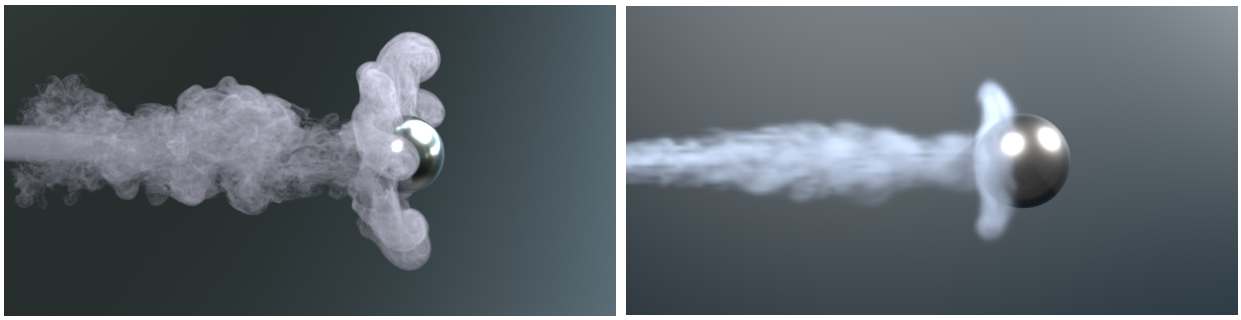


Figure 2.20: **Comparison with Houdini Smoke.** We compare the smoke jet of Figure 2.18 with Houdini's Billowy Smoke using basic settings at similar grid resolution. Our method displays more detail without additional disturbance or turbulence.

runtime performance (see Table 2.3 for 3D example performance numbers). The spatial domain is uniformly divided into subdomains assigned to distinct MPI ranks, which distributes much of the computational load at the expense of synchronization overhead exchanging ghost information across ranks. On each rank, steps of our time integration loop such as BSLQB advection are multithreaded using OpenMP or CUDA when appropriate. The dominant costs per time step are the solution of the pressure projection system and, in the case of free surface simulation, assembly of the pressure system and its preconditioner. We permute Equation (2.28) so that each rank's degrees of freedom are contiguous in the solution vector then solve the system using AMGCL [Dem19] using the multi-GPU VexCL backend (or the OpenMP CPU backend on more limited machines). Using a strong algebraic multigrid preconditioner with large-degree Chebyshev smoothing allows our system to be solved to desired tolerance in tens of iterations, even at fine spatial resolution. An important step in minimizing the cost of system assembly is to scalably parallelize sparse matrix-matrix multiplication, for which we use the algorithm of Saad [Saa03]. In the future, we are interested in implementing load balancing strategies such as the simple speculative load balancing approach of [SHQ18], particularly for free surface flows. We note that our implementation enables high-resolution simulations such as that in Figure 2.4 at relatively modest computational cost (see Table 2.3).

Example	Seconds	# Particles	# Nodes
Smoke Jet (Fig. 2.18)	1,212	12,502,349	$2 * 127^3$
Multiple Jets (Fig. 2.5)	53	25,004,699	63^3
Bunny Smoke (Fig. 2.7)	160	24,000,000	127^3
Smoke Spheres* (Fig. 2.4)	428	64,000,000	255^3
Narrow Band (Fig. 2.12)	396	1,008,187	$2 * 63^3$
Water Globe (Fig. 2.8)	242	524,415	127^3
Dam Break (Fig. 2.6)	870	3,251,409	$2 * 127^3$
Bunny Dam Break (Fig. 2.9)	1,171	4,797,535	$2 * 127^3$

Table 2.3: **Average Time Per Frame for 3D Examples.** Average time per frame (in seconds) for each of the 3D examples shown in this chapter. Examples were run on workstations with 16-core CPUs running at 2.20 GHz, except for the smoke spheres example, which was run on a cluster equipped with CPUs running at 3.07 GHz and Nvidia Tesla V100 GPUs which were used for the linear solves.

2.8 Discussion and Limitations

Our approach has several key limitations that could be improved. First, our adoption of collocated multiquadratic velocity and staggered multilinear pressure is a significant departure from most fluid solvers utilized in graphics applications. We note that BSLQB and BSLQB/PolyPIC could be used with a MAC grid; however, each velocity face component would have to be solved for individually. Another drawback for our multiquadratic velocity and multilinear pressure formulation is that it gives rise to a very wide pressure system stencil consisting of 49 non-zero entries per row in 2D and 343 in 3D. Collocated approaches that make use of multilinear velocities and constant pressure give rise to 9 (2D) and 27 (3D) entries per row [ZZS17], however they do not allow for C^1 continuity and require spurious pressure mode damping. Our wide stencils likely negatively affect the efficacy of precondi-

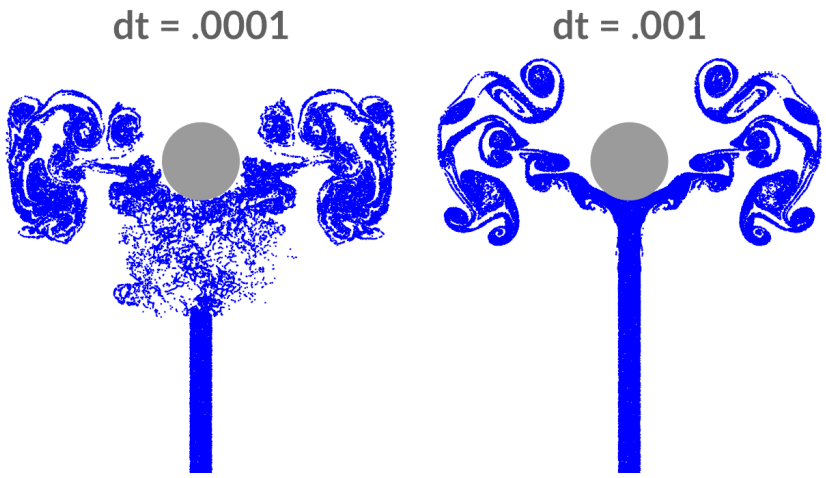


Figure 2.21: **Instability.** When λ is close (or equal) to 1, simulations can become unstable for smaller values of Δt. We show this here with a simulation of smoke against a circle at a spatial resolution of $\Delta x = 1/127$. This is seen for both semi-Lagrangian and BSLQB.

tioning techniques as well, however we were very pleased with the efficiency of the AMGCL [Dem19] library. Also, while the use of mass lumping in Equation (2.30) is necessary to ensure a sparse pressure projection system, Botella [Bot02] note that this has been shown to degrade accuracy. In fact, Botella [Bot02] introduce a sparse approximate inverse to the full mass matrix to avoid dense systems of equations in the pressure solve without degrading accuracy. Split cubic interpolation, which approximates similar systems with tridiagonal ones could also possibly be used for this [Hua94]. Adoption of one of these approaches with our formulation would be an interesting area of future work. Also, we note that the more sophisticated transition criteria for narrow banding techniques in Sato et al. [SWT18] could naturally be used with our method. Additionally, the free surface boundary in our approach is voxelized. In future work, we would like to use non-voxelized cut cell boundaries. Finally, we note that the work of Zehnder et al. [ZNT18, NZT19] could be easily applied to our technique to further reduce dissipation since it is based on the Chorin [Cho67] splitting techniques (Equations (2.9)-(2.11)) that we start from.

CHAPTER 3

A Robust Grid-Based Meshing Algorithm for Embedding Self-Intersecting Surfaces

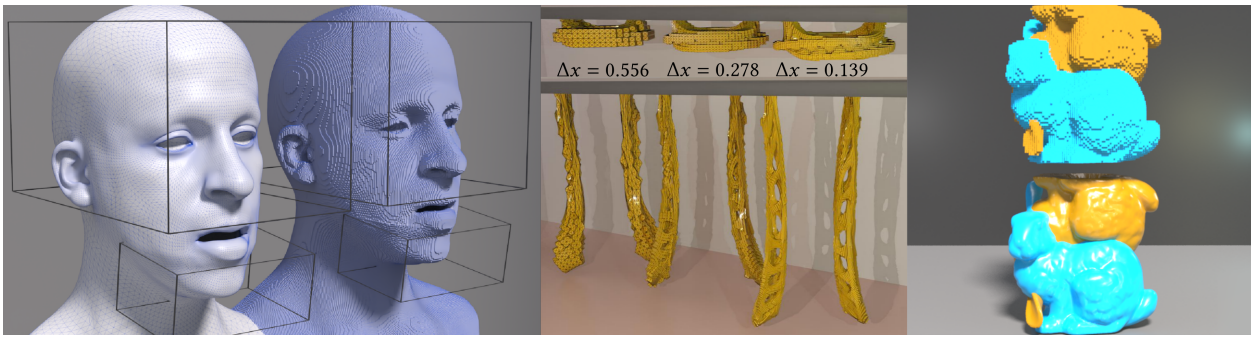


Figure 3.1: **Banner Image. Left:** Our method can generate a consistent volumetric mesh for a facial geometry that contains self-intersections e.g. around the lips. **Middle:** Two interlocking Möbius-strip-like bands separate freely at various spatial resolutions of the background grid, despite many near self-intersections in the surface geometry. **Right:** Two bunny geometries can naturally separate despite significant initial overlaps.

3.1 Introduction

It is often necessary in computer graphics and computational mechanics to create a volumetric mesh representing the interior of an input polygonal surface mesh. A volumetric tetrahedron mesh whose boundary coincides with an input surface triangle mesh is typically created [MBT03, LS07, HZG18, Si15]. It is also common to create a volumetric embedding mesh which contains the surface mesh but has a different boundary [SDF07, TBF19,

KBT17, TSB05]. In either case, a common requirement is that the surface mesh is closed and orientable.

Another common requirement is that the input surface mesh is not self-intersecting or overlapping. Despite many surface mesh generation techniques address the prevention of such features [HPS11, FTS06, Att10, ACW06, GD01], as noted in e.g. [SJP13, LB18], self-intersecting meshes are still common. This can occur in the production pipeline when the people creating the surface meshes are not involved in the processing of these meshes and are unaware of the importance of preventing self-intersection. Additionally, removal of self-intersection can be time consuming endeavor which is not worth the time investment. Additionally, it is even desirable on occasion for self-intersections to be present, such as when lips overlap in the neutral state of a deformable mesh, as non-overlapping contact is not a stress-free state [CBE15, CBF16].

Violation of this constraint can lead to the failure of various mesh creation methods, with either no output mesh being generated or an output mesh which has incorrect connectivity. There are, however, also methods which are robust to self-intersection [SJP13, LB18] or slight self-intersection [TSB05, LB18]. While some such approaches modify the input surface, Li and Barbič [LB18] create embedding tetrahedron meshes from an unmodified surface mesh with self-intersection by computing locally-injective immersions which are then used to unambiguously duplicate embedded mesh regions near overlapping regions.

We present a method for creating a well-defined uniform grid based embedding hexahedron mesh $\mathcal{V}$ corresponding to an input triangle surface mesh $\mathcal{S}$ which is in general self-intersecting. As in [SJP13], we assume there exists a nearby non-self-intersecting mesh $\tilde{\mathcal{S}}$ and a mapping $\boldsymbol{\phi}_{\tilde{S}}^{S} : \tilde{\mathcal{S}}^{V} \to \mathbb{R}^3$ with non-singular Jacobian determinant (see Figure 3.2); here $\tilde{\mathcal{S}}^{V}$ is the unambiguously defined interior of the non-self-intersecting $\tilde{\mathcal{S}}$. However, unlike [SJP13], we do not explicitly generate the mapping or the non-self-intersecting surface; we instead rely on their implicit existence. We build our volumetric mesh by copying background cells of contiguous regions to create sub-meshes which are then sewn together in an

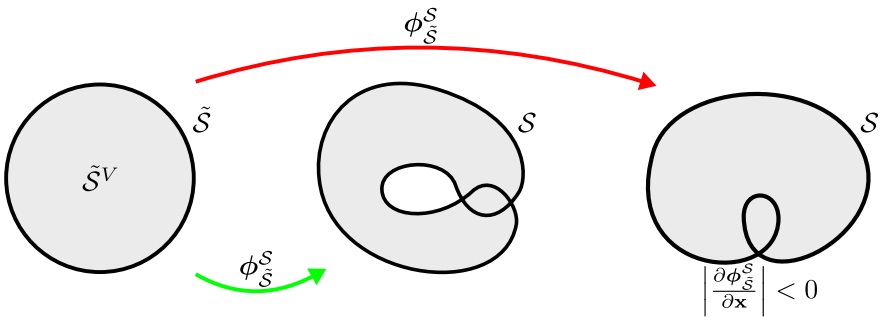


Figure 3.2: **Intersection-Free Mapping.** Two mappings from a non-self-intersecting region $\tilde{\mathcal{S}}^V$ to self-intersecting boundary $\mathcal{S}$ are shown. The second mapping (right) requires the existence of a negative Jacobian determinant.

appropriate manner using techniques inspired by Teran et al. [TSB05].

Our approach is quite similar to Li and Barbič [LB18] in various ways: both create a volumetric embedding mesh without input surface modification and our duplication and sewing method is equivalent to immersion computation. However, our method uses less exact and/or adaptive precision arithmetic since we do not need to resolve the exact intersection between triangle-triangle pairs or triangles and background cells; we only need to know whether or not triangles intersects a particular cell or edge. Additionally, we do not use CSG operations [SDF07].

This approach places limits on how coarse the grid spacing can be, and we often need to run with high resolutions. In order to limit element counts and reach a desired lower resolution, we introduce a topology-preserving mesh coarsening strategy similar to [WJS14]. Finally, we introduce a topology-preserving method for converting our hexahedron mesh to a tetrahedron mesh using a body-centered cubic (BCC) structure [MBT03] as in [LB18].

Our contributions are summarized below:

- An efficient technique with reduced use of exact/adaptive precision arithmetic for building an embedding hexahedron mesh for an input self-intersecting triangle mesh from a uniform grid that is equivalent to pushing forward one unambiguously defined from a

self-intersection-free state.

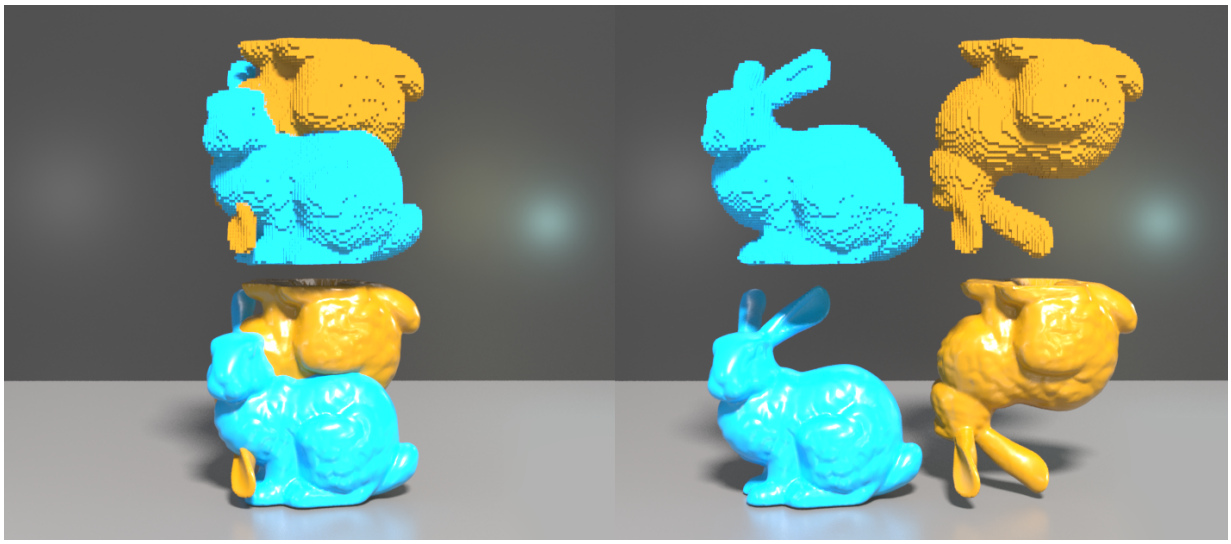
- A topology aware embedding mesh coarsening strategy to provide for flexible resolution/element count.
- A topology aware BCC approach for converting the embedding hexahedron mesh into an embedding tetrahedron mesh.

3.2 Related Work

3.2.1 Volumetric Mesh Creation from a Self-Intersecting Triangle Mesh

Sacht et al. [SJP13] were the first to design an approach that creates an appropriately overlapping tetrahedron mesh from a self-intersecting triangle mesh. As with our approach, they assume the existence of a mapping $\boldsymbol{\phi}_{\tilde{S}}^{S}$ from a non-self intersecting counterpart $\tilde{\mathcal{S}}$ to the input mesh $\mathcal{S}$. Unlike our approach, they explicitly form $\tilde{\mathcal{S}}$ and the mapping $\boldsymbol{\phi}_{\tilde{S}}^{S}$. $\tilde{\mathcal{S}}$ is created by a backward process using cMCF followed by a forward process that minimizes distortion-energy and deviation from $\mathcal{S}$ subject to collision constraints. The cMCF is known to remove self-intersections for sphere-topology surfaces [KSB12] and accordingly, their method is limited to input surfaces with genus zero. They create a tetrahedron mesh using the self-intersection free $\tilde{\mathcal{S}}$ and then push it forward under $\boldsymbol{\phi}_{\tilde{S}}^{S}$ which is created by mapping the boundary of the tetrahedron mesh to $\mathcal{S}$ and propagating deformation to the interior. Our approach is similar in spirit, but we do not explicitly create $\tilde{\mathcal{S}}$ or $\boldsymbol{\phi}_{\tilde{S}}^{S}$; furthermore, we can support input surfaces with genus larger than zero. In addition, since they do not directly generate tetrahedra in world space, they must take care to maintain tetrahedron mesh quality under deformation in $\boldsymbol{\phi}_{\tilde{S}}^{S}$.

Like Li and Barbič [LB18], we create a volumetric embedding mesh in world space. Li and Barbič [LB18] observed that the creation of a volumetric mesh from a self-intersecting surface is related to the geometric and algebraic topological determination of immersions (locally



(a) Frame 1 (b) Frame 54

Figure 3.3: **Twin Bunnies.** Two overlapping bunnies naturally separate. The top part of each subfigure shows the meshes generated by our algorithm, while the bottom part of each subfigure shows the corresponding surface meshes.

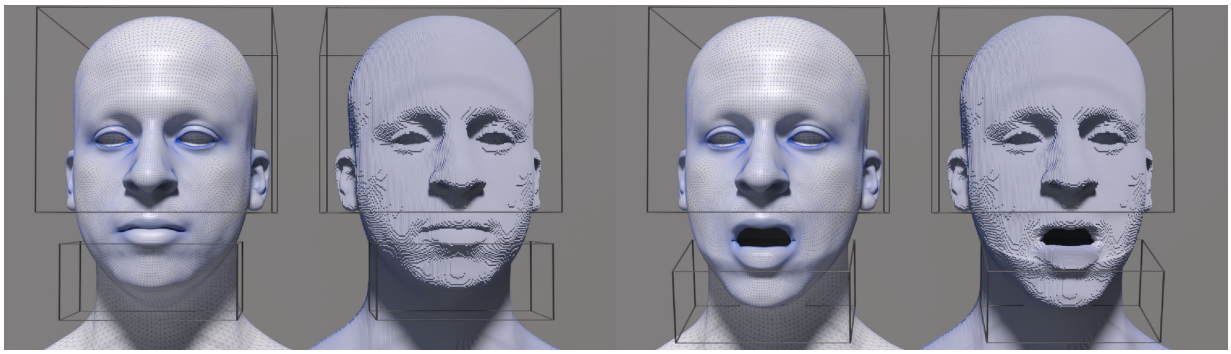
injective mappings) from a compact 3-manifold to a portion of the world space domain. As in our approach, they start by dividing world space into contiguous regions using the input surface mesh $\mathcal{S}$. However, they use exact/adaptive precision arithmetic to intersect $\mathcal{S}$ with itself to achieve this. We use simplified/less costly intersections of triangles in $\mathcal{S}$ with uniform background grid cells and edges. We only need to know whether an intersection occurs or not; we do not need to resolve the intersection geometry. Immersions do not always exist, and Li and Barbič [LB18] developed a graph based algorithm to determine if one exists. Their method for computing these is NP-complete; however, as they note, this is not a bottleneck for most computer graphics applications. When such an immersion exists, they compute it by duplicating the contiguous regions, intersecting each duplicate with a uniform background tetrahedron lattice to create local tetrahedron meshes that are then sewn together appropriately using their graph structure. We also duplicate and then sew together contiguous regions, but we use simplified criteria that, while more efficient, can only

give accurate results for simple immersions. Although, as Li and Barbič [LB18] note, the vast majority of applications in computer graphics only require simple immersions. As with our approach, they also prevent artificial glueing for embedded meshes with nearly intersecting features. While Li and Barbič [LB18] can accurately compute non-simple immersions, they cannot handle exactly coincident portions $\mathcal{S}$ with non-zero measure, which we can handle. Broadly speaking, the Li and Barbič [LB18] approach is more general than our method, but more costly, primarily due to the comparably large use of exact/adaptive precision arithmetic.

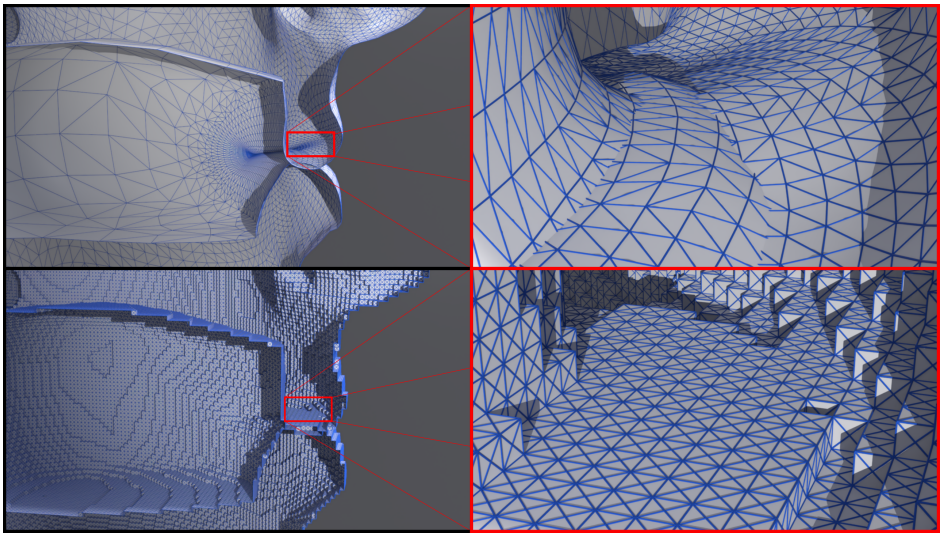
3.2.2 Mesh Creation and Mesh Cutting

The virtual node algorithm (VNA) of [MBF04] allows cutting a tetrahedron mesh along piecewise-linear paths through the mesh. As in our approach, duplicates of cut elements are used to resolve necessary topological features. Teran et al. [TSB05] built a generalization of this approach to create embedding meshes for nearly overlapping input triangle meshes. Sifakis et al. [SDF07] further extended the VNA to allow for arbitrary cut geometry. A downside to the geometric flexibility provided by these generalizations is their need for adaptive precision arithmetic and CSG. Motivated by this, Wang et al. [WJS14] developed a technique that allows for geometric flexibility without the need for adaptive precision arithmetic. Their approach allows for arbitrary cut surfaces by generalizing the original VNA [MBF04] to allow cuts to pass through vertices, edges, or faces of the embedding mesh. This alone does not provide sufficient geometric flexibility since cuts cannot pass through facets multiple times. To resolve such cuts, the algorithm is run at high-resolution where facets are only intersected once and then coarsened in a topologically-aware manner.

The extended finite element method (XFEM) [BB99] is very similar to VNA. An XFEM-based but remeshing-free approach for cutting of deformable bodies is presented in [KBT17]. In a similar spirit, Zhang et al. [ZDZ18] utilized the cracking node method [SB09], which is similar to XFEM but uses discontinuous cracks centered at nodes in order to approximate



(a) Frame 0 (b) Frame 80



(c) Interior view of lips

Figure 3.4: **Intersecting Lips.** A face surface with self-intersecting lips is successfully meshed. The right-hand side of each of the first four frames shows the deformed hexahedron mesh, while each left-hand side shows the corresponding surface mesh. The wireframe boxes represent Dirichlet boundary condition regions. In the bottom four subfigures, lip intersection is visualized in the input surface and subsequent hexahedron mesh.

crack paths. This yields an efficiency advantage over XFEM which in turn allows for simulating materials with many evolving, branching cracks. The reader is also referred to the survey of Wu et al. [WWD15] for more discussion of mesh cutting techniques in computer graphics.

More generally, tetrahedron mesh creation has been robustly addressed by a number of works [Si15, HZG18, LS07, MBF03, DCB13, JAY15]. For example, Si [Si15] pursued a Delaunay refinement strategy in order to provide certain guarantees on tetrahedron quality. However, sliver tetrahedra are still possible [HZG18]. The method presented in [HZG18] can handle arbitrary triangle soup as input and returns a high-quality approximated constrained tetrahedron mesh, though performance is hindered to an extent due to prominent usage of exact rational arithmetic. However, recently, those performance bottlenecks were alleviated and replaced with floating-point computations [HSW20]. Notably, researchers have recently presented a successful method for learning high-quality tetrahedron meshes from noisy point clouds or a single image [GCX20].

3.2.3 Self-Intersecting Curves and Surfaces

Self-intersecting curves and surface meshes have been considered for many years in both the mathematics and computer science literature. In two dimensions, algorithms and theorems related to identifying self-intersecting curves date back to [Tit61], with many more recent contributions [Bla67, Mar74, SV92, HL95, GC11, EFW20]. Notably, many problems related to identifying self-intersections are NP-complete [EM09]. Despite this, efficient algorithms frequently exist; for example, Mukherjee [Muk14] gave a quadratic algorithm (in the number of points on the discrete curve) to determine the mapping from a disk to an arbitrarily stretched, potentially self-overlapping curve, also known as computing an immersion of the disk. In another vein, Li [Li11] used Gauss diagrams from knot theory to characterize self-intersecting two-dimensional projections of three-dimensional polygons, in order to understand whether there are one or multiple ways to perform mesh repair algorithms like

[BWS09].

In the context of three-dimensional mesh generation and animation, self-intersections are typically treated as degeneracies to be avoided or removed. For example, Von Funck et al. [FTS06] provided a method for deforming surfaces that prevents new self-intersections from occurring, due to the smoothness requirements they place on the vector fields governing the deformation. The tool devised in [ACW06] allows for local prevention of self-intersections when deforming a mesh. A method for avoiding introducing self-intersections within the free-form deformation (FFD) modeling scheme [Bez70, SP86] was presented in [GD01]. The space-time interference volumes introduced in [HPS11] can be used to eliminate self-intersections in meshes, although this method is not always guaranteed to work (the method is primarily intended for interacting with non-self-intersecting input geometry). Shen et al. [SOS04] built an implicit surface from polygon soup, resulting in a watertight mesh that approximates the input surface data. Attene [Att10] deleted overlapping triangles and subsequently performed a gap-filling procedure in the resulting holes. Similarly, Jacobson et al. [JKS13] presented a method based on the generalized winding number (which, notably, is still applicable to triangle soups and point clouds [BDS18], unlike the standard winding number). Their method results in fusing together self-intersecting parts of the mesh. Recently, Tao et al. [TBF19] demonstrated a method for accurately and efficiently generating cut cell meshes for arbitrary triangulated surfaces, including those with degeneracies. However, again, they treat self-intersections as flaws to be removed, unlike in our method where self-intersections are valid features of our inputs and outputs. Nonetheless, an attractive aspect of their algorithm is robust resolution of mesh degeneracies and singularities, unlike methods like [EB14, KT10] which require random numerical perturbations of the background cut cell grid. Finally, we also highlight [MAS15], which describes a method for representing self-intersecting surfaces using implicit functions sampled on a specialized hexahedron mesh.

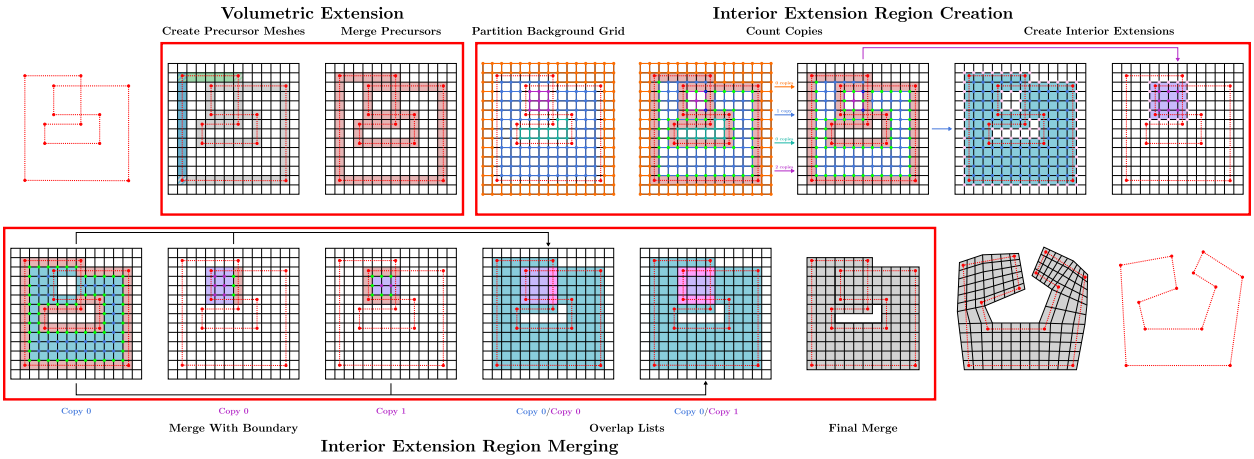


Figure 3.5: **Algorithm Overview.** Given an initial input surface mesh $\mathcal{S}$, there are three major steps in the computation of the final volumetric extension mesh $\mathcal{V}$: Volumetric Extension, Interior Extension Region Creation, and Interior Extension Region Merging. **Volumetric Extension:** In this step, we create a precursor mesh for each element in $\mathcal{S}$, and compute preliminary signing information for the vertices. We then merge the precursor meshes to create the volumetric extension $\mathcal{U}$ and correct the signing information where necessary. **Interior Extension Region Creation:** In preparation for growing the volumetric extension into the interior, we first partition the nodes of the background grid using the edges cut by $\mathcal{S}$. We decide which regions are interior and count the copies of each region using the vertices of $\mathcal{U}$ which have negative sign. For each interior region j^I with at least one copy, we then create a hexahedron mesh $\mathcal{V}^{j^I,c}$ for each copy c. **Interior Extension Region Merging:** The merging process begins with copying relevant hexahedra from $\mathcal{U}$ into $\mathcal{V}^{j^I,c}$. First, certain vertices of $\mathcal{V}^{j^I,c}$ are replaced by corresponding vertices from $\mathcal{U}$. Hexahedra to be replaced are then removed from $\mathcal{V}^{j^I,c}$ before the boundary hexahedra are copied in. We then merge the various meshes $\mathcal{V}^{j^I,c}$ by first determining where different meshes overlap, and then using these hexahedra overlap lists to perform the final merge.

3.3 Algorithm Overview

The input to our algorithm is a triangulated surface mesh $\mathcal{S}$. The output is a uniform-grid-based embedding hexahedron mesh counterpart $\mathcal{V}$ to $\mathcal{S}$ that is well-defined (i.e., free from numerical mesh "glueing" artifacts) even when $\mathcal{S}$ is self-intersecting (see Section 3.10 for examples).

We briefly summarize the three main stages of our algorithm, as detailed in Figure 3.5. In the first stage, volumetric extension (Section 3.5), we create a hexahedron mesh $\mathcal{U}$ from the background grid that only covers the input surface $\mathcal{S}$ with connectivity designed to mimic it. We sign its vertices depending on inside/outside information derived from the hypothetical self-intersection-free counterpart $\tilde{\mathcal{S}}$. We emphasize that this volumetric extension mesh only surrounds $\mathcal{S}$. Accordingly, the second stage of the algorithm is interior extension region creation (Section 3.6). Nodes of the background grid are partitioned using the edges cut by $\mathcal{S}$, and then we decide which regions are interior. Interior regions will be copied a certain number of times corresponding to the number of times which interior portions of the hypothetical self-intersection-free counterpart $\tilde{\mathcal{S}}^V$ will overlap under the hypothetical push forward mapping $\boldsymbol{\phi}_{\tilde{S}}^{S}$; the number of copies is approximate at this stage. For each interior region j^I with at least one copy, we create a hexahedron mesh $\mathcal{V}^{j^I,c}$ for each copy c. In the third stage of the algorithm (Section 3.7), interior extension regions meshes $\mathcal{V}^{j^I,c}$ are sewn together and into the volumetric extension $\mathcal{U}$ to produce the final output mesh. We additionally provide a coarsening approach in Section 3.8 to provide user control over the embedding mesh resolution as well as a topologically-aware technique for converting the hexahedron mesh $\mathcal{V}$ into a tetrahedron mesh $\mathcal{T}$.

3.4 Definitions and Notation

We take a triangle mesh $\mathcal{S} = (\mathbf{x}^S, \mathbf{m}^S)$ as input. We use $\mathbf{x}^S = [\mathbf{x}_0^S, \ldots, \mathbf{x}_{N_v^S-1}^S] \in \mathbb{R}^{3N_p^S}$ to denote the vector of triangle vertices $\mathbf{x}_i^S \in \mathbb{R}^3$ and $\mathbf{m}^S \in \mathbb{N}^{3N_e^S}$ to denote the vector of

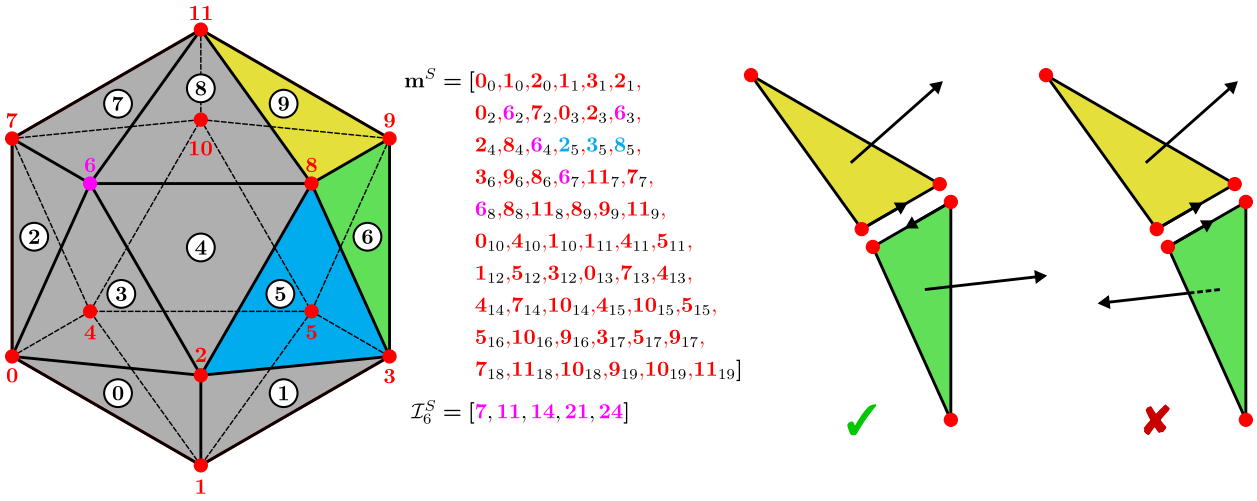


Figure 3.6: **Mesh Conventions. Left:** A sample triangle mesh is shown, along with the vector $\mathbf{m}^S$. The incident mesh indices $\mathcal{I}_6^S$ for vertex 6 are also shown. For example, triangle 4 has vertices $2, 8, 6$ at indices $12, 13, 14$ in $\mathbf{m}^S$. Hence, 14 is in $\mathcal{I}_6^S$. The first 10 faces, visible from the front, have been labeled on the mesh. **Right:** The left pair of triangles are consistently oriented; the orientations of the edge induced by the normals point in opposite directions. For the right pair, the orientations on the common edge point in the same direction; this is not consistent.

indices m_j^S for vertices in $\mathbf{x}^S$ corresponding triangles $t^S_{\lfloor \frac{j}{3} \rfloor}$, $0 \leq \lfloor \frac{j}{3} \rfloor < N_e^S$. For example, for the mesh $\mathcal{S}$ in Figure 3.6, triangle t_5^S is made up of vertices $\mathbf{x}^S_{m_j^S}$ with $j = 15, 16, 17$ where m_j^S equals $2, 3, 8$, respectively. We assume that $\mathcal{S}$ is closed (every edge in the mesh has two incident triangles) and consistently oriented (each edge appears with opposite orientations in its two incident triangles). For each vertex $\mathbf{x}_i^S$ of $\mathcal{S}$, we use $\mathcal{I}_i^S$ to denote the set of incident mesh indices j such that $i = m_j^S$. Figure 3.6 demonstrates these conventions. We output a hexahedron mesh $\mathcal{V} = (\mathbf{x}^V, \mathbf{m}^V)$ with $\mathbf{x}^V \in \mathbb{R}^{3N_p^V}$ denoting the vector of hexahedron vertices and $\mathbf{m}^V \in \mathbb{N}^{8N_e^V}$ denoting the vector of indices in $\mathbf{x}^V$ corresponding to vertices in hexahedron h_e^V, $0 \leq e < N_e^V$. Each hexahedron in the mesh is geometrically coincident with one grid cell in a background uniform grid $\mathcal{G}_{\Delta x}$. We denote the spacing of this grid as Δx (uniformly in each direction). For ease of visualization, we use 2D counterparts to $\mathcal{S}$ and $\mathcal{V}$ in illustrative

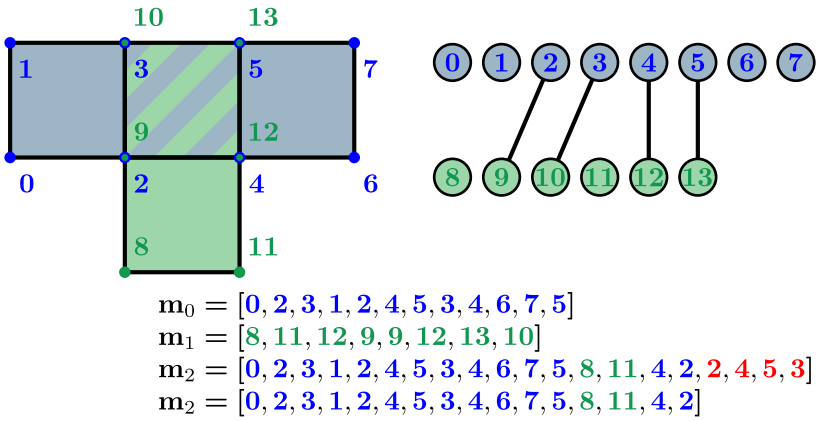


Figure 3.7: **Mesh Merge.** An example of two meshes merging together. Vertices 2, 3, 4 and 5 merge with vertices 9, 10, 12 and 13, respectively. A new vector $\mathbf{m}_2$ is created to hold all of the hexahedron vertices post-merge, and the extra hexahedron (in red) is then removed.

figures. In this case, $\mathcal{S}$ is a segment mesh and $\mathcal{V}$ is a quadrilateral mesh.

3.4.1 Merging

We construct the final hexahedron mesh $\mathcal{V}$ by merging portions of various precursor hexahedron meshes in a manner similar to techniques used in [TSB05, WDG19, WJS14, LB18]. As with $\mathcal{V}$, each hexahedron in a precursor mesh is geometrically coincident with background grid cells. All precursor meshes share the same vertex array $\mathbf{x}^V$, although its size will change as we converge to the final $\mathcal{V}$. At various stages of the algorithm, we will merge certain geometrically coincident precursor hexahedra. To perform a merge, we view the set of all vertices in $\mathbf{x}^V$ as nodes in a single undirected graph and introduce graph edges between nodes corresponding to geometrically coincident vertices. In subsequent sections, we refer to such edges in the undirected graph as adjacencies to distinguish them from edges in the various meshes. Once all adjacencies are defined, we compute the connected components of the graph using depth-first search. All vertices in a connected component are considered to be the same and we choose one representative for all mesh entries. We note that this operation may be carried out on more than two meshes at once and that it can lead to duplicate

hexahedra and in this case we remove all but one. Furthermore, replacing all vertices in a connected component with one representative results in unused vertices in $\mathbf{x}^V$. We remove all unused vertices in a final pass, changing indexing in $\mathbf{m}^V$ accordingly. We illustrate the connected component calculation, vertex replacement and unused vertex removal in Figure 3.7.

3.5 Volumetric Extension

We first create a volumetric extension $\mathcal{U}$ of the surface $\mathcal{S}$. It is a hexahedron mesh that contains the input surface $\mathcal{S}$ and is designed to have topological properties analogous to $\mathcal{S}$. Since it is an extension of $\mathcal{S}$, we can sign the vertices of $\mathcal{U}$ depending on which side of the surface they lie on. Overlapping regions in $\mathcal{S}$ complicate this process, but it can be disambiguated by considering the pre-image of the surface to its overlap-free counterpart $\tilde{S}$ under the mapping $\boldsymbol{\phi}^S_{\tilde{S}}$. Signing points in $\mathbb{R}^3$ depending on whether or not they are inside $\tilde{S}$ is well-defined and our procedure for signing the vertices in the volumetric extension $\mathcal{U}$ is designed considering its pre-image under $\boldsymbol{\phi}^S_{\tilde{S}}$.

3.5.1 Surface Element Precursor Meshes

In order to mimic the topology of the $\mathcal{S}$, we create its volumetric extension $\mathcal{U}$ from precursor meshes $\mathcal{U}_e = (\mathbf{x}^V, \mathbf{m}^U_e)$ associated with each triangle t^S_e in $\mathcal{S}$. Note that all precursor meshes share the common vertex array $\mathbf{x}^V$ and that this process begins its evolution to the final $\mathcal{V}$ vertex array. For each triangle t^S_e in $\mathcal{S}$, we define a hexahedron mesh from the subgrid $\mathcal{G}^{U_e}_{\Delta x}$ of $\mathcal{G}_{\Delta x}$ defined by the grid-cell-aligned bounding box of t^S_e. We add a new hexahedron to $\mathcal{U}_e$ corresponding to each background grid cell in $G^{U_e}_{\Delta x}$ intersected by t^S_e. We perform this operation using the intersection function from CGAL's 2D/3D Linear Geometry Kernel [The20, BFG20]. We note that every call to the CGAL library here and in subsequent sections uses the exact arithmetic kernel; conversely, all of our exact/adaptive precision arithmetic is

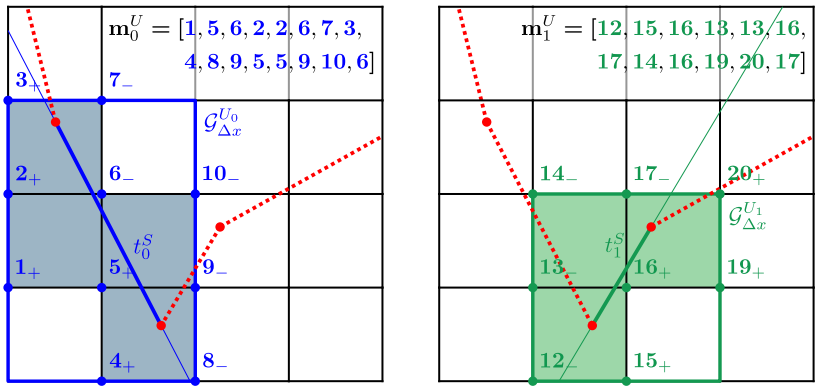


Figure 3.8: **Precursor Meshes. Left:** Surface element t_0^S creates quadrilateral mesh $\mathcal{U}_0$. **Right:** Surface element t_1^S creates quadrilateral mesh $\mathcal{U}_1$. Each element creates copies of the grid cells it intersects by introducing new vertices which are geometrically coincident to grid nodes.

limited to CGAL. The hexahedron is geometrically coincident to the intersected grid cell in $\mathcal{G}_{\Delta x}$, however the vertices introduced into the vertex vector $\mathbf{x}^V$ are copies of the background grid nodes associated with the sub grid $\mathcal{G}_{\Delta x}^{U_e}$. Note that even though different triangles may intersect the same grid cells, their respective hexahedra correspond to distinct vertices in $\mathbf{x}^V$. Further note that mesh elements in $\mathcal{U}_e$ inherit the connectivity of the sub grid $\mathcal{G}_{\Delta x}^{U_e}$, that is, hexahedra share common vertices if they are neighbors in $\mathcal{G}_{\Delta x}^{U_e}$. We sign the vertices in each $\mathcal{U}_e$ depending on which side of the plane containing the triangle t_e^S that they lie on. We illustrate this process in Figure 3.8. Lastly, we note that these signs are low-cost preliminary approximations to the signs in the final volumetric extension $\mathcal{U}$. In some cases the signs computed in this phase will not be accurate in the volumetric extension, and we provide a more accurate but costly signing when this occurs (discussed in Section 3.5.2; however, in many cases, they are equal to the final signs, and their comparably low computational cost improves overall algorithm performance.

3.5.2 Merge Surface Element Meshes

We merge portions of the precursor meshes $\mathcal{U}_e$ to form the volumetric extension hexahedron mesh $\mathcal{U}$ by defining adjacency between vertices in $\mathbf{x}^V$ as described in Section 3.4.1. We define this adjacency from the mesh connectivity of $\mathcal{S}$ using its incident elements $\mathcal{I}_i^S$ for each vertex $\mathbf{x}_i^S$. Geometrically coincident vertices in $\mathcal{U}_{\lfloor j_{i,0}^S/3 \rfloor}$ and $\mathcal{U}_{\lfloor j_{i,1}^S/3 \rfloor}$ for $j_{i,0}^S, j_{i,1}^S \in \mathcal{I}_i^S$ are defined to be adjacent if each are on hexahedrons in their respective meshes which are geometrically coincident. Note in particular that this is different from defining geometrically coincident vertices in $\mathcal{U}_{\lfloor j_i^S/3 \rfloor}$ for $j_i^S \in \mathcal{I}_i^S$ to be adjacent (see the geometry of Figure 3.16). In other words, all geometrically coincident hexahedra in element precursor meshes associated with triangles that share a common vertex are merged (see Figure 3.9), including hexahedra along common edges. Merged vertices retain the sign they were given in $\mathcal{U}_e$ when possible. However, if merged vertices have differing signs, e.g. in regions with higher curvature (see Figure 3.10), then we must recompute the sign from their geometric relation to $\mathcal{S}$.

In regions of higher curvature where the preliminary signs of vertices in $\mathcal{U}_e$ cannot be adopted in $\mathcal{U}$, we use an eikonal strategy [OF03] to propagate positive signs from $\mathcal{S}$ in the direction of the surface normal and minus signs in the opposite direction. This is well defined in light of the assumed existence of the pre-image $\tilde{\mathcal{S}}$ of $\mathcal{S}$ under $\boldsymbol{\phi}_{\tilde{S}}^S$. Here, each vertex $\mathbf{x}_i^V$ in the volumetric extension $\mathcal{U}$ is associated with some collection of precursor meshes $\mathcal{U}_{e_i}$ where $\mathbf{x}_i^V$ was created in the merge of vertices in the $\mathcal{U}_{e_i}$. This defines a local patch S_{iV} of surface triangles $t_{e_i}^S$ in $\mathcal{S}$ associated with $\mathbf{x}_i^V$. When propagating signs from $\mathcal{S}$ to $\mathbf{x}_i^V$, only these triangles are considered. It is important to only use this local surface patch since there may be triangles in $\mathcal{S}$ that are geometrically close to $\mathbf{x}_i^V$ but topologically distant. Note that this precludes the use of global point-in-polygon algorithms based on ray casting or winding numbers since those will not give correct results when $\mathcal{S}$ has self-intersection. Instead we adopt the local point-in-polygon method of Horn and Taylor [HT89]. First, we compute the closest mesh facet (triangle, edge, or point) in S_{iV} to $\mathbf{x}_i^V$. The closest facet calculation is performed by first storing S_{iV} in a CGAL surface mesh and then using its class functions

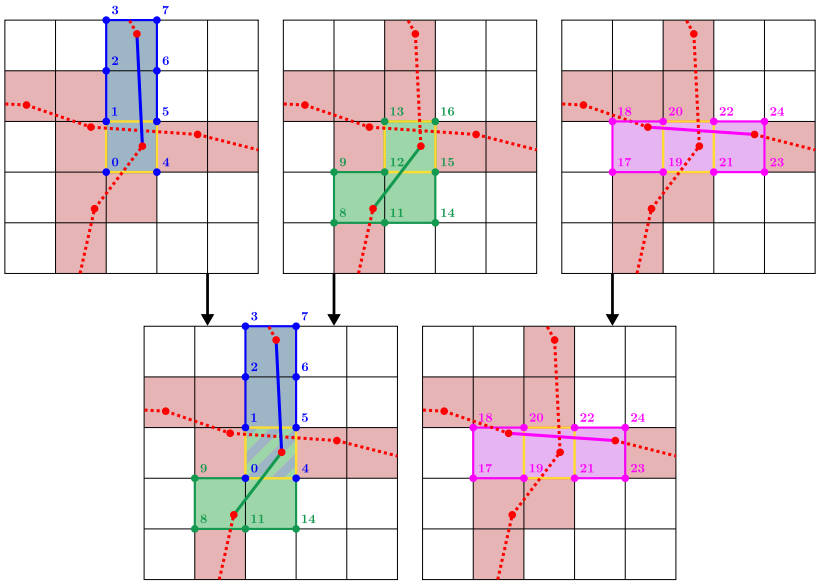


Figure 3.9: **Precursor Merge.** The 12 vertices bordering the cell marked in yellow are merged into 8 resulting vertices. Blue vertices 0, 1, 4, 5 and green vertices 12, 13, 15, 16 are merged, respectively. However, magenta vertices 19, 20, 21, 22 do not merge with the blue or green vertices since their associated surface element is topologically distant.

and the locate function from the Polygon Mesh Processing package [BSM20, LRT20]. If the closest facet is an edge or a point, we add triangles from $\mathcal{S}$ that are incident to the vertices in the edge or the point respectively to the patch S_{iV} (if they are not already in it). If more triangles were added, we recompute the closest mesh facet. We illustrate this process in Figures 3.10 and 3.11. If the closest facet is a triangle, we compute the sign depending on the side of the plane containing the triangle that the point lies on. If the closest faces is an edge or point we use the conditions from [HT89], which we summarize below:

- If the closest facet is an edge, then the sign is -1 if the edge is concave (as determined by the normals of the incident faces) and $+1$ if it is convex.
- If the closest facet is a vertex, then there exists a discrimination plane with an empty half-space. Choosing any such plane, the sign is -1 if the edges defining the plane are concave and $+1$ if they are convex.

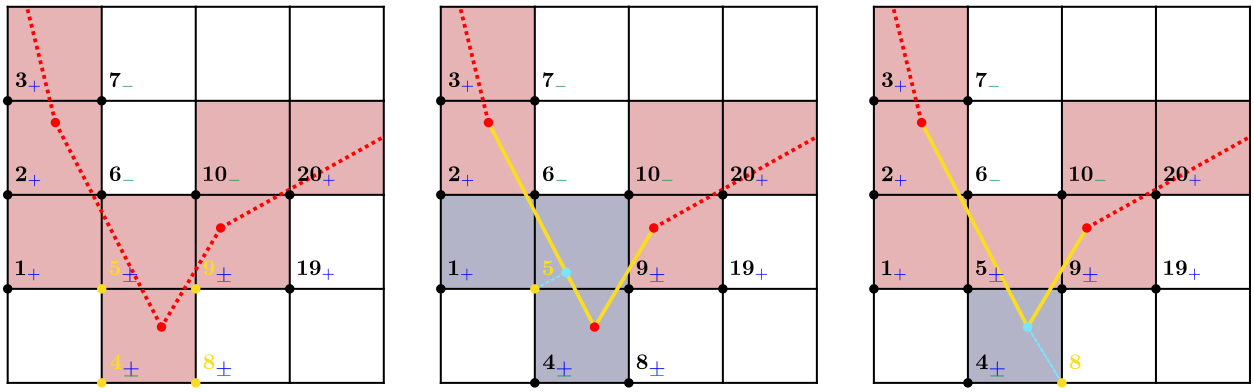


Figure 3.10: **Closest Facet. Left:** The four vertices in yellow all have ambiguous signs. **Middle:** To sign vertex 5, we generate the local patch S_{5^V}, which are the segments shown in yellow. The closest facet (indicated in cyan) lies on a face. **Right:** A similar process is illustrated for vertex 8, but here the closest facet is a vertex.

A discrimination plane is defined by two non-collinear incident edges and it has an empty half-space if all incident faces and edges lie on one side of the plane or on the plane itself. Note that geometries can be constructed at any resolution for which the addition of more triangles in the closest facet detection can result in an incorrect sign evaluated from the above conditions. However, we did not observe this failure in any practical mesh. Even the addition of more triangles was only rarely needed, among the meshes used in the present work, for the mesh of example 3.29 at a few resolutions.

3.6 Interior Extension Region Creation

We grow the volumetric extension $\mathcal{U}$ on its interior boundary (defined by vertices with negative sign) to create the remainder of the volumetric mesh $\mathcal{V}$. We determine where to grow the extension by examining connected components of the background grid defined by its intersections with $\mathcal{S}$. We compute these components using depth-first search (as discussed in Section 3.4.1), where adjacency between nodes in the background grid is defined between edge neighbors not divided by $\mathcal{S}$. We again use CGAL's intersection function from the

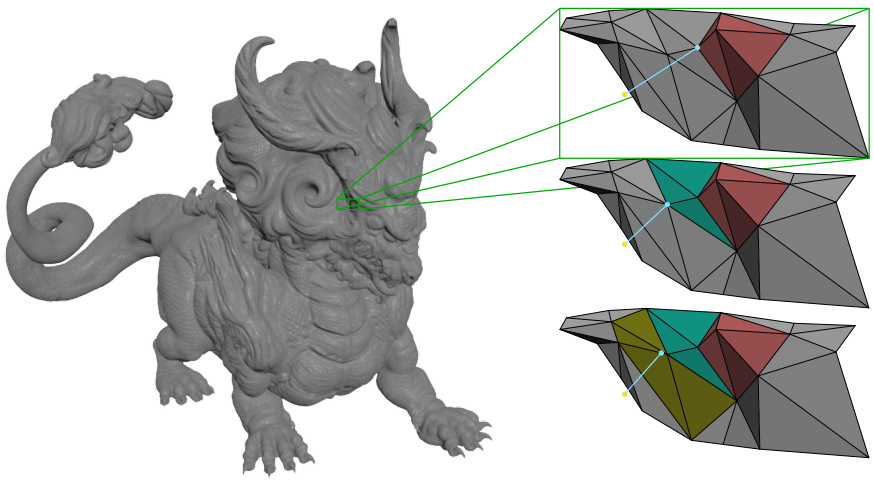


Figure 3.11: **Patch Expansion.** The local patch S_{iv} corresponding to the yellow vertex is shown. The initial patch is indicated in red, and the closest facet is a vertex of the red patch. We add the missing incident triangles (turquoise) and recompute the closest facet. This is again a vertex with incident triangles not in the patch, so we repeat the process (with new triangles in dark yellow). The closest feature is now on an edge, and we proceed to the edge criteria for signing.

2D/3D Linear Kernel to determine whether or not an edge is divided. This is a simplistic criterion which can lead to an over-count in the number of interior regions, as demonstrated in Figure 3.12. A more accurate criteria would use material connectivity determined from the intersection of the surface $\mathcal{S}$ with the relevant background grid cells, similar to the CSG operations in [SDF07]. However, as noted in [LB18] these operations are extremely costly and our approach is robust to over-counting the number of interior regions since they are all merged together appropriately in the later stages of the algorithm.

Each connected component of background grid nodes constitutes a contiguous region. Regions that have a grid node with at least one geometrically coincident vertex in $\mathbf{x}^V$ with negative sign are defined to be interior. Exterior regions, those not containing a grid node with a geometrically coincident vertex in $\mathbf{x}^V$ with negative sign, are discarded. We create at least one hexahedron mesh $\mathcal{V}^{j^I,c}$ for each interior region j^I. Multiple copies of interior

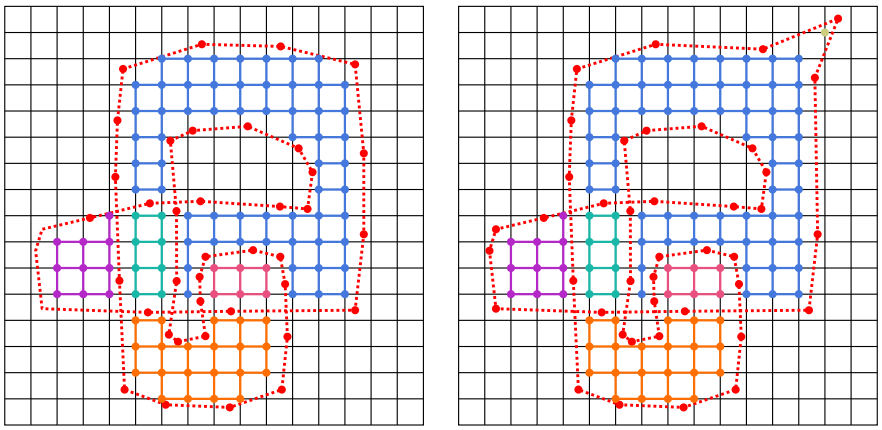


Figure 3.12: **Region Over-Count.** As the process of partitioning the grid only uses connectivity based on grid edges, it is possible for a contiguous region to be split into multiple regions. Shifting some of the vertices of $\mathcal{S}$ on the left results in the geometry on the right, which contains an additional region in the upper right corner since no edge connects this grid node to the larger blue region.

meshes are created near self-intersecting portions of $\mathcal{S}$ since here they represent multiple overlapping portions of the volumetric domain. We illustrate this process in Figure 3.13. We note that as before, each hexahedron mesh $\mathcal{V}^{j^I,c}$ uses the common vertex array $\mathbf{x}^V$.

We determine interior regions j^I that require multiple copies as those with grid nodes that have more than one geometrically coincident vertex in $\mathbf{x}^V$ with negative sign. For these regions, we create a copy $\mathcal{V}^{j^I,c}$ for each connected component c of vertices in $\mathbf{x}^V$ with negative sign that are geometrically coincident with a grid node in the region, as shown in Figure 3.14. Adjacency between these vertices is defined if they are in a common hexahedron in the volumetric extension $\mathcal{U}$. In general, this will be an over-count as multiple connected components may ultimately correspond to the same copy. We note that this process is analogous to the cell creation portion of the method of Li and Barbič [LB18]. They show that in the case of simple immersions, the correct number of copies is equal to the winding number of the region. We do not compute the winding number since our over-count is typically resolved during the merging process described in Section 3.7. However, failure

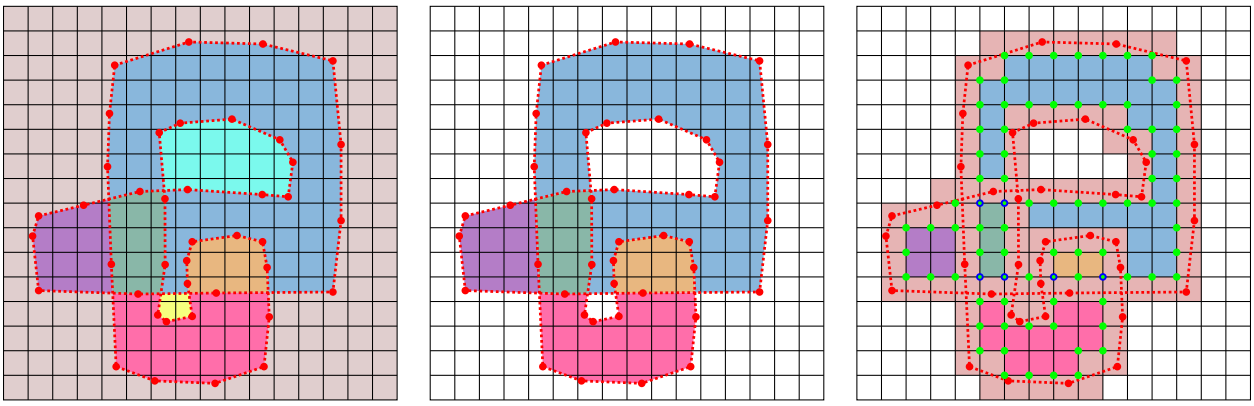


Figure 3.13: **Connected Regions. Left:** The surface partitions the background grid into contiguous regions. **Middle:** The exterior regions are removed. **Right:** The volumetric extension $\mathcal{U}$ is shown, along with the negatively signed vertices in green. Multiple geometrically coincident vertices are indicated using blue circles with green centers.

cases occur when the background uniform grid $\mathcal{G}_{\Delta x}$ cannot resolve thin features or high-curvature in $\mathcal{S}$. In these cases, an over-count that cannot be resolved in the later merging stages occurs. The background grid must be refined to resolve these cases, however using a strategy similar to that of Wang et al. [WJS14] we use a topology-preserving coarsening strategy (see Section 3.8) after the algorithm has run to prevent excessively small element sizes and associated high element counts. We also note again that unlike Li and Barbič [LB18], we cannot handle non-simple immersions.

As with $\mathcal{U}$, we construct the first copy of the hexahedron mesh for each interior region $\mathcal{V}^{j^I,0}$ from precursor hexahedron meshes $\mathcal{V}_{\mathbf{i}}^{j^I,0} = (\mathbf{x}^V, \mathbf{m}_{\mathbf{i}}^{V^{j^I,0}})$. Here $\mathbf{x_i}$ are the grid nodes in region j^I. It should be noted that these are different than the vertices $\mathbf{x}_i^V \in \mathbf{x}^V$ and that $\mathbf{i} = (i_0, i_1, i_2)$ is used to denote the grid multi-index associated with the node. For each $\mathbf{x_i}$, $\mathbf{m}_{\mathbf{i}}^{V^{j^I,0}}$ consists of 8 hexahedra which are geometrically coincident with the 8 local background grid cells incident to $\mathbf{x_i}$. Copies of $\mathbf{x_i}$ and the 26 background grid nodes surrounding $\mathbf{x_i}$ (whether or not they are in region j^I) are introduced into $\mathbf{x}^V$ to achieve this. We again merge these precursors as described in Section 3.4.1 where adjacencies between the vertices of $\mathbf{x}^V$

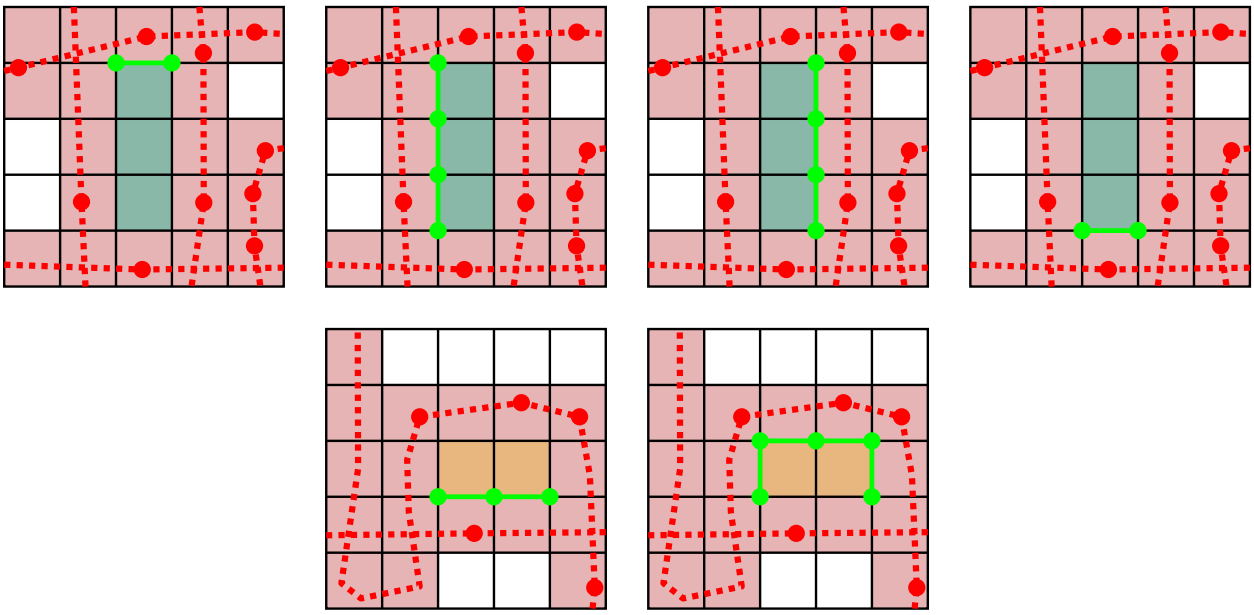


Figure 3.14: **Copy Counting.** The two regions from Figure 3.13 having multiple copies are shown. Each copy is displayed with its corresponding connected component of vertices with negative sign.

are defined as follows. For each pair of grid nodes $\mathbf{x_i}$ and $\mathbf{x_j}$ in region j^I, the geometrically coincident vertices in $\mathbf{x}^V$ corresponding to the hexahedra of $\mathcal{V}_{\mathbf{i}}^{j^I,0}$ and $\mathcal{V}_{\mathbf{j}}^{j^I,0}$ are adjacent if $\mathbf{x_i}$ and $\mathbf{x_j}$ are connected by an edge in $\mathcal{G}_{\Delta x}$ that is not cut by a triangle in $\mathcal{S}$. This edge cut criteria prevents connection between geometrically close but topologically distant features, as illustrated in Figure 3.15. We reemphasize that as described in Section 3.4.1 the final $\mathbf{m}^{V^{j^I,0}}$ is formed by concatenating all of the arrays $\mathbf{m}_{\mathbf{i}}^{V^{j^I,0}}$ (modified to account for merged vertex numbering) and removing any duplicated hexahedra. The remaining copies $\mathcal{V}^{j^I,c}$ are created by duplicating $\mathbf{m}^{V^{j^I,0}}$ with new vertices distinct from those corresponding to $\mathcal{V}^{j^I,0}$ and any other copy.

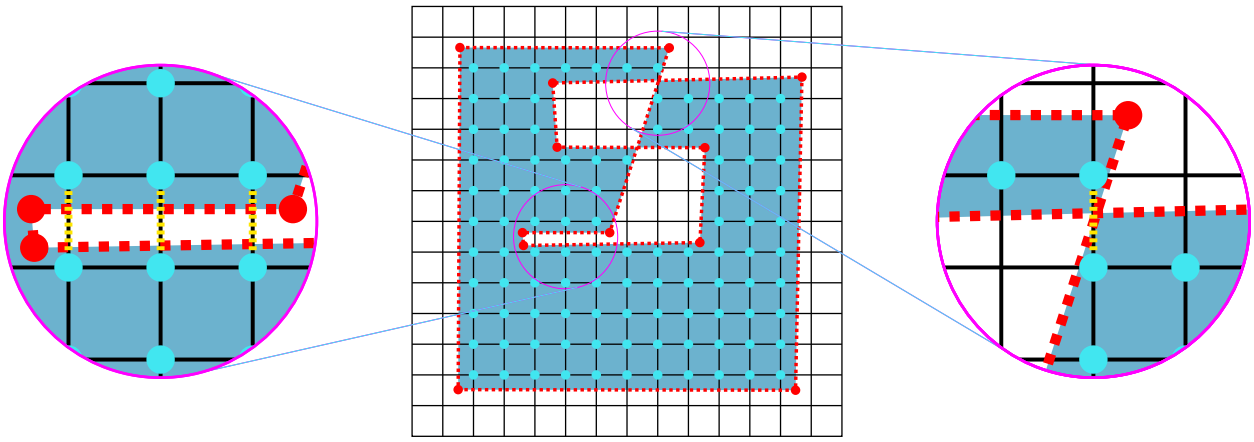


Figure 3.15: **Edge Cut Criterion.** Grid nodes $\mathbf{x_i}$ of a region are shown, along with two examples showing that adjacent grid nodes may have their common edge cut by a triangle (cut edges are indicated by the dashed yellow lines). In this case, adjacencies are not built between the corresponding vertices in $\mathcal{V}_{\mathbf{i}}^{j^I,0}$ to avoid unwanted sewing.

3.7 Interior Extension Region Merging

Having created the interior extensions $\mathcal{V}^{j^I,c}$, the merging of these meshes with the volumetric extension $\mathcal{U}$ and with each other (to account for possible over-counting in their creation) is carried out in multiple steps. We first merge hexahedra from $\mathcal{U}$ into $\mathcal{V}^{j^I,c}$ in a process described below. We then determine which of the interior extensions should merge to each other, using hexahedra from $\mathcal{U}$ which merge into multiple $\mathcal{V}^{j^I,c}$ to generate a list of overlapping hexahedra between meshes of different regions and copies. Next, we use these overlaps to determine which copies of the same region are duplicated and merge the duplicates together. Finally, these overlapping hexahedra are used to define the adjacencies in the final merging process.

3.7.1 Merge With Boundary

Recall from Section 3.6 that in regions with more than one copy, we create a copy $\mathcal{V}^{j^I,c}$ for each connected component c of vertices in $\mathbf{x}^V$ located in region j^I with negative sign. We

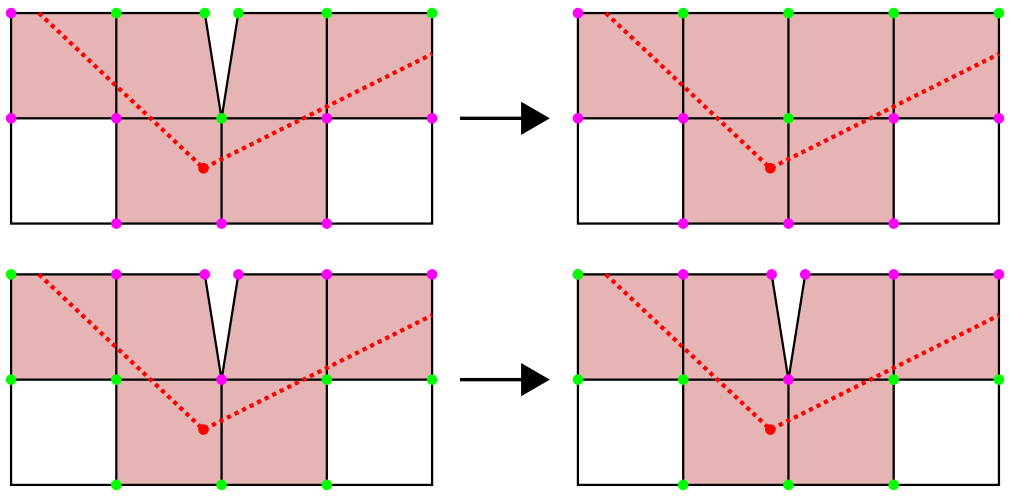


Figure 3.16: **Preliminary Merge.** The construction of the volumetric extension $\mathcal{U}$ may result in geometrically coincident vertices which do not come from topologically distant parts of the mesh. Green vertices have negative signs, while purple vertices have positive sign. **Top:** The process in Section 3.7.1 merges these vertices into a single vertex. **Bottom:** We do not merge coincident positive vertices, to avoid unnecessarily sewing the exterior.

use $\mathcal{C}_c^{j^I}$ to denote the collection of these nodes in the connected component c. For regions with only one copy, $\mathcal{C}_0^{j^I}$ instead denotes the collection of all vertices in $\mathbf{x}^V$ located in region j^I with negative sign, as we do not generate connected components in this case. Not that for these single copy regions, the vertices of $\mathcal{C}_0^{j^I}$ need not be connected (see the geometry of Figure 3.17, where the vertices $\mathcal{C}_0^{j^I}$ are composed of two connected components on the outer and inner boundaries). We merge vertices of $\mathcal{V}^{j^I,c}$ with vertices in $\mathcal{C}_c^{j^I}$ using the merge described in Section 3.4.1. Before this merge, we first perform a preliminary merge of vertices in $\mathcal{C}_c^{j^I}$ which are geometrically coincident. Here, two vertices of $\mathbf{x}^V$ are adjacent if they are geometrically coincident and both in $\mathcal{C}_c^{j^I}$. The effect of this preliminary merge is to close unwanted interior voids without 'sewing' the exterior and without merging topologically distant vertices of $\mathcal{U}$, as shown in Figure 3.16. The merge between the vertices of $\mathcal{V}^{j^I,c}$ and $\mathcal{C}_c^{j^I}$ is then defined by the following adjacency. Vertices of $\mathcal{V}^{j^I,c}$ and $\mathcal{C}_c^{j^I}$ are adjacent if they are geometrically coincident and the vertex of $\mathcal{V}^{j^I,c}$ was created from an interior

connected component of vertices in the $\mathcal{V}_{\mathbf{i}}^{j^I,0}$ that gave rise to $\mathcal{V}^{j^I,c}$ via the merge described in Section 3.6. Here, an interior connected component is one that contains the center vertex (as opposed to one of the surrounding 26 vertices) introduced in the creation of $\mathcal{V}_{\mathbf{j}}^{j^I,0}$ for some grid node $\mathbf{x_j}$ in the region j^I. This requirement effectively means that vertices of $\mathcal{C}_c^{j^I}$ should only merge to the those vertices of $\mathcal{V}^{j^I,c}$ which are actually interior to the region, and not the vertices which are overlapping from a topologically far part of $\mathcal{V}^{j^I,c}$. We illustrate this in Figure 3.17. Note that after this merge has been performed, we update the indices in $\mathcal{C}_c^{j^I}$ accordingly as this set will be used in latter steps of the merging procedure.

We next use a strategy different to that in Section 3.4.1 for merging hexahedral elements in $\mathcal{U}$ to their geometrically coincident counterparts in$\mathcal{V}^{j^I,c}$. This modified merging strategy is designed to prefer the structure of $\mathcal{U}$ over that in $\mathcal{V}^{j^I,c}$. For instance, if two hexahedra of $\mathcal{U}$ are geometrically coincident but share only vertices on one face, then they will still have this connectivity after merging to $\mathcal{V}^{j^I,c}$. We merge the hexahedra in $\mathcal{U}$ incident to the vertices in $\mathcal{C}_c^{j^I}$ to their geometrically coincident counterparts in $\mathcal{V}^{j^I,c}$. Specifically, for each vertex $\mathbf{x}_i^V$ with $i \in \mathcal{C}_c^{j^I}$ and $k_i^{\mathcal{U}} \in \mathcal{I}_i^{\mathcal{U}}$, the hexahedron $\lfloor \frac{k_i^{\mathcal{U}}}{8} \rfloor$ is marked for merging. We denote the collection of hexahedra in $\mathcal{U}$ marked to be merged with their counterparts in copy c of region j^I as $\mathcal{I}_H^{j^I,c}$. Note that it is possible that some hexahedra of $\mathcal{U}$ are not included in any such collection. To perform this modified merging procedure, we first remove hexahedra from $\mathbf{m}^{V^{j^I,c}}$ that are geometrically coincident with a hexahedron from $\mathcal{I}_H^{j^I,c}$ and incident to a vertex in $\mathcal{C}_c^{j^I}$. Note that a hexahedron in $\mathbf{m}^{V^{j^I,c}}$ can only be incident to a node in $\mathcal{C}_c^{j^I}$ after the merge described in the previous paragraph has been completed. Next, copies of the hexahedra in $\mathcal{I}_H^{j^I,c}$ are added to $\mathbf{m}^{V^{j^I,c}}$. The process following the preliminary merge is outlined in Figure 3.18.

3.7.2 Overlap Lists

We next merge differing regions $\mathcal{V}^{j_0^I,c}$ along their appropriately defined common boundaries. The boundary region between any two region copy meshes $\mathcal{V}^{j_0^I,c_0}$ and $\mathcal{V}^{j_1^I,c_1}$ is grown from

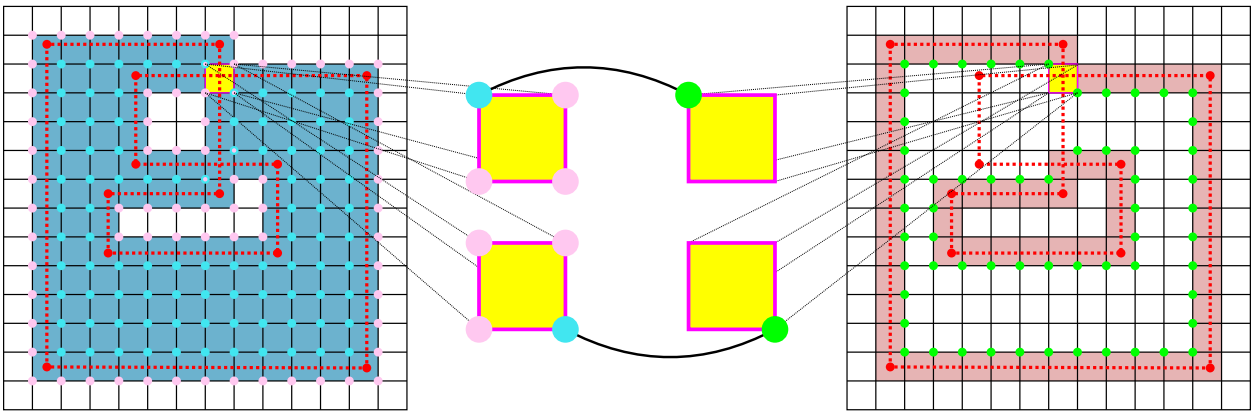


Figure 3.17: **Vertex Adjacency.** The merge process between vertices of $\mathcal{V}^{j^I,c}$ and $\mathcal{C}_c^{j^I}$. For the cell highlighted in yellow, there are 2 hexahedra from $\mathcal{V}^{j^I,c}$ and therefore 4 pairs of geometrically coincident vertices. The two negatively signed vertices (in green) from $\mathcal{C}_c^{j^I}$ are matched to the vertices which came from an interior connected component (marked in cyan) and not the ones which did not (marked in pink).

seeds which we define by hexahedra in the respective meshes that are equal and in $\mathcal{U}$. For example, suppose that $\mathcal{V}^{j_0^I,c_0}$ and $\mathcal{V}^{j_1^I,c_1}$ contain such a hexahedron. In this case there are hexahedra with indices $h_{e_0}^{V^{j_0^I,c_0}}, h_{f_0}^{V^{j_1^I,c_1}} \in \mathbb{N}$ sharing the same vertices as a hexahedron in $\mathcal{U}$ with index $h_{g_0}^U \in \mathbb{N}$ such that

$$m^{V^{j_0^I,c_0}}_{8h_{e_0}^{V^{j_0^I,c_0}}+i^e} = m^{V^{j_1^I,c_1}}_{8h_{f_0}^{V^{j_1^I,c_1}}+i^e} = m^{V^S}_{8h_{g_0}^{V^S}+i^e}, \;\; i^e \in \{0,1,\ldots,7\}\,. \tag{3.1}$$

When these hexahedra exist in two region copies j_0^I, c_0 and j_1^I, c_1 we use the notation $\mathbf{q} = (j_0^I, c_0, j_1^I, c_1)$ to denote a pair of region copies with common boundary (that which will eventually merge). We define $\mathbf{s}_0^{\mathbf{q}} = (h_{e_0}^{V^{j_0^I,c_0}}, h_{f_0}^{V^{j_1^I,c_1}})$ as a seed between the pair of region copies. Furthermore, we use $\mathbf{p}^{\mathbf{q}} = [\mathbf{s}_0^{\mathbf{q}}, \ldots, \mathbf{s}_{N_s^{\mathbf{q}}-1}^{\mathbf{q}}]$ to denote the collection of all such seeds between j_0^I, c_0 and j_1^I, c_1 with $N_s^{\mathbf{q}}$ being the number of seeds. This collection, which we call an overlap list, is grown into the complete overlapping common boundary between j_0^I, c_0 and j_1^I, c_1.

We expand the initial seed collections $\mathbf{p}^{\mathbf{q}}$ by first marking background grid cells geomet-

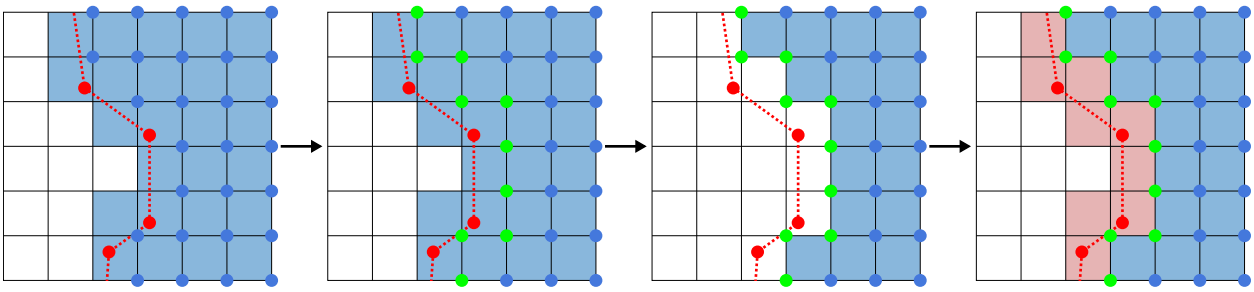


Figure 3.18: **Merge with Boundary.** We illustrate the process of Section 3.7.1 following the preliminary merge of negatively signed vertices. First, specific vertices of $\mathcal{V}^{j^I,c}$ are merged with vertices of $\mathcal{C}_c^{j^I}$. Next, hexahedra to be replaced are removed from the $\mathcal{V}^{j^I,c}$. Finally, copies of hexahedra from $\mathcal{U}$ are added to this mesh.

rically coincident with hexahedra in the seeds as being visited. Then, starting with the seed $\mathbf{s}_0^{\mathbf{q}}$, we compute the neighbor hexahedra of each hexahedron in the seed (the neighbors of a hexahedron are those which share a common vertex). Geometrically coincident neighbors of the two hexahedra in the seed are added to $\mathbf{p}^{\mathbf{q}}$ if the background grid cell to which they are geometrically coincident is unvisited. We then mark the cell as visited, and continue until every seed has been processed in this way. At the end of this expansion, $\mathbf{p}^{\mathbf{q}}$ is a list of overlapping hexahedra that will be used to sew the regions together. We illustrated this process in Figure 3.19.

3.7.3 Deduplication

As mentioned in Section 3.6, the number of copies is generally an over count. We use the overlap lists $\mathbf{p}^{\mathbf{q}}$ to deduce which copies c of a region j^I are redundant. For each hexahedron h_e^S in $\mathcal{U}$, we create a list of hexahedra from geometrically coincident counterparts in interior region copies. This list is formed by considering each pair $\mathbf{q}$: if either hexahedron in a seed of $\mathbf{p}^{\mathbf{q}}$ is a copy of h_e^S (i.e. it uses the same vertices in $\mathbf{x}^S$ as in Equation (3.1)), both hexahedra in the seed are added to the list associated with h_e^S. Note that while the hexahedron pairs of the initial seeds in $\mathbf{p}^{\mathbf{q}}$ are both copies of hexahedra from $\mathcal{U}$ in accordance with Equation (3.1),

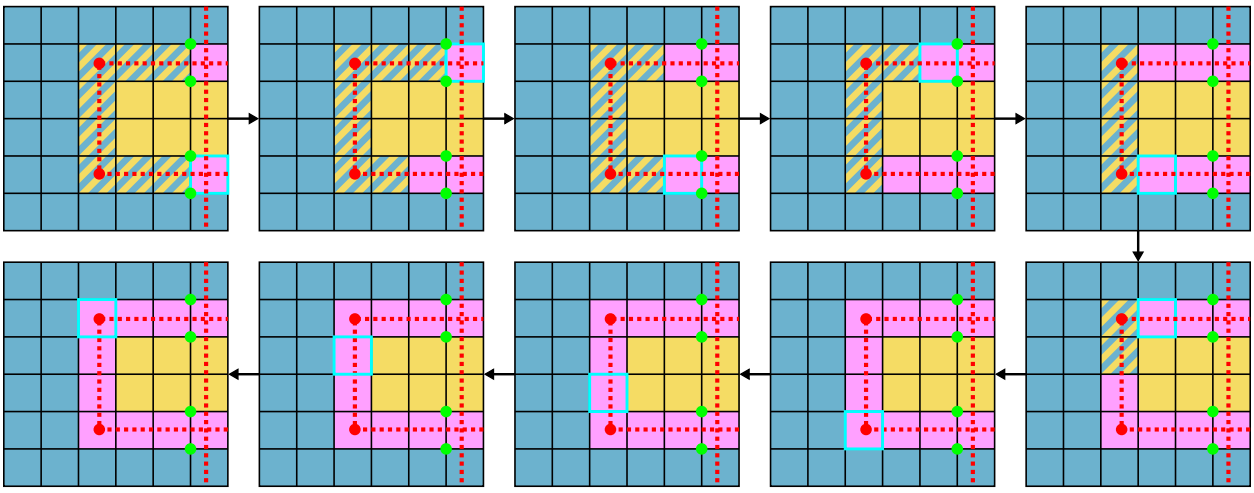


Figure 3.19: **Overlap Lists.** A closeup of the overlap region from the geometry of Figure 3.17 is shown here. At the upper left, the seeds for the overlap between the two copies are shown in purple, as well as the incident negative vertices (green) to the seeds from each copy. At each step, the current seed is marked with a cyan border. New geometrically coincident neighbors of the seed hexahedra are then added in the next step. When all seeds have been traversed, the process stops.

subsequent seeds added during the overlap process may have both, one, or neither hexahedra equal to copies of hexahedra from $\mathcal{U}$. Should any list for any hexahedron h_e^S in $\mathcal{U}$ contain hexahedra from multiple copies c_0 and c_1 of the same region j^I, copies c_0 and c_1 are considered to be redundant duplicates of each other. Redundant copies are merged using the process of Section 3.7.1. This process is shown in Figure 3.20.

For each region, we compute connected components of its copies using duplication as the notion of adjacency. For each connected component of copies, we take the copy with the smallest index c_i as the representative copy. However, this copy's mesh only has the vertices of the component c_i. Likewise, only copies of the hexes in $\mathcal{I}_H^{c_i}$ are in $\mathcal{V}^{j^I,c_i}$. We remedy this by repeating the merge with boundary process of Section 3.7.1 on updated data. Specifically, we replace the connected component c_i of vertices with the union of all components c_j for copies in the connected component of copies. We then form an updated collection of incident

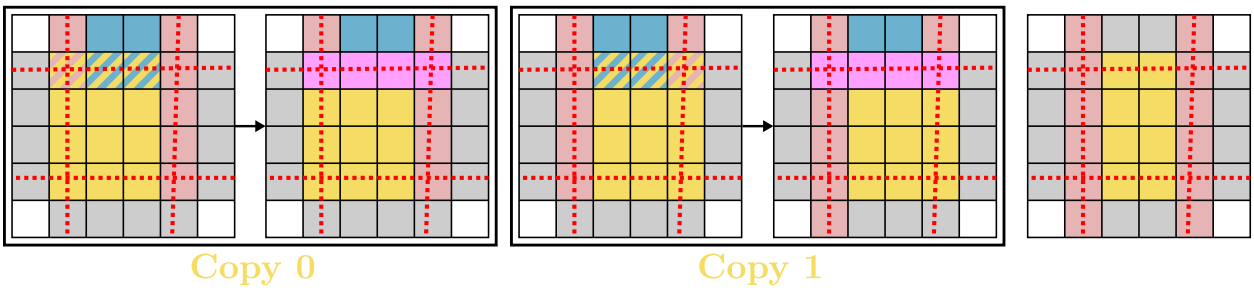


Figure 3.20: **Deduplication.** We show two of the four copies of the central region (yellow), corresponding to the right and left segments of $\mathcal{U}$. Each of copies 0 and 1 create an overlap list with the upper region (blue). The overlap list for copy 0 creates a pair between a non-boundary yellow hexahedron and a boundary hexahedron from the blue region. This boundary hexahedron is in a pair with a boundary hexahedron of copy 1, allowing us to deduce that copies 0 and 1 of the yellow region are duplicates. We then repeat the boundary merge process to create a deduplicated copy with complete boundary information.

hexahedra $\mathcal{I}_H^{c_i}$ before repeating the boundary merge process. Finally, we update the overlap lists. Any overlap list corresponding to a duplicated copy is recreated using the minimum representative in place of the original copy to account for updated hexahedron ordering. Redundant overlap lists resulting from this update are then discarded.

3.7.4 Final Merge

We now merge the vertices of $\mathbf{x}^V$ using the pattern of Section 3.4.1 with adjacencies defined by the overlap lists. For each seed $\mathbf{s}$ in an overlap list, the geometrically coincident nodes of the two hexahedra in $\mathbf{s}$ are considered adjacent. We then create the final mesh $\mathcal{V}$ by combining all of the arrays $\mathbf{m}^{V^{J^I,c}}$ from copies which are either the minimum representative, or not duplicated. Recall from Section 3.4.1 that some hexahedra of $\mathcal{U}$ are not copied into any copy's mesh. We add all such hexahedra to $\mathcal{V}$ to guarantee that $\mathcal{U}$ is contained in this final mesh, completing the interior extension region merging process.

3.8 Coarsening

Our method requires high resolution (small Δx) background grids for high-curvature/detailed surfaces. We provide a topology-aware coarsening strategy to provide user control over the final volumetric mesh resolution/element counts. After the hexhedron mesh $\mathcal{V}$ is created, we coarsen the underlying grid by doubling Δx. We then create a maximal coarse mesh $\mathcal{M}$ based on the fine mesh $\mathcal{V}$. For each index m_j^V in $\mathcal{V}$, we define the initial connectivity for $\mathcal{M}$ as $m_j^M = j$. We then bin the center of each fine hexahedron $h^M \in \mathbb{N}^{N_e^M}$ into the coarsened grid and keep track of its multi-dimensional grid index $\mathbf{i}^{h^M}$. We initialize the position array $\mathbf{x}^M$ for $\mathcal{M}$ from the coarse grid cell corners of cell $\mathbf{i}^{h^M}$. Specifically, for each hexahedron in h^M in $\mathcal{M}$ we define $\mathbf{x}^M_{8h^M+i^e} = \mathbf{x}^{2\Delta x}_{\mathbf{i}^{h^M}} + \mathbf{o}_{i^e}$ where $\mathbf{o}_{i^e}$ is an offset from the coarse cell center $\mathbf{x}^{2\Delta x}_{\mathbf{i}^{h^M}}$ to the eight respective corners of the coarse grid cell $\mathbf{i}^{h^M}$. To build the final coarsened mesh, we merge portions of the maximal coarse mesh using Section 3.4.1 where adjacencies are defined from a hexahedron-wise notion of connectivity. Two maximal coarse hexahedra h_0^M and h_1^M are connected if their corresponding fine hexahedra $h_0^V = h_0^M$ and $h_1^V = h_1^M$ share a face $\mathbf{f}_i^V = \left[f_{i0}^V, f_{i1}^V, f_{i2}^V, f_{i3}^V\right] \in \mathbb{N}^4$ in $\mathcal{V}$. We define two types of connection: totally connected and partially connected. Maximal coarse hexahedra are totally connected if they have the same coarse grid index $\mathbf{i}^{h_0^M} = \mathbf{i}^{h_1^M}$ and their corresponding fine hexahedra h_0^V and h_1^V are not geometrically coincident. Maximal coarse hexahedra are partially connected if they are connected but are not totally connected. We define vertex adjacency from our notions of hexahedron connectivity. If two hexahedra h_0^M and h_1^M in the maximal coarse mesh are totally connected, then their eight respective geometrically coincident vertices are defined to be adjacent, i.e. vertex $m^M_{8h_0^M+i^e}$ is adjacent to vertex $m^M_{8h_1^M+i^e}$, $0 \leq i^e < 8$. If they are partially connected, then their corresponding fine hexahedra h_0^V, h_1^V share a face $\mathbf{f}_i^V = \left[f_{i0}^V, f_{i1}^V, f_{i2}^V, f_{i3}^V\right]$. We then identify an analogous face in each of h_0^V and h_1^V which we define in terms of the indices $k^V_{0\alpha}, k^V_{1\alpha}$, $\alpha \in \{0, 1, 2, 3\}$. Only the vertices corresponding to

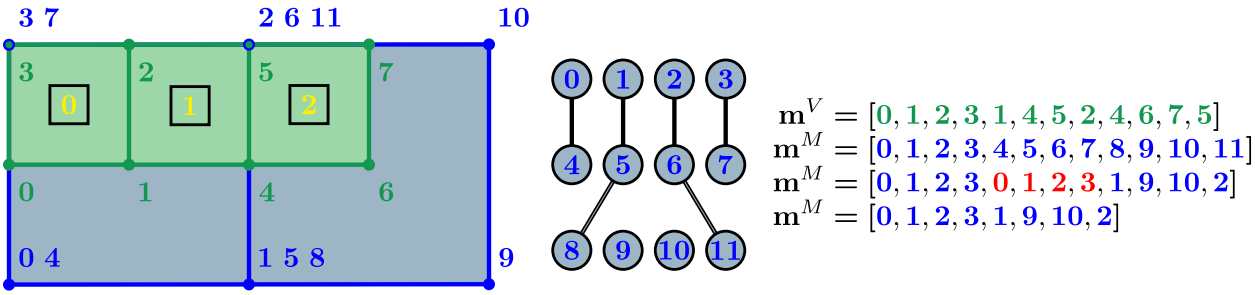


Figure 3.21: **Coarsening.** An example of fine mesh connections. Hexahedra 0 and 1 are totally connected, while hexahedra 1 and 2 are connected by a face. After merging the vertices of the coarse mesh (blue), the duplicated hexahedron (indicated in red) is removed.

the analogous face are defined to be adjacent

$$m^M_{8h^M_0+k^V_{0\alpha}} = m^M_{8h^M_1+k^V_{1\alpha}}, \ \alpha \in \{0, 1, 2, 3\}. \tag{3.2}$$

There are two cases that define the analogous face. First, if the fine hexahedron counterparts h^V_0, h^V_1 are geometrically coincident, then the analogous face is the one on the analogous side of the coarse hexahedron. If they are not geometrically coincident, then the analogous face is the one geometrically coincident with the fine face defined from $\mathbf{f}^V_i$. The general coarsening procedure is illustrated in Figure 3.21.

3.9 Hexahedron Mesh To Tetrahedron Mesh Conversion

We design a topologically-aware BCC-based approach for the creation of a tetrahedron mesh $\mathcal{T}$ from the hexahedron mesh $\mathcal{V}$. We initialize the particle array for the tetrahedron mesh $\mathbf{x}^T$ to be the same as $\mathbf{x}^V$, but we add a new vertex in the center of each hexahedron and each boundary face. Tetrahedra are computed from the faces in the mesh $\mathcal{V}$. Normally a face in $\mathcal{V}$ would have one (boundary face) or two (interior face) incident hexahedra. However, since $\mathcal{V}$ is comprised of many geometrically coincident hexahedra there are more cases. We classify them as: standard boundary face (one incident hexahedraon), standard interior face (two non-geometrically coincident incident hexahedra), non-standard interior (more than two in-

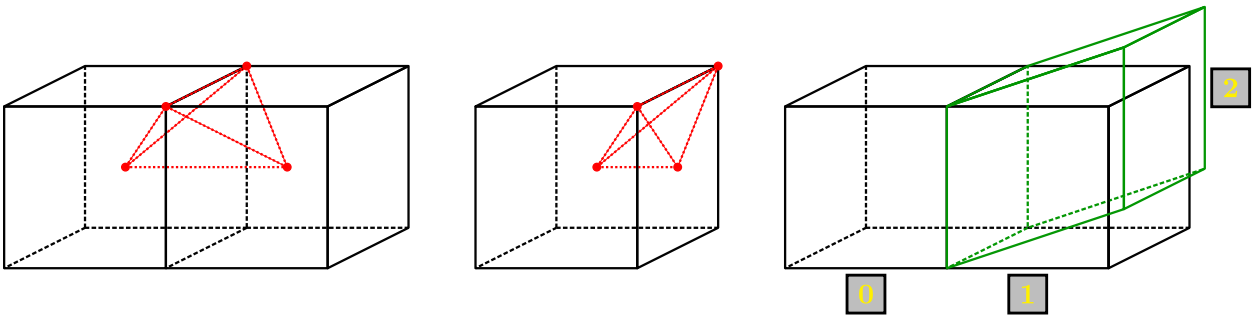
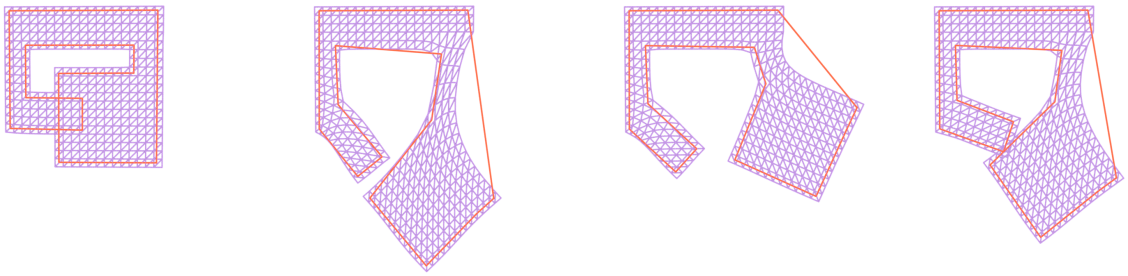


Figure 3.22: **Hexahedra Tetrahedralization**. **Left:** a standard interior face in $\mathcal{V}$. The centers of the two incident hexahedra are combined with two face vertices to form the tetrahedra (red). **Middle:** a standard boundary face uses a face center instead of the missing incident hexahedron center. **Right:** a non-standard interior face is shown. The right-most incident hexahedra are geometrically coincident. We form hexahedra pairs/faces (0,1), (0,2) and treat them respectively as standard interior, as in the left-most image.

cident hexahedra, some geometrically coincident and some not geometrically coincident) and non-standard boundary (more than one incident hexahderon, all geometrically coincident). Each face contributes four tetrahedra to $\mathcal{T}$ in the case of standard boundary and standard interior faces. The tetrahedra consist of two vertices from the face and the cell centers on either side of the face in the case of standard interior faces. In the case of standard boundary faces, the face center is used in place of the second hexahedron center. For non-standard interior faces, we take all pairs of non-geometrically coincident incident hexahedra and add tetrahedra as if their common face was a standard interior face. For non-standard boundary faces, tetrahedra are added for each incident hexahedron as if it were incident to a standard boundary face. We illustrate this procedure in Figure 3.22.

3.10 Examples

We consider a variety of examples in both two and three dimensions. To illustrate the capabilities of the final mesh connectivites, we treat the objects as deformable solids and



(a) Frame 0 (b) Frame 11 (c) Frame 27 (d) Frame 60

Figure 3.23: **2D Simple Overlap.** A self-intersecting shape is suspended from a ceiling. The geometry deforms under gravity, and both sides freely move regardless of the initial overlap.

run a finite element (FEM) simulation [SB12]. Performance statistics for the 3D examples are presented in Table 3.1. All experiments were run on a workstation with a single Intel® Core™ i9-10980XE CPU at 3.00GHz.

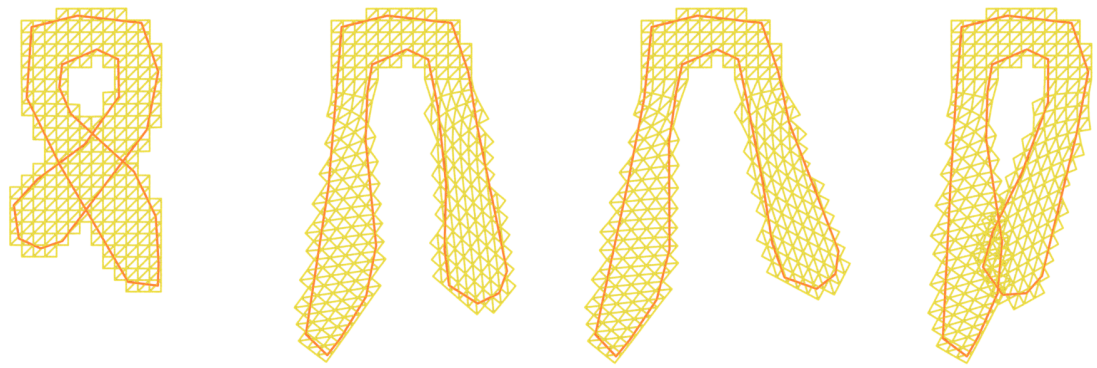
3.10.1 2D Examples

3.10.1.1 Single Overlap

Figure 3.23 shows a deformable FEM simulation using a volumetric mesh produced by our algorithm. As evidenced by the geometry's ability to separate and freely move, our algorithm produces a mesh that properly resolves the single self-intersection present in the initial configuration.

3.10.1.2 Ribbon

Our algorithm can also handle more complex self-intersections. In Figure 3.24, one end of a ribbon shape passes through the other, partitioning the surface into several components. These intersections are successfully resolved, and the mesh is allowed to move as in the



(a) Frame 0 (b) Frame 14 (c) Frame 59 (d) Frame 74

Figure 3.24: **2D Ribbon.** A ribbon with a more complicated initial self-intersection is also treated properly by our method.

previous example.

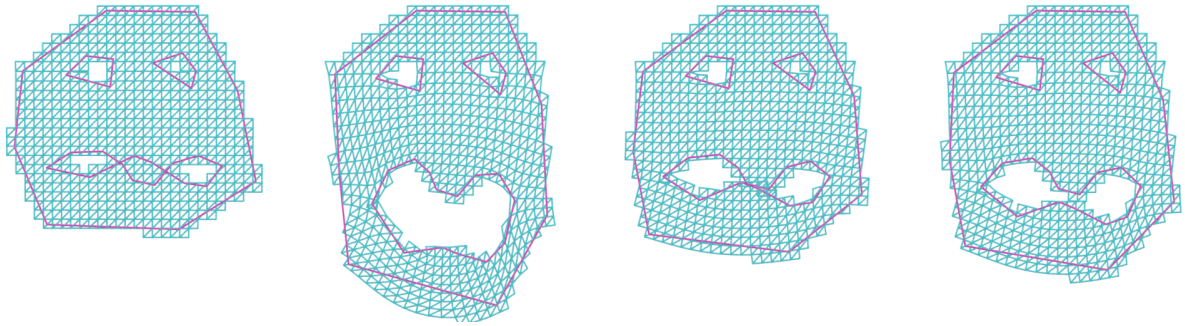
3.10.1.3 Face

Figure 3.25 demonstrates a similar scenario. In this case, the lips of the face geometry initially overlap; and, as an added challenge, the boundary of the input geometry consists of multiple disconnected components. Our method successfully treats cases like these by design.

3.10.2 3D Examples

3.10.2.1 Two Boxes & Simple Overlap

We begin our 3D examples by demonstrating that our algorithm is able to quickly generate consistent meshes for simple self-intersecting geometries. In Figure 3.26, basic hand-made geometries are allowed to separate and unfurl from their initial self-intersecting states. The two boxes in the left-hand side of each subfigure were meshed using a background grid



(a) Frame 0 (b) Frame 8 (c) Frame 21 (d) Frame 92

Figure 3.25: **2D Face.** A face with multiple boundary components and initially self-intersecting lips is successfully animated.

resolution of $66 \times 64 \times 86$ cells and $\Delta x = .00955671$, taking 2.80219s to generate the resulting 256,368 hexahedra in the output mesh. The simple overlapping shape in the right-hand side of each subfigure was meshed using a grid with $194 \times 64 \times 194$ cells and $\Delta x = .00328125$, resulting in 1,606,296 hexahedra in the output mesh.

3.10.2.2 Double Möbius

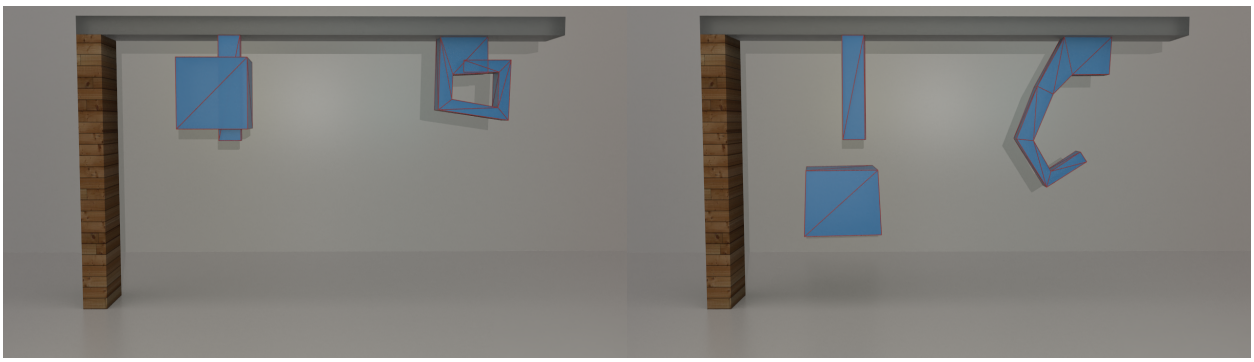
Figure 3.27 shows two Möbius-strip-like geometries[1] falling and separating under the effects of gravity, despite substantial intersections at the start of the simulation. This example was run using a background grid with $294 \times 288 \times 64$ cells and a Δx of 0.0347391. The resulting hexahedron mesh has 903,653 elements. Generating the volumetric mesh using our algorithm takes 33.6324s.

We also consider repeating this example at multiple spatial resolutions in order to demonstrate the effect of resolution on the quality of meshing results (see Figure 3.28). The coarsest grid (corresponding to the leftmost meshes in each subfigure) is $21 \times 19 \times 5$ with $\Delta x = 0.556$. An intermediate grid resolution of $39 \times 37 \times 9$ cells with $\Delta x = 0.278$ corresponds to the

[1]"Mobius Bangle" by Creative_Hacker is licensed under CC BY 4.0.

Example	Grid dim.	Relative Δx	# Hex	Time (s)
Two Boxes	66×64×86	0.0158730	256368	2.80219
Simple Overlap	194×64×194	0.0158730	1606296	24.0179
Double Möbius	294×288×64	0.0158730	903653	33.6324
Twin Bunnies	162×166×128	0.00787402	1525821	31.1815
Dragon	512×690×520	0.00195313	20110457	303.301
Fancy Ball	130×132×128	0.00787402	515400	25.8388
Head	512×830×718	0.00195313	62444819	839.951
Sacht	52×104×42	0.0243902	135736	6.97091

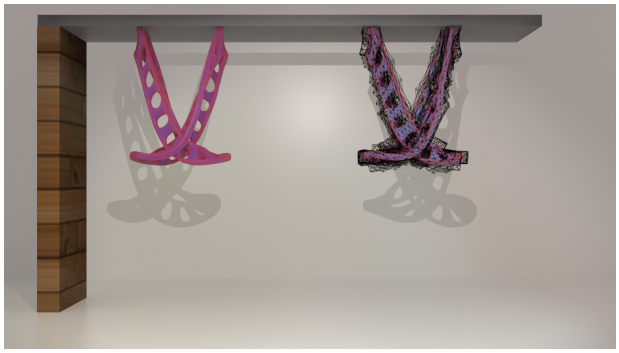
Table 3.1: **Volumetric Mesh Generation Times for Various 3D Examples.** All times are in seconds and represent the total runtime of the algorithm. Relative Δx is ratio of Δx to the shortest side length of the bounding box.



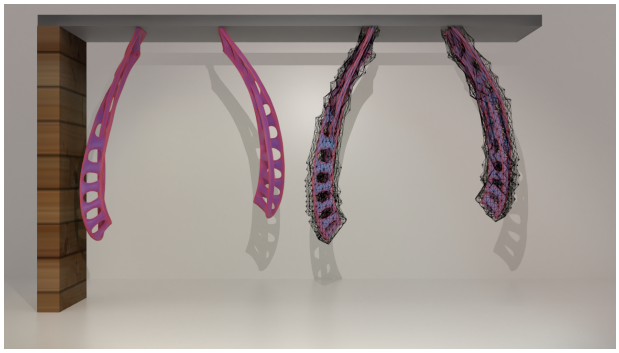
(a) Frame 4 (b) Frame 33

Figure 3.26: **3D Simple Overlap.** Simple self-intersecting 3D geometries are able to separate and unfurl with our algorithm.

(a) Frame 0

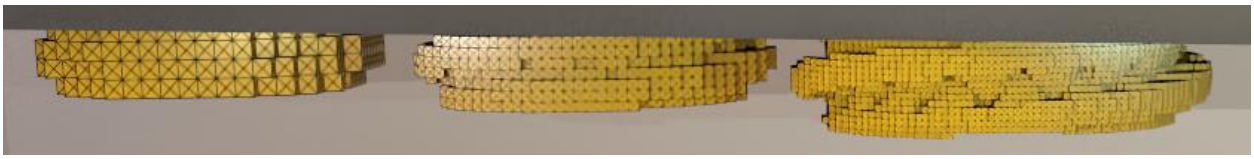


(b) Frame 44

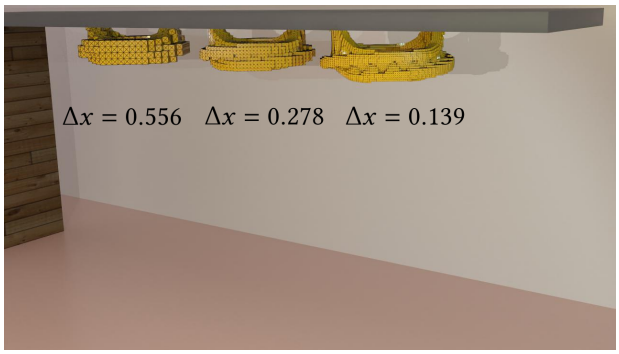


(c) Frame 110

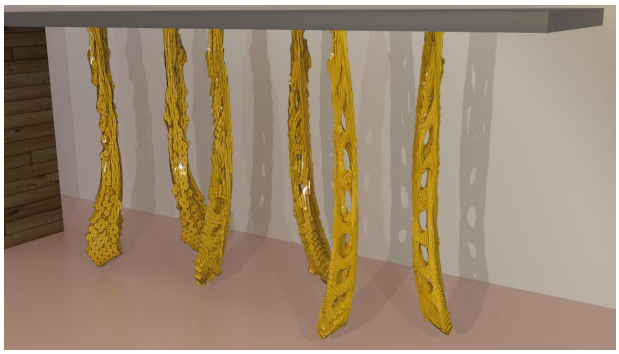
Figure 3.27: **Double Möbius.** Two intersecting Möbius-strip-like geometries (pink) naturally fall and separate under our method. The associated hexahedron meshes are shown in the right half of each frame.



(a) Frame 0



(b) Frame 16



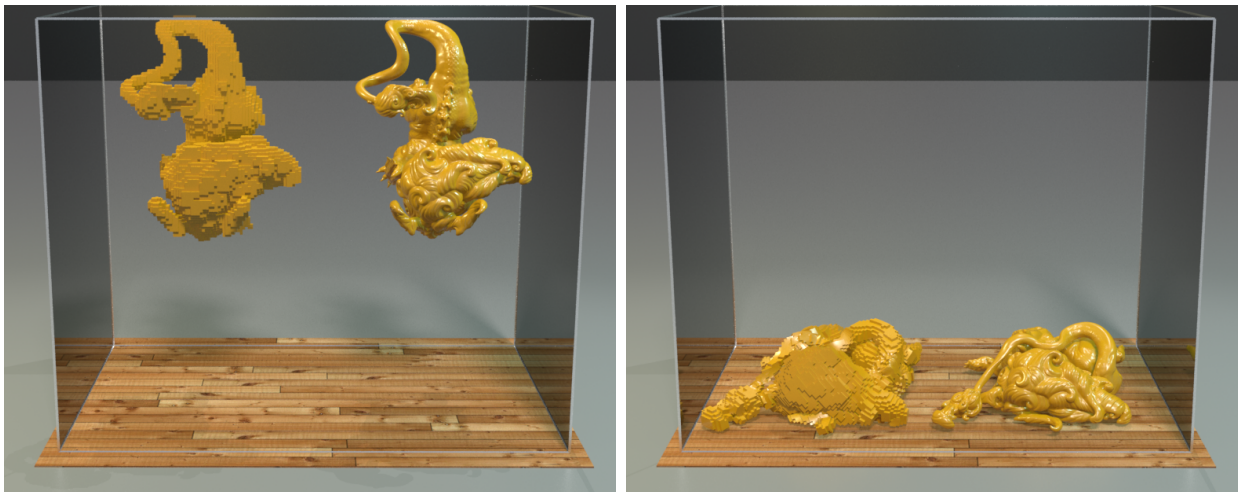
(c) Frame 84

Figure 3.28: **Double Möbius Refinement.** Running the example shown in Figure 3.27 at different spatial resolutions. In each frame, from left to right, the background grids have $\Delta x = 0.556$, 0.278, and 0.139.

middle meshes in each subfigure. The rightmost meshes in each subfigure come from using a grid with $75 \times 73 \times 17$ cells with $\Delta x = 0.139$. Proper separation is achieved at all three of these tested resolutions, and in particular, our algorithm performs quite well on this example even at extremely low spatial resolution.

3.10.2.3 Twin Bunnies

Another standard example is the Stanford bunny. Figure 3.3 demonstrates that two almost completely overlapping bunny meshes can naturally separate under our method. No issues are encountered as different segments of the bunnies pass through one another. This example uses a grid resolution of $162 \times 166 \times 128$ cells with $\Delta x = 0.0203027$, resulting in a mesh with 1,525,821 hexahedra.



(a) Frame 0 (b) Frame 300

Figure 3.29: **Dragon.** A complex mesh of a dragon is allowed to fall under gravity. The left-hand side of each subfigure shows the deforming mesh we generate, and each right-hand side shows the corresponding surface mesh.

3.10.2.4 Dragon

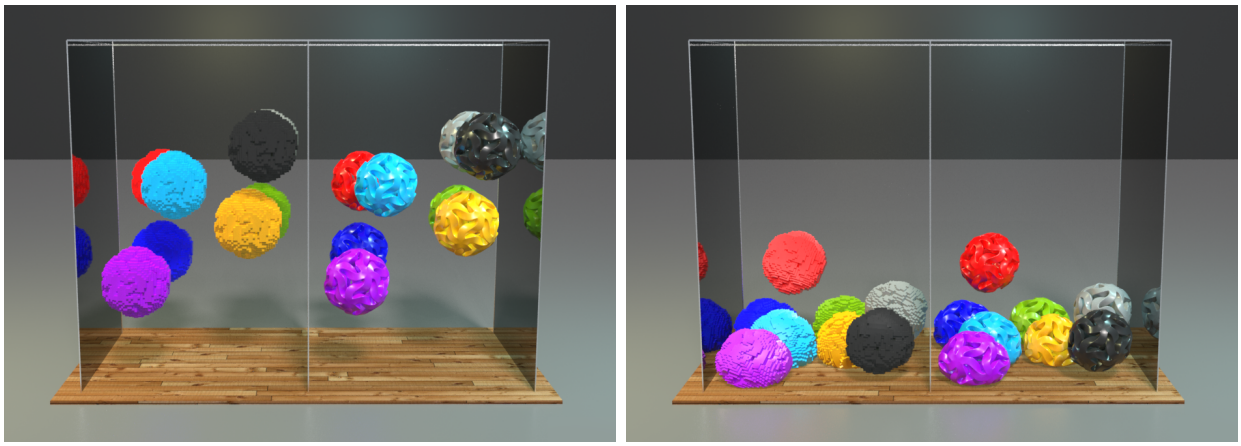
The most complicated geometry we test our method on is the dragon[2] shown in Figure 3.29 (and also shown in Figure 3.11). Adequate resolution is required in order to resolve all the fine-scale features of this mesh; accordingly, we use a grid resolution of $512 \times 690 \times 520$ cells with $\Delta x = 0.0708709$. Our final mesh, generated in five minutes, contains just over 20 million hexahedra.

3.10.2.5 Fancy Ball

Figure 3.30 shows another interesting case where several ball-like geometries[3] deform and collide after being meshed with our algorithm. Each ball has a number of thin cuts and

[2]"Asian Dragon" by Lalo-Bravo.

[3]"Abstract object" by sonic art.



(a) Frame 20 (b) Frame 80

Figure 3.30: **Fancy Ball.** Several ball-like geometries with intricate slices and holes are successfully meshed with our algorithm and then deform and collide under an FEM simulation.

fine-scale features, which our algorithm is able to resolve using a grid with $130 \times 132 \times 128$ cells and $\Delta x = 2.82671$. The 515,400 resulting hexahedra are generated in 25.8388s.

3.10.2.6 Head

Modeling of the human body often gives rise to self-intersection. This is particularly common in the faces, where lip geometries often self-intersect. To that end, we consider a real-world head geometry in Figure 3.4. Note that the lips separate effectively. This example results in a volumetric mesh with over 62 million elements, using a background grid resolution of $512 \times 830 \times 718$ cells and $\Delta x = 0.000501962$. Generating the hexahedron mesh takes 839.951s.

3.10.2.7 Collection

Various objects from 3D examples are dropped in a tank in Figure 3.31. The objects naturally deform and collide without meshing or simulation issues.

(a) Frame 80 (b) Frame 200

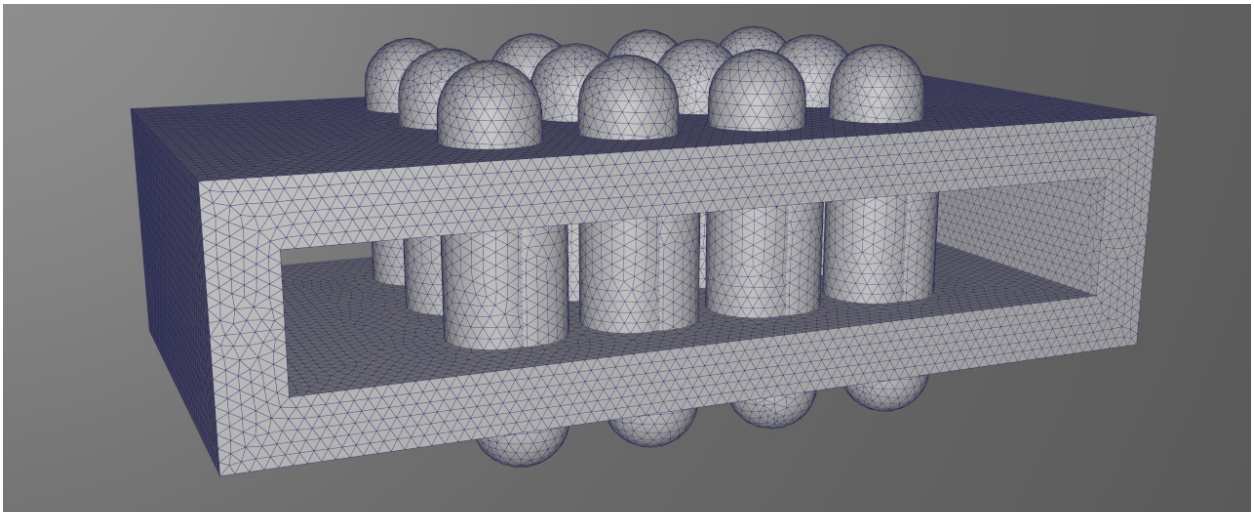
Figure 3.31: **3D Mesh Collection.** We simulated dropping our 3D examples into a box with a FEM sim.

3.10.2.8 Sacht et al. Mesh

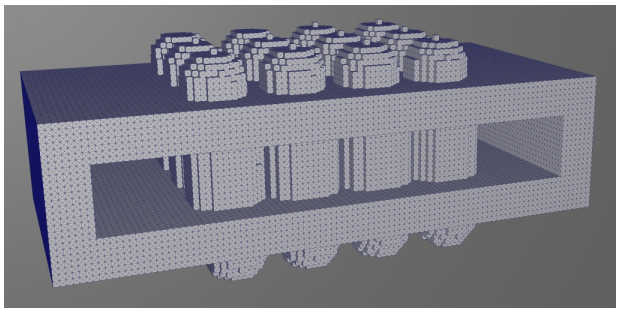
Finally, we demonstrate that our method, like that of Li and Barbič [LB18], can successfully resolve the self intersections of the geometry shown in Figure 3.32 that is not supported by the method of Sacht et al. [SJP13]. In [SJP13], the bristles in this geometry get locked by the surrounding torus. However, both our method and [LB18] properly resolve all self-intersections. Of note, for a similar number of output mesh elements (135,736 vs. 112,554), our method runs noticeably faster than that of Li and Barbič [LB18] (6.97s vs. 22.5s).

3.11 Discussion and Limitations

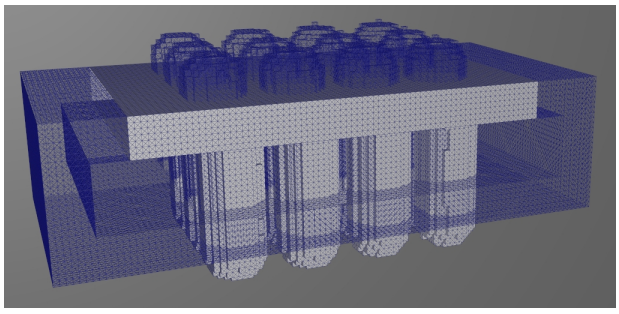
Our method has various limitations, most of which are attributed to our reduced use of exact/adaptive precision arithmetic. The most prominent limitations of our approach are in the types of input surface mesh $\mathcal{S}$ that we support. Fine-scale features, e.g., thin parallel sheets, can cause negatively signed vertices to be located in regions of the grid corresponding



(a) Original surface mesh



(b) Output tetrahedral mesh



(c) A contiguous subsection

Figure 3.32: **Sacht Geometry.** Our method can successfully resolve the self intersecting geometry proposed in [SJP13]. Here, we visualize the surface of the output after conversion to a tetrahedral mesh. We emphasize a subsection of the mesh on the right sub-figure; the bristles are attached to the correct parts of the torus and are not connected to the bristles from the opposite side of the torus.

to an incorrect region. This may result in exterior regions erroneously generating copies, or interior regions creating extra copies which will not be correctly merged or deduplicated. In these pathological cases, the output mesh will have undesirable extraneous collections of hexahedra. Such cases result from a background grid which is too coarse relative to the size of polygonal faces in the finer regions. Hence, we resolve this by iteratively halving the cell width until fine scale features can be accurately resolved. We note that such a heuristic strategy may result in an undesirable level of refinement. However, our coarsening approach is designed to mitigate this. Even using added resolution and subsequent coarsening, our methodological simplifications prevent us from handling certain classes of cases that Li and Barbič [LB18] can handle, e.g., we cannot resolve non-simple immersions. It would be interesting to investigate whether our minimal-exact-arithmetic approach could be extended to handle non-simple immersions as well. A further failure case arises from one surface mesh being fully contained within the volume defined by a larger surface mesh. Our method will generally fail to create enough copies of the smaller volume. This is not a serious limitation, however, as our focus is on self-intersection and such situations can easily be avoided by translating separate meshes by a sufficient amount. Future work includes improvements to the algorithm to handle known pathological cases without the need for refinement and subsequent coarsening, as well as improved detection mechanisms for such cases. In particular, additional use of exact arithmetic in the form of segment-triangle intersections should allow for the detection of fine scale signing failures. It should be possible to combine such a detection mechanism with an adaptive refinement/coarsening scheme to create an intermediate step which eliminates the need for heuristic global refinement and allows for graceful failure (i.e. when a refinement limit is reached).

Lastly, Figure 3.2 illustrates an interesting case which neither our approach, that of Li and Barbič [LB18] nor that of Sacht et al. [SJP13] can handle. In this case, which is common near e.g. elbows and even shoulders in an upper torso, a portion of the domain overlaps in such a way that $\boldsymbol{\phi}_{\tilde{S}}^{S}$ must have negative Jacobian determinant in some regions.

Our approach returns a mesh for this case, but it does not properly copy the overlap region and one of the two copies that would be required is rejected. I.e. our approach does not give a result consistent with creating a mesh in $\tilde{\mathcal{S}}^V$ and pushing it forward under $\boldsymbol{\phi}^S_{\tilde{S}}$. In Li and Barbič [LB18], this is noted as a case for which an immersion does not exist and Sacht et al. [SJP13] explicitly require the Jacobian determinant of $\boldsymbol{\phi}^S_{\tilde{S}}$ to be non-negative. However, this is a commonly occurring case which would be beneficial to resolve.

CHAPTER 4

Other Contributions

This chapter is organized into two sections. In Section 4.1, we compute the eigenvalues and eigenvectors of the Hessian of the surface tension potential energy density defined in [HGM20]. This is a modified version of material from the supplementary material of [HGM20].

In Section 4.2, we derive the expression for the inertia tensor from [JSS15]. While we provide definitions of relevant terms as necessary, a general summary of the APIC method has been omitted. We direct the reader to [JST16] for a summary of APIC and the material point method in general.

4.1 Eigenstructure of the Hessian of a Surface Tension Energy Term

We define the following surface tension energy density as a function of the deformation gradient $\hat{\mathbf{F}}$ (see Section 1.2.1.2; we use the hat notation to match [HGM20]) and area weighted normal $d\mathbf{A}$:

$$\Psi(\hat{\mathbf{F}}, d\mathbf{A}) = k\|J\hat{\mathbf{F}}^{-T}d\mathbf{A}\|, \tag{4.1}$$

where $J = \det\hat{\mathbf{F}}$ and k is the surface tension coefficient. We now compute the Hessian and its structure in both 2D and 3D.

4.1.1 2D

In 2D, we may write $J\hat{\mathbf{F}}^{-T}$ in terms of the Levi-Civita symbol as follows:

$$J\hat{F}^{-T}_{\alpha\beta} = \varepsilon_{\alpha\gamma}\varepsilon_{\beta\delta}\hat{F}_{\gamma\delta}. \tag{4.2}$$

Then we can write $J\hat{\mathbf{F}}^{-T}d\mathbf{A}$ as

$$J\hat{F}^{-T}_{\alpha\beta}dA_\beta = \varepsilon_{\alpha\gamma}\varepsilon_{\beta\delta}dA_\beta\hat{F}_{\gamma\delta}. \tag{4.3}$$

We then define the third order tensor $\mathbf{C}$ by

$$C_{\alpha\gamma\delta} = \varepsilon_{\alpha\gamma}\varepsilon_{\beta\delta}dA_\beta. \tag{4.4}$$

With this notation,

$$J\hat{F}^{-T}_{\alpha\beta}dA_\beta = C_{\alpha\gamma\delta}\hat{F}_{\gamma\delta}. \tag{4.5}$$

We denote this by $\mathbf{C}\hat{\mathbf{F}}$. The tensor $\mathbf{C}$ is a function of $d\mathbf{A}$. Note that the energy written in this notation is $\Psi(\hat{\mathbf{F}}, d\mathbf{A}) = k\|\mathbf{C}\hat{\mathbf{F}}\|$.

4.1.1.1 Gradient

With this notation, the gradient of the energy is given by

$$\begin{aligned}\frac{\partial\Psi}{\partial\hat{F}_{\alpha\beta}} &= \frac{k}{2}\frac{1}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\partial}{\partial\hat{F}_{\alpha\beta}}\left(C_{\gamma\delta\epsilon}\hat{F}_{\delta\epsilon}C_{\gamma\zeta\eta}\hat{F}_{\zeta\eta}\right)\\ &= k\frac{C_{\gamma\delta\epsilon}\hat{F}_{\delta\epsilon}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\partial}{\partial\hat{F}_{\alpha\beta}}\left(C_{\gamma\zeta\eta}\hat{F}_{\zeta\eta}\right)\\ &= k\frac{C_{\gamma\alpha\beta}C_{\gamma\delta\epsilon}\hat{F}_{\delta\epsilon}}{\|\mathbf{C}\hat{\mathbf{F}}\|}.\end{aligned} \tag{4.6}$$

4.1.1.2 Hessian

The Hessian is given by

$$\begin{aligned}\frac{\partial^2\Psi}{\partial\hat{F}_{\gamma\delta}\hat{F}_{\alpha\beta}} &= k\frac{C_{\epsilon\alpha\beta}C_{\epsilon\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\partial\hat{F}_{\zeta\eta}}{\partial\hat{F}_{\gamma\delta}} - \frac{k}{2}\frac{C_{\epsilon\alpha\beta}C_{\epsilon\zeta\eta}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|^3}\frac{\partial}{\partial\hat{F}_{\gamma\delta}}\left(C_{\theta\iota\kappa}\hat{F}_{\iota\kappa}C_{\theta\lambda\mu}\hat{F}_{\lambda\mu}\right)\\ &= k\frac{C_{\epsilon\alpha\beta}C_{\epsilon\gamma\delta}}{\|\mathbf{C}\hat{\mathbf{F}}\|} - k\frac{C_{\epsilon\alpha\beta}C_{\epsilon\zeta\eta}\hat{F}_{\zeta\eta}C_{\theta\gamma\delta}C_{\theta\iota\kappa}\hat{F}_{\iota\kappa}}{\|\mathbf{C}\hat{\mathbf{F}}\|^3}.\end{aligned} \tag{4.7}$$

We can simplify these expressions using the property $\varepsilon_{\alpha\beta}\varepsilon_{\alpha\gamma} = \delta_{\beta\gamma}$. First, note that

$$\begin{aligned}C_{\epsilon\alpha\beta}C_{\epsilon\gamma\delta} &= \varepsilon_{\epsilon\alpha}\varepsilon_{\zeta\beta}dA_\zeta\varepsilon_{\epsilon\gamma}\varepsilon_{\eta\delta}dA_\eta\\ &= \delta_{\alpha\gamma}\varepsilon_{\zeta\beta}dA_\zeta\varepsilon_{\eta\delta}dA_\eta.\end{aligned} \tag{4.8}$$

Then, define $d\mathbf{A}^\perp$ by

$$dA^\perp_\beta = \varepsilon_{\zeta\beta}dA_\zeta. \tag{4.9}$$

This notation is motivated by the observation that $d\mathbf{A}^\perp \cdot d\mathbf{A} = \varepsilon_{\zeta\beta}dA_\zeta dA_\beta = 0$. With this notation,

$$C_{\epsilon\alpha\beta}C_{\epsilon\gamma\delta} = \delta_{\alpha\gamma}dA^\perp_\beta dA^\perp_\delta. \tag{4.10}$$

The gradient and Hessian in this notation are

$$\frac{\partial\Psi}{\partial\hat{F}_{\alpha\beta}} = k\frac{\hat{F}_{\alpha\epsilon}dA^\perp_\epsilon dA^\perp_\beta}{\|\mathbf{C}\hat{\mathbf{F}}\|} \tag{4.11}$$

and

$$\frac{\partial^2\Psi}{\partial\hat{F}_{\gamma\delta}\hat{F}_{\alpha\beta}} = k\frac{\delta_{\alpha\gamma}dA^\perp_\beta dA^\perp_\delta}{\|\mathbf{C}\hat{\mathbf{F}}\|} - k\frac{\hat{F}_{\alpha\eta}dA^\perp_\eta dA^\perp_\beta\hat{F}_{\gamma\kappa}dA^\perp_\kappa dA^\perp_\delta}{\|\mathbf{C}\hat{\mathbf{F}}\|^3}. \tag{4.12}$$

4.1.1.3 Eigenvalues and Eigenvectors

By inspection, the $dA^\perp_\delta$ in both terms suggest that tensors of the form

$$\delta\hat{F}_{\gamma\delta} = u_\gamma dA_\delta \tag{4.13}$$

are in the kernel of the Hessian:

$$\frac{\partial^2\Psi}{\partial\hat{F}_{\gamma\delta}\hat{F}_{\alpha\beta}}u_\gamma dA_\delta = k\frac{\delta_{\alpha\gamma}u_\gamma dA^\perp_\beta}{\|\mathbf{C}\hat{\mathbf{F}}\|}dA^\perp_\delta dA_\delta - k\frac{\hat{F}_{\alpha\eta}dA^\perp_\eta dA^\perp_\beta \hat{F}_{\gamma\kappa}dA^\perp_\kappa u_\gamma}{\|\mathbf{C}\hat{\mathbf{F}}\|^3}dA^\perp_\delta dA_\delta$$
$$= 0. \tag{4.14}$$

So we obtain two orthogonal eigenvectors with associated eigenvalue 0 by choosing any two nonzero orthogonal vectors $\mathbf{u}_0$ and $\mathbf{u}_1$ and forming the tensor products $\mathbf{u}_0 d\mathbf{A}^T$ and $\mathbf{u}_1 d\mathbf{A}^T$.

For the remaining two eigenvectors, consider tensors of the form

$$\delta\hat{F}_{\gamma\delta} = v_\gamma dA^\perp_\delta. \tag{4.15}$$

For the first term, note that

$$k\frac{\delta_{\alpha\gamma}dA^\perp_\beta dA^\perp_\delta}{\|\mathbf{C}\hat{\mathbf{F}}\|}v_\gamma dA^\perp_\delta = k\frac{\|d\mathbf{A}^\perp\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}v_\alpha dA^\perp_\beta = k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}v_\alpha dA^\perp_\beta, \tag{4.16}$$

where we have used the fact that $\|d\mathbf{A}^\perp\| = \|d\mathbf{A}\|$. So the first term acts as a uniform scaling on the subspace spanned by vectors of the form $\mathbf{v}(d\mathbf{A}^\perp)^T$. So eigenvectors of the second term in this subspace will also be eigenvectors of the first term, and hence eigenvectors of the Hessian. Note that the second term is rank 1 and has the form

$$-k\frac{\hat{F}_{\alpha\eta}dA^\perp_\eta dA^\perp_\beta \hat{F}_{\gamma\kappa}dA^\perp_\kappa dA^\perp_\delta}{\|\mathbf{C}\hat{\mathbf{F}}\|^3} = -k\frac{(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\alpha dA^\perp_\beta (\hat{\mathbf{F}}d\mathbf{A}^\perp)_\gamma dA^\perp_\delta}{\|\mathbf{C}\hat{\mathbf{F}}\|^3}, \tag{4.17}$$

so it has eigenvectors $(\hat{\mathbf{F}}d\mathbf{A}^\perp)(d\mathbf{A}^\perp)^T$ and $(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp(d\mathbf{A}^\perp)^T$, where $(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp$ is defined in the same way as $d\mathbf{A}^\perp$, i.e. $(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp_\alpha = \varepsilon_{\beta\alpha}(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\beta$, and is likewise orthogonal to $\hat{\mathbf{F}}d\mathbf{A}^\perp$. The eigenvalues of the second term corresponding to these eigenvectors are, respectively, $-k\frac{\|\hat{\mathbf{F}}d\mathbf{A}^\perp\|^2\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|^3}$ and 0.

Taking into account the eigenvalues of the first term,

$$(\hat{\mathbf{F}}d\mathbf{A}^\perp)(d\mathbf{A}^\perp)^T \text{ and } (\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp(d\mathbf{A}^\perp)^T \tag{4.18}$$

are eigenvectors of the Hessian with eigenvalues

$$k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|} - k\frac{\|\hat{\mathbf{F}}d\mathbf{A}^\perp\|^2\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|^3} = k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left(1 - \frac{\|\hat{\mathbf{F}}d\mathbf{A}^\perp\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|^2}\right) \tag{4.19}$$

and

$$k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}. \tag{4.20}$$

The first eigenvalue here can be simplified:

$$\begin{aligned}
\|\mathbf{C}\hat{\mathbf{F}}\|^2 &= C_{\alpha\beta\gamma}\hat{F}_{\beta\gamma}C_{\alpha\delta\epsilon}\hat{F}_{\delta\epsilon} \\
&= \varepsilon_{\alpha\beta}\varepsilon_{\zeta\gamma}dA_\zeta\hat{F}_{\beta\gamma}\varepsilon_{\alpha\delta}\varepsilon_{\eta\epsilon}dA_\eta\hat{F}_{\delta\epsilon} \\
&= \varepsilon_{\alpha\beta}\hat{F}_{\beta\gamma}dA^\perp_\gamma\varepsilon_{\alpha\delta}\hat{F}_{\delta\epsilon}dA^\perp_\epsilon \\
&= \delta_{\beta\delta}(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\beta(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\delta \\
&= \|\hat{\mathbf{F}}d\mathbf{A}^\perp\|^2.
\end{aligned} \tag{4.21}$$

Hence, the first eigenvalue here is also 0.

We then have the following eigenvalues and corresponding orthonormal eigenvectors:

Eigenvalue	Eigenvectors
$k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}$	$\frac{(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{(d\mathbf{A}^\perp)^T}{\|d\mathbf{A}\|}$
0	$\frac{\hat{\mathbf{F}}d\mathbf{A}^\perp}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{(d\mathbf{A}^\perp)^T}{\|d\mathbf{A}\|}$
	$\mathbf{u}_0\frac{d\mathbf{A}^T}{\|d\mathbf{A}\|}$
	$\mathbf{u}_1\frac{d\mathbf{A}^T}{\|d\mathbf{A}\|}$

where $\mathbf{u}_0$ and $\mathbf{u}_1$ is any orthonormal basis for $\mathbb{R}^2$. So the Hessian is positive semi-definite in 2D. Note that the gradient can be written as

$$\frac{\partial\Psi}{\partial\hat{\mathbf{F}}} = k\frac{\hat{\mathbf{F}}d\mathbf{A}^\perp}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left(d\mathbf{A}^\perp\right)^T, \tag{4.22}$$

and is therefore in the kernel of the Hessian.

4.1.1.4 Behavior of Ψ under perturbations

Consider the function

$$h(\epsilon,\nu) = \Psi\left(\hat{\mathbf{F}} + \epsilon c_1\hat{\mathbf{F}}d\mathbf{A}^\perp(d\mathbf{A}^\perp)^T + \nu c_2(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp(d\mathbf{A}^\perp)^T\right), \tag{4.23}$$

where c_1 and c_2 are normalizing constants. Then, its square is

$$\begin{aligned}
h(\epsilon,\nu)^2 &= k^2\varepsilon_{\alpha\beta}\left(\hat{F}_{\beta\gamma}+\epsilon c_1(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\beta d\mathbf{A}^\perp_\gamma+\nu c_2(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp_\beta d\mathbf{A}^\perp_\gamma\right)d\mathbf{A}^\perp_\gamma \\
&\quad * \varepsilon_{\alpha\delta}\left(\hat{F}_{\delta\epsilon}+\epsilon c_1(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\delta d\mathbf{A}^\perp_\epsilon+\nu c_2(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp_\delta d\mathbf{A}^\perp_\epsilon\right)d\mathbf{A}^\perp_\epsilon \\
&= k^2\delta_{\beta\delta}\left(\hat{F}_{\beta\gamma}+\epsilon c_1(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\beta d\mathbf{A}^\perp_\gamma+\nu c_2(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp_\beta d\mathbf{A}^\perp_\gamma\right)d\mathbf{A}^\perp_\gamma \\
&\quad * \left(\hat{F}_{\delta\epsilon}+\epsilon c_1(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\delta d\mathbf{A}^\perp_\epsilon+\nu c_2(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp_\delta d\mathbf{A}^\perp_\epsilon\right)d\mathbf{A}^\perp_\epsilon \\
&= k^2\delta_{\beta\delta}\left((\hat{\mathbf{F}}d\mathbf{A}^\perp)_\beta+\epsilon c_1\|d\mathbf{A}\|^2(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\beta+\nu c_2\|d\mathbf{A}\|^2(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp_\beta\right) \\
&\quad * \left((\hat{\mathbf{F}}d\mathbf{A}^\perp)_\delta+\epsilon c_1\|d\mathbf{A}\|^2(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\delta+\nu c_2\|d\mathbf{A}\|^2(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp_\delta\right) \\
&= k^2\delta_{\beta\delta}\left((1+\epsilon c_1\|d\mathbf{A}\|^2)(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\beta+\nu c_2\|d\mathbf{A}\|^2(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp_\beta\right) \\
&\quad * \left((1+\epsilon c_1\|d\mathbf{A}\|^2)(\hat{\mathbf{F}}d\mathbf{A}^\perp)_\delta+\nu c_2\|d\mathbf{A}\|^2(\hat{\mathbf{F}}d\mathbf{A}^\perp)^\perp_\delta\right) \\
&= k^2\left[(1+\epsilon c_1\|d\mathbf{A}\|^2)^2\|\mathbf{C}\hat{\mathbf{F}}\|^2+\nu^2c_2^2\|\mathbf{C}\hat{\mathbf{F}}\|^2\|d\mathbf{A}\|^4\right]. \qquad (4.24)
\end{aligned}$$

Since c_1 and c_2 are normalizing, constants, $c_1 = c_2 = (\|\mathbf{C}\hat{\mathbf{F}}\|\|d\mathbf{A}\|)^{-1}$, and hence

$$\begin{aligned}
h(\epsilon,\nu)^2 &= k^2\left[(\|\mathbf{C}\hat{\mathbf{F}}\|+\|d\mathbf{A}\|\epsilon)^2+\|d\mathbf{A}\|^2\nu^2\right] \\
&= k^2\left[\left(\frac{1}{k}\Psi(\hat{\mathbf{F}})+\|d\mathbf{A}\|\epsilon\right)+\|d\mathbf{A}\|^2\nu^2\right]. \qquad (4.25)
\end{aligned}$$

So

$$h(\epsilon,\nu) = k\sqrt{\left(\frac{1}{k}\Psi(\hat{\mathbf{F}})+\|d\mathbf{A}\|\epsilon\right)+\|d\mathbf{A}\|^2\nu^2}. \qquad (4.26)$$

4.1.2 3D

In 3D, we may likewise write $J\hat{\mathbf{F}}^{-T}$ in terms of the Levi-Civita symbol:

$$J\hat{F}^{-T}_{\alpha\beta} = \frac{1}{2}\varepsilon_{\alpha\gamma\delta}\varepsilon_{\beta\epsilon\zeta}\hat{F}_{\gamma\epsilon}\hat{F}_{\delta\zeta}. \qquad (4.27)$$

Then we write $J\hat{\mathbf{F}}d\mathbf{A}$ as

$$J\hat{F}_{\alpha\beta}dA_\beta = \frac{1}{2}\varepsilon_{\alpha\gamma\delta}\varepsilon_{\beta\epsilon\zeta}\hat{F}_{\gamma\epsilon}\hat{F}_{\delta\zeta}dA_\beta. \qquad (4.28)$$

We then define the fifth order tensor $\mathbf{C}$ by

$$C_{\alpha\gamma\epsilon\delta\zeta} = \frac{1}{2}\varepsilon_{\alpha\gamma\delta}\varepsilon_{\beta\epsilon\zeta}dA_\beta. \tag{4.29}$$

With this notation,

$$J\hat{F}_{\alpha\beta}dA_\beta = C_{\alpha\gamma\epsilon\delta\zeta}\hat{F}_{\gamma\epsilon}\hat{F}_{\delta\zeta}. \tag{4.30}$$

We again denote this by $\mathbf{C}\hat{\mathbf{F}}$ for convenience, even though this is really the tensor $\mathbf{C}$ contracted with two copies of $\hat{\mathbf{F}}$, not one. Once again, we have $\Psi(\hat{\mathbf{F}}, d\mathbf{A}) = k\|\mathbf{C}\hat{\mathbf{F}}\|$.

4.1.2.1 Gradient

The gradient in 3D is given by

$$\begin{aligned}
\frac{\partial\Psi}{\partial\hat{F}_{\alpha\beta}} &= \frac{k}{2}\frac{1}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\partial}{\partial\hat{F}_{\alpha\beta}}\left(C_{\gamma\delta\epsilon\zeta\eta}\hat{F}_{\delta\epsilon}\hat{F}_{\zeta\eta}C_{\gamma\theta\iota\kappa\lambda}\hat{F}_{\theta\iota}\hat{F}_{\kappa\lambda}\right) \\
&= k\frac{C_{\gamma\delta\epsilon\zeta\eta}\hat{F}_{\delta\epsilon}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\partial}{\partial\hat{F}_{\alpha\beta}}\left(C_{\gamma\theta\iota\kappa\lambda}\hat{F}_{\theta\iota}\hat{F}_{\kappa\lambda}\right) \\
&= k\frac{C_{\gamma\delta\epsilon\zeta\eta}\hat{F}_{\delta\epsilon}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left[C_{\gamma\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda} + C_{\gamma\theta\iota\alpha\beta}\hat{F}_{\theta\iota}\right] \\
&= 2k\frac{C_{\gamma\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}C_{\gamma\delta\epsilon\zeta\eta}\hat{F}_{\delta\epsilon}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \\
&= 2k\frac{C_{\gamma\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}(\mathbf{C}\hat{\mathbf{F}})_\gamma}{\|\mathbf{C}\hat{\mathbf{F}}\|},
\end{aligned} \tag{4.31}$$

where we have used the fact that $C_{\gamma\theta\iota\alpha\beta} = C_{\gamma\alpha\beta\theta\iota}$, which follows from our definition of $\mathbf{C}$ and the properties of the Levi-Civita symbol:

$$C_{\gamma\theta\iota\alpha\beta} = \frac{1}{2}\varepsilon_{\gamma\theta\alpha}\varepsilon_{\delta\iota\beta}dA_\delta = \frac{1}{2}\varepsilon_{\gamma\alpha\theta}\varepsilon_{\delta\beta\iota}dA_\delta = C_{\gamma\alpha\beta\theta\iota}. \tag{4.32}$$

4.1.2.2 Hessian

The Hessian is given by

$$\begin{aligned}
\frac{\partial^2\Psi}{\partial\hat{F}_{\gamma\delta}\hat{F}_{\alpha\beta}} &= 2k\frac{C_{\theta\alpha\beta\kappa\lambda}C_{\theta\iota\epsilon\zeta\eta}\hat{F}_{\iota\epsilon}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\partial\hat{F}_{\kappa\lambda}}{\partial\hat{F}_{\gamma\delta}} + 2k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\partial}{\partial\hat{F}_{\gamma\delta}}\left(C_{\theta\iota\epsilon\zeta\eta}\hat{F}_{\iota\epsilon}\hat{F}_{\zeta\eta}\right) \\
&\quad - k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}C_{\theta\iota\epsilon\zeta\eta}\hat{F}_{\iota\epsilon}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\partial}{\partial\hat{F}_{\gamma\delta}}\left(C_{\mu\nu\xi\rho\sigma}\hat{F}_{\nu\xi}\hat{F}_{\rho\sigma}C_{\mu\tau\upsilon\phi\chi}\hat{F}_{\tau\upsilon}\hat{F}_{\phi\chi}\right) \\
&= 2k\frac{C_{\theta\alpha\beta\gamma\delta}(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|} + 4k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}C_{\theta\gamma\delta\zeta\eta}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \\
&\quad - 2k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}C_{\theta\iota\epsilon\zeta\eta}\hat{F}_{\iota\epsilon}\hat{F}_{\zeta\eta}C_{\mu\nu\xi\rho\sigma}\hat{F}_{\nu\xi}\hat{F}_{\rho\sigma}}{\|\mathbf{C}\hat{\mathbf{F}}\|^3}\frac{\partial}{\partial\hat{F}_{\gamma\delta}}\left(C_{\mu\tau\upsilon\phi\chi}\hat{F}_{\tau\upsilon}\hat{F}_{\phi\chi}\right) \\
&= 2k\frac{C_{\theta\alpha\beta\gamma\delta}(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|} + 4k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}C_{\theta\gamma\delta\zeta\eta}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \\
&\quad - 4k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}C_{\theta\iota\epsilon\zeta\eta}\hat{F}_{\iota\epsilon}\hat{F}_{\zeta\eta}C_{\mu\gamma\delta\phi\chi}\hat{F}_{\phi\chi}C_{\mu\nu\xi\rho\sigma}\hat{F}_{\nu\xi}\hat{F}_{\rho\sigma}}{\|\mathbf{C}\hat{\mathbf{F}}\|^3} \\
&= 2k\frac{C_{\theta\alpha\beta\gamma\delta}(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|} + 4k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}C_{\theta\gamma\delta\zeta\eta}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \\
&\quad - 4k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}(\mathbf{C}\hat{\mathbf{F}})_\theta C_{\mu\gamma\delta\phi\chi}\hat{F}_{\phi\chi}(\mathbf{C}\hat{\mathbf{F}})_\mu}{\|\mathbf{C}\hat{\mathbf{F}}\|^3}.
\end{aligned} \tag{4.33}$$

4.1.2.3 Eigenvalues and Eigenvectors

First consider tensors of the form

$$\partial\hat{F}_{\gamma\delta} = u_\gamma dA_\delta, \tag{4.34}$$

for arbitrary nonzero $\mathbf{u} \in \mathbb{R}^3$. Then,

$$C_{\theta\alpha\beta\gamma\delta}u_\gamma dA_\delta = \frac{1}{2}\varepsilon_{\theta\alpha\gamma}\varepsilon_{\epsilon\beta\delta}dA_\epsilon u_\gamma dA_\delta = \frac{1}{2}\varepsilon_{\theta\alpha\gamma}u_\gamma\varepsilon_{\epsilon\beta\delta}dA_\delta dA_\epsilon = \frac{1}{2}\varepsilon_{\theta\alpha\gamma}u_\gamma(d\mathbf{A}\times d\mathbf{A})_\beta = 0. \tag{4.35}$$

Likewise, since $C_{\theta\gamma\delta\zeta\eta} = C_{\theta\zeta\eta\gamma\delta}$, every term of the Hessian contains a summation of this form. Thus,

$$\frac{\partial^2\Psi}{\partial\hat{F}_{\gamma\delta}\partial\hat{F}_{\alpha\beta}}u_\gamma dA_\delta = 0 \tag{4.36}$$

for arbitrary $\mathbf{u}$. So we obtain three eigenvectors $\mathbf{u}_0 d\mathbf{A}^T$, $\mathbf{u}_1 d\mathbf{A}^T$, and $\mathbf{u}_2 d\mathbf{A}^T$ for three orthogonal vectors $\mathbf{u}_0$, $\mathbf{u}_1$, and $\mathbf{u}_2$.

To find more eigenvalues and eigenvectors, we split the Hessian into a sum of two tensors. We then look for eigenvectors of one and verify that they are eigenvectors of the other, and therefore eigenvectors of the Hessian. Define

$$\partial^2 \Psi^1_{\alpha\beta\gamma\delta} = 2k \frac{C_{\theta\alpha\beta\gamma\delta}(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|} \tag{4.37}$$

and

$$\partial^2 \Psi^2_{\alpha\beta\gamma\delta} = 4k \frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda} C_{\theta\gamma\delta\zeta\eta}\hat{F}_{\zeta\eta}}{\|\mathbf{C}\hat{\mathbf{F}}\|} - 4k \frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}(\mathbf{C}\hat{\mathbf{F}})_\theta C_{\mu\gamma\delta\phi\chi}\hat{F}_{\phi\chi}(\mathbf{C}\hat{\mathbf{F}})_\mu}{\|\mathbf{C}\hat{\mathbf{F}}\|^3}, \tag{4.38}$$

so that

$$\frac{\partial^2 \Psi}{\partial \hat{F}_{\gamma\delta} \partial \hat{F}_{\alpha\beta}} = \partial^2 \Psi^1_{\alpha\beta\gamma\delta} + \partial^2 \Psi^2_{\alpha\beta\gamma\delta}. \tag{4.39}$$

Note that each of these are real and symmetric in the indices $\alpha\beta$ and $\gamma\delta$.

4.1.2.4 Eigenvalues and Eigenvectors of $\partial^2 \Psi^1$

Consider how $\partial^2 \Psi^1$ acts on tensors of the form $\mathbf{u}\mathbf{v}^T$:

$$\begin{aligned}
\partial^2 \Psi^1_{\alpha\beta\gamma\delta} u_\gamma v_\delta &= 2k \frac{C_{\theta\alpha\beta\gamma\delta}(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|} u_\gamma v_\delta \\
&= k \frac{(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|} \varepsilon_{\theta\alpha\gamma}\varepsilon_{\epsilon\beta\delta} dA_\epsilon u_\gamma v_\delta \\
&= k\varepsilon_{\alpha\theta\gamma} \frac{(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|} u_\gamma \varepsilon_{\beta\epsilon\delta} dA_\epsilon v_\delta \\
&= k \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{u} \right)_\alpha (d\mathbf{A} \times \mathbf{v})_\beta \\
&= k \|d\mathbf{A}\| \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{u} \right)_\alpha \left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{v} \right)_\beta .
\end{aligned} \tag{4.40}$$

Note that we do not pick up a minus sign in the third step because we have interchanged indices in both of the Levi-Civita symbols.

This form suggests that we look for eigenvectors of the linear transformations defined by $\mathbf{u} \mapsto \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{u}$ and $\mathbf{v} \mapsto \frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{v}$. Let $[\mathbf{C}\hat{\mathbf{F}}]_\times$ and $[d\mathbf{A}]_\times$ denote the skew symmetric matrices associated with these linear transformations, i.e. $[\mathbf{a}]_\times \mathbf{u} = \frac{\mathbf{a}}{\|\mathbf{a}\|} \times \mathbf{u}$ for $\mathbf{a} = \mathbf{C}\hat{\mathbf{F}}$ or $\mathbf{a} = d\mathbf{A}$.

The characteristic polynomial of $[\mathbf{a}]_\times$ is $\lambda(\lambda^2 + 1)$, so the eigenvalues of these matrices are $0, \pm i$. Since these matrices are normal, the corresponding eigenvectors are orthogonal (with respect to the Hermitian inner product).

Let $\mathbf{u}_0$, $\mathbf{u}_1$, and $\mathbf{u}_2$ be eigenvectors of $[\mathbf{C}\hat{\mathbf{F}}]_\times$ corresponding to 0, i, and $-i$ respectively. Despite the notation, these are not the vectors from the previous section. Similarly, let $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$ be eigenvectors of $[d\mathbf{A}]_\times$ corresponding to 0, i, and $-i$. If we denote these eigenvectors by $\lambda_0 = 0$, $\lambda_1 = i$, and $\lambda_2 = -i$, then the tensor products $\mathbf{u}_j \mathbf{v}_k^T$ are eigenvectors of $\partial^2 \Psi^1$:

$$\frac{\partial^2 \Psi^1}{\partial \hat{F}_{\gamma\delta} \partial \hat{F}_{\alpha\beta}} u_{j\gamma} v_{k\delta} = k\|d\mathbf{A}\| \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{u}_j \right)_\alpha \left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{v}_k \right)_\beta = k\|d\mathbf{A}\| \lambda_j \lambda_k u_{j\alpha} v_{k\beta}. \quad (4.41)$$

The corresponding eigenvalues of $\partial^2 \Psi^1$ are $k\|d\mathbf{A}\|\lambda_j\lambda_k$, which are all real:

$$\begin{aligned} &\lambda_0\lambda_0 = 0, \quad \lambda_0\lambda_1 = 0, \quad \lambda_0\lambda_2 = 0, \\ &\lambda_1\lambda_0 = 0, \quad \lambda_1\lambda_1 = -1, \quad \lambda_1\lambda_2 = 1, \\ &\lambda_2\lambda_0 = 0, \quad \lambda_2\lambda_1 = 1, \quad \lambda_2\lambda_2 = -1. \end{aligned} \quad (4.42)$$

Thus, there are three distinct eigenvalues for $\partial^2 \Psi^1$: 0 with multiplicity five, $k\|d\mathbf{A}\|$ with multiplicity two, and $-k\|d\mathbf{A}\|$ with multiplicity two.

It is clear from properties of the cross product that the vectors $\mathbf{u}_0$ and $\mathbf{v}_0$ are parallel to $\mathbf{C}\hat{\mathbf{F}}$ and $d\mathbf{A}$, respectively. The following five orthogonal vectors therefore span the kernel of $\partial^2 \Psi^1$:

$$\mathbf{u}_0 d\mathbf{A}^T, \quad \mathbf{u}_1 d\mathbf{A}^T, \quad \mathbf{u}_2 d\mathbf{A}^T, \quad (\mathbf{C}\hat{\mathbf{F}})\mathbf{v}_1^T, \quad (\mathbf{C}\hat{\mathbf{F}})\mathbf{v}_2^T. \quad (4.43)$$

Aside from $\mathbf{u}_0$, the vectors $\mathbf{u}_j$ and $\mathbf{v}_k$ here are complex. But complex eigenvectors of real matrices come in complex conjugate pairs. So by taking linear combinations, we can construct five real orthogonal vectors out of these. The first three are then equivalent to the three

vectors in the kernel of the Hessian from before, and the last two are of the form $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_0^T$ and $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_1^T$ for some real vectors $\mathbf{w}_0$ and $\mathbf{w}_1$ orthogonal to $d\mathbf{A}$.

To identify the four remaining eigenvectors, we use a common strategy of applying the Cayley-Hamilton Theorem to construct them. By Cayley-Hamilton, $[\mathbf{a}]_\times$ satisfies its own characteristic polynomial

$$([\mathbf{a}]_\times - i\mathbf{I})[\mathbf{a}]_\times([\mathbf{a}]_\times + i\mathbf{I}) = \mathbf{0}. \tag{4.44}$$

So for any vector $\mathbf{b}$ not parallel to $\mathbf{a}$, the vector

$$[\mathbf{a}]_\times([\mathbf{a}]_\times + i\mathbf{I})\mathbf{b} \tag{4.45}$$

is an eigenvector of $[\mathbf{a}]_\times$ with eigenvalue i, since it is in the kernel of $([\mathbf{a}]_\times - i\mathbf{I})$. Its complex conjugate pair

$$[\mathbf{a}]_\times([\mathbf{a}]_\times - i\mathbf{I})\mathbf{b} \tag{4.46}$$

is then an eigenvector with eigenvalue $-i$. We may rewrite these using cross products:

$$\mathbf{a} \times (\mathbf{a} \times \mathbf{b} \pm i\mathbf{b}) = \mathbf{a} \times (\mathbf{a} \times \mathbf{b}) \pm i\mathbf{a} \times \mathbf{b}. \tag{4.47}$$

If we further specify that $\mathbf{b}$ is orthogonal to $\mathbf{a}$, we get the simplified vectors

$$-\mathbf{b} + i\mathbf{a} \times \mathbf{b}, \quad -\mathbf{b} - i\mathbf{a} \times \mathbf{b}, \tag{4.48}$$

as eigenvectors of $[\mathbf{a}]_\times$ for i and $-i$.

So let $\mathbf{b}_1$ be orthogonal to $\mathbf{C}\hat{\mathbf{F}}$ and let $\mathbf{b}_2$ be orthogonal to $d\mathbf{A}$. We therefore have the following eigenvectors for $[\mathbf{C}\hat{\mathbf{F}}]_\times$ and $[d\mathbf{A}]_\times$:

$$\begin{aligned} \mathbf{u}_1 &= -\mathbf{b}_1 + i\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1, \quad \mathbf{u}_2 = -\mathbf{b}_1 - i\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1, \\ \mathbf{v}_1 &= -\mathbf{b}_2 + i\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2, \quad \mathbf{v}_2 = -\mathbf{b}_2 - i\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2. \end{aligned} \tag{4.49}$$

We can now construct the eigenvectors we are looking for. The two eigenvectors corresponding to eigenvalue $k\|d\mathbf{A}\|$ are

$$\mathbf{u}_1\mathbf{v}_2^T = \left(-\mathbf{b}_1 + i\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(-\mathbf{b}_2 - i\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T \tag{4.50}$$

and

$$\mathbf{u}_2\mathbf{v}_1^T = \left(-\mathbf{b}_1 - i\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(-\mathbf{b}_2 + i\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T, \tag{4.51}$$

which expand to

$$\mathbf{b}_1\mathbf{b}_2^T + \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T + i\left[\mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T - \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\mathbf{b}_2^T\right] \tag{4.52}$$

and

$$\mathbf{b}_1\mathbf{b}_2^T + \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T - i\left[\mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T - \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\mathbf{b}_2^T\right], \tag{4.53}$$

respectively. Since these are complex conjugate pairs, we take the real and imaginary parts to get two real eigenvectors for $\partial^2\Psi^1$ with eigenvalue $k\|d\mathbf{A}\|$:

$$\mathbf{b}_1\mathbf{b}_2^T + \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T \tag{4.54}$$

and

$$\mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T - \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\mathbf{b}_2^T. \tag{4.55}$$

Since $\mathbf{b}_1$ is orthogonal to $\mathbf{C}\hat{\mathbf{F}} \times \mathbf{b}_1$ and $\mathbf{b}_2$ is orthogonal to $d\mathbf{A} \times \mathbf{b}_2$, these eigenvectors are still orthogonal. By similar reasoning, they are also orthogonal to the five eigenvectors in the kernel.

Next, the eigenvectors corresponding to $-k\|d\mathbf{A}\|$ are

$$\mathbf{u}_1\mathbf{v}_1^T = \left(-\mathbf{b}_1 + i\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(-\mathbf{b}_2 + i\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T \tag{4.56}$$

and

$$\mathbf{u}_2\mathbf{v}_2^T = \left(-\mathbf{b}_1 - i\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(-\mathbf{b}_2 - i\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T \tag{4.57}$$

which expand to

$$\mathbf{b}_1\mathbf{b}_2^T - \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T - i\left[\mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T + \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\mathbf{b}_2^T\right] \tag{4.58}$$

and

$$\mathbf{b}_1\mathbf{b}_2^T - \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T + i\left[\mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T + \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\mathbf{b}_2^T\right], \tag{4.59}$$

respectively. As before, we can take the real and imaginary parts to get real eigenvectors for $-k\|d\mathbf{A}\|$:

$$\mathbf{b}_1\mathbf{b}_2^T - \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T \tag{4.60}$$

and

$$\mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T + \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\mathbf{b}_2^T. \tag{4.61}$$

We must know these four eigenvectors are also eigenvectors for $\partial^2\Psi^2$ in order to conclude that they are eigenvectors of the Hessian. As each of these eigenvectors is a sum of tensor products, consider the action of $\partial^2\Psi^2$ on a tensor product $\mathbf{u}\mathbf{v}^T$:

$$\begin{aligned}
\partial^2\Psi^2_{\alpha\beta\gamma\delta}u_\gamma v_\delta &= 4k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left[C_{\theta\gamma\delta\zeta\eta}\hat{F}_{\zeta\eta}u_\gamma v_\delta - \frac{(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|}C_{\mu\gamma\delta\phi\chi}\hat{F}_{\phi\chi}u_\gamma v_\delta\frac{(\mathbf{C}\hat{\mathbf{F}})_\mu}{\|\mathbf{C}\hat{\mathbf{F}}\|}\right]\\
&= 4k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left[\frac{1}{2}\varepsilon_{\theta\gamma\zeta}\varepsilon_{\epsilon\delta\eta}dA_\epsilon\hat{F}_{\zeta\eta}u_\gamma v_\delta - \frac{(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{1}{2}\varepsilon_{\mu\gamma\phi}\varepsilon_{\iota\delta\chi}dA_\iota\hat{F}_{\phi\chi}u_\gamma v_\delta\frac{(\mathbf{C}\hat{\mathbf{F}})_\mu}{\|\mathbf{C}\hat{\mathbf{F}}\|}\right]\\
&= 2k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left[\varepsilon_{\theta\gamma\zeta}u_\gamma\hat{F}_{\zeta\eta}(d\mathbf{A}\times\mathbf{v})_\eta - \frac{(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{F}_{\phi\chi}(d\mathbf{A}\times\mathbf{v})_\chi\varepsilon_{\phi\mu\gamma}\frac{(\mathbf{C}\hat{\mathbf{F}})_\mu}{\|\mathbf{C}\hat{\mathbf{F}}\|}u_\gamma\right]\\
&= 2k\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left[\Big(\mathbf{u}\times\hat{\mathbf{F}}(d\mathbf{A}\times\mathbf{v})\Big)_\theta - \frac{(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}(d\mathbf{A}\times\mathbf{v})\cdot\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{u}\right)\right]\\
&= 2k\|d\mathbf{A}\|\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left[\left(\mathbf{u}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{v}\right)\right)_\theta\right.\\
&\qquad\left. -\frac{(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{v}\right)\cdot\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{u}\right)\right].
\end{aligned} \tag{4.62}$$

We show that all four of these are in the kernel of $\partial^2\Psi^2$. Now, note that for any choice of $\mathbf{b}_1$ orthogonal to $\mathbf{C}\hat{\mathbf{F}}$ and $\mathbf{b}_2$ orthogonal to $d\mathbf{A}$, the eigenvectors for $k\|d\mathbf{A}\|$ form a basis for that two dimensional eigenspace, and similarly for the other two. The corresponding vectors for any other valid choice of $\mathbf{b}_1$ and $\mathbf{b}_2$ will then be linear combinations of these basis vectors. So if we show for one particular choice of $\mathbf{b}_2$ that all four are in the kernel, then the result will also hold for any other valid choice of $\mathbf{b}_2$.

So for convenience, we choose $\mathbf{b}_2 = \hat{\mathbf{F}}^{-1}\mathbf{b}_1$ for any $\mathbf{b}_1$ orthogonal to $\mathbf{C}\hat{\mathbf{F}}$. This is a valid choice of $\mathbf{b}_2$, since

$$0 = \mathbf{b}_1^T\mathbf{C}\hat{\mathbf{F}} = \mathbf{b}_1 J\hat{\mathbf{F}}^{-T}d\mathbf{A} = J\mathbf{b}_1^T\hat{\mathbf{F}}^{-T}d\mathbf{A} = J(\hat{\mathbf{F}}^{-1}\mathbf{b}_1)^T d\mathbf{A} = J\mathbf{b}_2^T d\mathbf{A}. \tag{4.63}$$

As $J \neq 0$, we have $\mathbf{b}_2$ orthogonal to $d\mathbf{A}$. Note that we also have $\mathbf{b}_1 = \hat{\mathbf{F}}\mathbf{b}_2$ with this choice. We show that the term in brackets above is 0.

First consider the eigenvectors of the form

$$\mathbf{b}_1\mathbf{b}_2^T \pm \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T. \tag{4.64}$$

The term in brackets applied to each of the tensor products in this sum is

$$
\begin{aligned}
&\mathbf{b}_1 \times \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) - \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \\
&\quad \pm \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \times \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)\right) \\
&\quad \mp \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)\right) \cdot \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\right) \\
&= \hat{\mathbf{F}}\mathbf{b}_2 \times \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) - \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \cdot \left(\mathbf{b}_1 \times \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)\right) \\
&\quad \mp \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \times \hat{\mathbf{F}}\mathbf{b}_2 \mp \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\mathbf{b}_2 \cdot \mathbf{b}_1 \\
&= J\hat{\mathbf{F}}^{-T}\left(\mathbf{b}_2 \times \left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)\right) - \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \cdot \left(\hat{\mathbf{F}}\mathbf{b}_2 \times \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)\right) \\
&\quad \mp \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \times \mathbf{b}_1 \mp \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\mathbf{b}_1 \cdot \mathbf{b}_1 \\
&= \frac{\|\mathbf{b}_2\|^2}{\|d\mathbf{A}\|}J\hat{\mathbf{F}}^{-T}d\mathbf{A} - \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \cdot J\hat{\mathbf{F}}^{-T}\left(\mathbf{b}_2 \times \left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)\right) \\
&\quad \pm \|\mathbf{b}_1\|^2 \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \mp \|\mathbf{b}_1\|^2 \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \\
&= \frac{\|\mathbf{b}_2\|^2}{\|d\mathbf{A}\|}\mathbf{C}\hat{\mathbf{F}} - \frac{\|\mathbf{b}_2\|^2}{\|d\mathbf{A}\|}\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \cdot J\hat{\mathbf{F}}^{-T}d\mathbf{A} \\
&= 0, \qquad (4.65)
\end{aligned}
$$

where the last line follows from the fact that $J\hat{\mathbf{F}}^{-T}d\mathbf{A} = \mathbf{C}\hat{\mathbf{F}}$. We have also made use of the following property of cross products in this calculation: $\mathbf{A}\mathbf{u} \times \mathbf{A}\mathbf{v} = \det(\mathbf{A})\mathbf{A}^{-T}(\mathbf{u} \times \mathbf{v})$.

Next, consider the other two eigenvectors of the form

$$
\mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T \mp \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\mathbf{b}_2^T. \qquad (4.66)
$$

The term in brackets is now

$$
\begin{aligned}
&\mathbf{b}_1 \times \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)\right) - \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)\right) \cdot \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \\
&\quad \mp \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \times \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \\
&\quad \pm \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\right) \\
&= -\mathbf{b}_1 \times \hat{\mathbf{F}}\mathbf{b}_2 + \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\mathbf{b}_2 \cdot \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \mp \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \times \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \\
&\quad \mp \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot \mathbf{b}_1 \\
&= -\mathbf{b}_1 \times \mathbf{b}_1 + \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\mathbf{b}_1 \cdot \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \pm \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \times \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right) \\
&\quad \mp \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot \mathbf{b}_1 \\
&= \pm\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot \mathbf{b}_1 \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \mp \hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\mathbf{b}_1 \\
&\quad \mp \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot \mathbf{b}_1 \\
&= \mp\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot \hat{\mathbf{F}}^T \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\mathbf{b}_1 \\
&= \mp\frac{\mathbf{b}_1}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot \hat{\mathbf{F}}^T J\hat{\mathbf{F}}^{-T} d\mathbf{A} \\
&= \mp J\frac{\mathbf{b}_1}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right) \cdot d\mathbf{A} \\
&= 0. \qquad (4.67)
\end{aligned}
$$

We have thus shown that the eigenvectors of the first term for $\pm k\|d\mathbf{A}\|$ are also eigenvectors of the Hessian, with the same eigenvalues. Since $\mathbf{b}_2$ and $d\mathbf{A} \times \mathbf{b}_2$ are both orthogonal to $d\mathbf{A}$, it follows that we now have seven orthogonal eigenvectors for the Hessian.

4.1.2.5 Eigenvalues and Eigenvectors of $\partial^2\Psi^2$

Denote the seven eigenvectors found so far by

$$\delta\hat{\mathbf{F}}_i = \mathbf{u}_i d\mathbf{A}^T \tag{4.68}$$

for $i = 0, 1, 2$ (where the vectors u here are arbitrary orthogonal vectors in $\mathbb{R}^3$), and

$$\begin{aligned}
\delta\hat{\mathbf{F}}_3 &= \mathbf{b}_1\mathbf{b}_2^T + \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T, \\
\delta\hat{\mathbf{F}}_4 &= \mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T - \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\mathbf{b}_2^T, \\
\delta\hat{\mathbf{F}}_5 &= \mathbf{b}_1\mathbf{b}_2^T - \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T, \\
\delta\hat{\mathbf{F}}_6 &= \mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|} \times \mathbf{b}_2\right)^T + \left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|} \times \mathbf{b}_1\right)\mathbf{b}_2^T,
\end{aligned} \tag{4.69}$$

for $\mathbf{b}_1$ orthogonal to $\mathbf{C}\hat{\mathbf{F}}$ and $\mathbf{b}_2$ orthogonal to $d\mathbf{A}$. From the previous two sections, we see that $\mathsf{span}(\delta\hat{\mathbf{F}}_0, \ldots, \delta\hat{\mathbf{F}}_6)$ is in the kernel of $\partial^2\Psi^2$. Hence, $\partial^2\Psi^2$ is at most rank 2. So we look for eigenvectors of the form $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_0^T$ and $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_1^T$, which are a basis for the orthogonal complement of $\mathsf{span}(\delta\hat{\mathbf{F}}_0, \ldots, \delta\hat{\mathbf{F}}_6)$ for any linearly independent $\mathbf{w}_0$ and $\mathbf{w}_1$ both orthogonal to $d\mathbf{A}$.

The remaining two eigenvectors for $\partial^2\Psi^2$ must be of this form. If the rank is 2, then this follows from the fact that eigenvectors of distinct eigenvalues for symmetric matrices are orthogonal, and hence the eigenvectors must be in the orthogonal complement of $\mathsf{span}(\delta\hat{\mathbf{F}}_0, \ldots, \delta\hat{\mathbf{F}}_6)$. If instead the rank is 1, then the eigenvector for the nonzero eigenvalue must for the same reason be of the form $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_0^T$. Then, $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_1^T$ for any $\mathbf{w}_1$ orthogonal to $\mathbf{w}_0$ and $d\mathbf{A}$ will be in the kernel and therefore also an eigenvector.

From the previous section, we have that

$$\partial^2\Psi^2_{\alpha\beta\gamma\delta}\frac{(\mathbf{C}\hat{\mathbf{F}})_\gamma}{\|\mathbf{C}\hat{\mathbf{F}}\|}w_{i\delta} = 2k\|d\mathbf{A}\|\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left[\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right)\right)_\theta - \frac{(\mathbf{C}\hat{\mathbf{F}})_\theta}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right)\cdot\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\right)\right]$$
$$= 2k\|d\mathbf{A}\|\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right)\right)_\theta. \tag{4.70}$$

Multiplying this to $\frac{\mathbf{C}\hat{\mathbf{F}}_\alpha}{\|\mathbf{C}\hat{\mathbf{F}}\|}w_{j\beta}$, we get

$$2k\|d\mathbf{A}\|\frac{C_{\theta\alpha\beta\kappa\lambda}\hat{F}_{\kappa\lambda}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right)\right)_\theta\frac{\mathbf{C}\hat{\mathbf{F}}_\alpha}{\|\mathbf{C}\hat{\mathbf{F}}\|}w_{j\beta}$$
$$= k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\varepsilon_{\theta\alpha\kappa}\varepsilon_{\epsilon\beta\lambda}\frac{dA_\epsilon}{\|d\mathbf{A}\|}\hat{F}_{\kappa\lambda}\frac{(\mathbf{C}\hat{\mathbf{F}})_\alpha}{\|\mathbf{C}\hat{\mathbf{F}}\|}w_{j\beta}\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right)\right)_\theta$$
$$= k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\varepsilon_{\theta\alpha\kappa}\frac{(\mathbf{C}\hat{\mathbf{F}})_\alpha}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{F}_{\kappa\lambda}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_j\right)_\lambda\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right)\right)_\theta$$
$$= k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_j\right)\right)\cdot\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right)\right). \tag{4.71}$$

This expression can be simplified, as follows:

$$k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_j\right)\right)\cdot\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right)\right)$$
$$= k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left[\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_j\right)\cdot\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right) - \frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\cdot\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_j\right)\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\cdot\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right)\right]$$
$$= k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_j\right)\cdot\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right), \tag{4.72}$$

where the last line follows from

$$(\mathbf{C}\hat{\mathbf{F}})\cdot\hat{\mathbf{F}}(d\mathbf{A}\times\mathbf{w}) = \hat{\mathbf{F}}^T(\mathbf{C}\hat{\mathbf{F}})\cdot(d\mathbf{A}\times\mathbf{w}) = Jd\mathbf{A}\cdot(d\mathbf{A}\times\mathbf{w}) = 0. \tag{4.73}$$

Hence,

$$\left(\frac{\mathbf{C}\hat{\mathbf{F}}_\alpha}{\|\mathbf{C}\hat{\mathbf{F}}\|}w_{j\beta}\right)\partial^2\Psi^2_{\alpha\beta\gamma\delta}\left(\frac{(\mathbf{C}\hat{\mathbf{F}})_\gamma}{\|\mathbf{C}\hat{\mathbf{F}}\|}w_{i\delta}\right) = k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_j\right)\cdot\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_i\right), \tag{4.74}$$

and so $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_0^T$ and $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_1^T$ will be eigenvectors if and only if

$$\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_1\right)\cdot\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_0\right)=0. \tag{4.75}$$

In this case, if we further assume that $\mathbf{w}_0$ and $\mathbf{w}_1$ have been normalized, this is equivalent to

$$\hat{\mathbf{F}}\mathbf{w}_0\cdot\hat{\mathbf{F}}\mathbf{w}_1=0. \tag{4.76}$$

With this normalization assumption, we also have the remaining two eigenvalues

$$\begin{aligned}
k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left\|\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_0\right)\right\|^2 &= k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\|\hat{\mathbf{F}}\mathbf{w}_1\|^2,\\
k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\left\|\hat{\mathbf{F}}\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{w}_1\right)\right\|^2 &= k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\|\hat{\mathbf{F}}\mathbf{w}_0\|^2,
\end{aligned} \tag{4.77}$$

where we have used the fact that $\frac{d\mathbf{A}}{\|d\mathbf{A}\|}$ cross either $\mathbf{w}_0$ or $\mathbf{w}_1$ gives plus or minus the other one.

To construct these, let $\tilde{\mathbf{w}}_0$ and $\tilde{\mathbf{w}}_1$ be any pair of orthonormal vectors which are also orthogonal to $d\mathbf{A}$. Consider the following 2×2 matrix

$$\mathbf{B}=\begin{bmatrix}\hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot\hat{\mathbf{F}}\tilde{\mathbf{w}}_0 & \hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot\hat{\mathbf{F}}\tilde{\mathbf{w}}_1\\ \hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot\hat{\mathbf{F}}\tilde{\mathbf{w}}_1 & \hat{\mathbf{F}}\tilde{\mathbf{w}}_1\cdot\hat{\mathbf{F}}\tilde{\mathbf{w}}_1\end{bmatrix}. \tag{4.78}$$

This is a real symmetric matrix, and therefore there exists an orthogonal matrix $\mathbf{U}$ such that $\mathbf{U}^T\mathbf{B}\mathbf{U}$ is diagonal. Let

$$\mathbf{U}=\begin{bmatrix}u_{11} & u_{12}\\ u_{21} & u_{22}\end{bmatrix}. \tag{4.79}$$

Then,

$$\begin{aligned}
\mathbf{B}\mathbf{U} &= \begin{bmatrix}\hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot\hat{\mathbf{F}}\tilde{\mathbf{w}}_0 & \hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot\hat{\mathbf{F}}\tilde{\mathbf{w}}_1\\ \hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot\hat{\mathbf{F}}\tilde{\mathbf{w}}_1 & \hat{\mathbf{F}}\tilde{\mathbf{w}}_1\cdot\hat{\mathbf{F}}\tilde{\mathbf{w}}_1\end{bmatrix}\begin{bmatrix}u_{11} & u_{12}\\ u_{21} & u_{22}\end{bmatrix}\\
&= \begin{bmatrix}\hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot(u_{11}\hat{\mathbf{F}}\tilde{\mathbf{w}}_0+u_{21}\hat{\mathbf{F}}\tilde{\mathbf{w}}_1) & \hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot(u_{12}\hat{\mathbf{F}}\tilde{\mathbf{w}}_0+u_{22}\hat{\mathbf{F}}\tilde{\mathbf{w}}_1)\\ \hat{\mathbf{F}}\tilde{\mathbf{w}}_1\cdot(u_{11}\hat{\mathbf{F}}\tilde{\mathbf{w}}_0+u_{21}\hat{\mathbf{F}}\tilde{\mathbf{w}}_1) & \hat{\mathbf{F}}\tilde{\mathbf{w}}_1\cdot(u_{12}\hat{\mathbf{F}}\tilde{\mathbf{w}}_0+u_{22}\hat{\mathbf{F}}\tilde{\mathbf{w}}_1)\end{bmatrix}\\
&= \begin{bmatrix}\hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot\hat{\mathbf{F}}(u_{11}\tilde{\mathbf{w}}_0+u_{21}\tilde{\mathbf{w}}_1) & \hat{\mathbf{F}}\tilde{\mathbf{w}}_0\cdot\hat{\mathbf{F}}(u_{12}\tilde{\mathbf{w}}_0+u_{22}\tilde{\mathbf{w}}_1)\\ \hat{\mathbf{F}}\tilde{\mathbf{w}}_1\cdot\hat{\mathbf{F}}(u_{11}\tilde{\mathbf{w}}_0+u_{21}\tilde{\mathbf{w}}_1) & \hat{\mathbf{F}}\tilde{\mathbf{w}}_1\cdot\hat{\mathbf{F}}(u_{12}\tilde{\mathbf{w}}_0+u_{22}\tilde{\mathbf{w}}_1)\end{bmatrix}
\end{aligned} \tag{4.80}$$

and $\mathbf{U}^T\mathbf{B}\mathbf{U}$ simplifies to

$$\begin{bmatrix} \|\hat{\mathbf{F}}(u_{11}\tilde{\mathbf{w}}_0 + u_{21}\tilde{\mathbf{w}}_1)\|^2 & \hat{\mathbf{F}}(u_{11}\tilde{\mathbf{w}}_0 + u_{21}\tilde{\mathbf{w}}_1) \cdot \hat{\mathbf{F}}(u_{12}\tilde{\mathbf{w}}_0 + u_{22}\tilde{\mathbf{w}}_1) \\ \hat{\mathbf{F}}(u_{11}\tilde{\mathbf{w}}_0 + u_{21}\tilde{\mathbf{w}}_1) \cdot \hat{\mathbf{F}}(u_{12}\tilde{\mathbf{w}}_0 + u_{22}\tilde{\mathbf{w}}_1) & \|\hat{\mathbf{F}}(u_{12}\tilde{\mathbf{w}}_0 + u_{22}\tilde{\mathbf{w}}_1)\|^2 \end{bmatrix}, \quad (4.81)$$

which is diagonal. So if we define

$$\mathbf{w}_0 = u_{11}\tilde{\mathbf{w}}_0 + u_{21}\tilde{\mathbf{w}}_1 \quad (4.82)$$

and

$$\mathbf{w}_1 = u_{12}\tilde{\mathbf{w}}_0 + u_{22}\tilde{\mathbf{w}}_1, \quad (4.83)$$

the condition

$$\hat{\mathbf{F}}\mathbf{w}_0 \cdot \hat{\mathbf{F}}\mathbf{w}_1 = 0 \quad (4.84)$$

is satisfied. The vectors are clearly still orthogonal to $d\mathbf{A}$, and also orthogonal to each other since

$$\mathbf{w}_0 \cdot \mathbf{w}_1 = u_{11}u_{12} + u_{21}u_{22} = 0, \quad (4.85)$$

where the last step is the dot product of the columns of $\mathbf{U}$. The vectors are also still unit length.

Lastly, note that $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_0^T$ and $(\mathbf{C}\hat{\mathbf{F}})\mathbf{w}_1^T$ are in the kernel of $\partial^2\Psi^1$, and hence they are the last two eigenvectors of the Hessian.

We then have the following eigenvalues and corresponding eigenvectors for the Hessian:

Eigenvalue	Eigenvectors
$k\|d\mathbf{A}\|$	$\frac{1}{\sqrt{2}}\mathbf{b}_1\mathbf{b}_2^T + \frac{1}{\sqrt{2}}\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T$ $\frac{1}{\sqrt{2}}\mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T - \frac{1}{\sqrt{2}}\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\mathbf{b}_2^T$
$-k\|d\mathbf{A}\|$	$\frac{1}{\sqrt{2}}\mathbf{b}_1\mathbf{b}_2^T - \frac{1}{\sqrt{2}}\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T$ $\frac{1}{\sqrt{2}}\mathbf{b}_1\left(\frac{d\mathbf{A}}{\|d\mathbf{A}\|}\times\mathbf{b}_2\right)^T + \frac{1}{\sqrt{2}}\left(\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\times\mathbf{b}_1\right)\mathbf{b}_2^T$
$k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\|\hat{\mathbf{F}}\mathbf{w}_1\|^2$	$\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\mathbf{w}_0^T$
$k\frac{\|d\mathbf{A}\|^2}{\|\mathbf{C}\hat{\mathbf{F}}\|}\|\hat{\mathbf{F}}\mathbf{w}_0\|^2$	$\frac{\mathbf{C}\hat{\mathbf{F}}}{\|\mathbf{C}\hat{\mathbf{F}}\|}\mathbf{w}_1^T$
0	$\mathbf{u}_0\frac{d\mathbf{A}^T}{\|d\mathbf{A}\|}$ $\mathbf{u}_1\frac{d\mathbf{A}^T}{\|d\mathbf{A}\|}$ $\mathbf{u}_2\frac{d\mathbf{A}^T}{\|d\mathbf{A}\|}$

where $\mathbf{b}_1$ and $\mathbf{b}_2$ are any unit vectors orthogonal to $d\mathbf{A}$ and $\mathbf{C}\hat{\mathbf{F}}$ respectively, $\mathbf{w}_0$ and $\mathbf{w}_1$ are any orthonormal vectors orthogonal to $d\mathbf{A}$ satisfying

$$\hat{\mathbf{F}}\mathbf{w}_0 \cdot \hat{\mathbf{F}}\mathbf{w}_1 = 0, \tag{4.86}$$

and $\mathbf{u}_0$, $\mathbf{u}_1$, and $\mathbf{u}_2$ are any orthonormal basis for $\mathbb{R}^3$. where $\mathbf{b}_1$ and $\mathbf{b}_2$ are any unit vectors orthogonal to $d\mathbf{A}$ and $\mathbf{C}\hat{\mathbf{F}}$ respectively, $\mathbf{w}_0$ and $\mathbf{w}_1$ are any orthonormal vectors orthogonal to $d\mathbf{A}$ satisfying

$$\hat{\mathbf{F}}\mathbf{w}_0 \cdot \hat{\mathbf{F}}\mathbf{w}_1 = 0, \tag{4.87}$$

and $\mathbf{u}_0$, $\mathbf{u}_1$, and $\mathbf{u}_2$ are any orthonormal basis for $\mathbb{R}^3$.

4.2 Derivation of the APIC Inertia Tensor

When using APIC, the angular momentum associated with a particle $\mathbf{x}_p$ with mass m_p and linear and affine velocities $\mathbf{v}_p$ and $\mathbf{A}_p$ is defined from its associated mass/momentum distribution on the grid as

$$\mathbf{l}_p = \sum_{\mathbf{i}} \left(\mathbf{x}_\mathbf{i} - \mathbf{x}_p\right) \times m_p N_\mathbf{i}(\mathbf{x}_p)\left(\mathbf{v}_p + \mathbf{A}_p\left(\mathbf{x}_\mathbf{i} - \mathbf{x}_p\right)\right), \tag{4.88}$$

where $N_{\mathbf{i}}(\mathbf{x})$ are quadratic B-spline interpolating functions associated with grid node $\mathbf{x_i}$ used to transfer particle quantities to grid quantities. Using the fact that linear reproduction property of B-splines (see Section 4.2.1.2) we may remove the constant velocity term and rewrite this as:

$$\begin{aligned}\mathbf{l}_p &= \sum_{\mathbf{i}} (\mathbf{x_i} - \mathbf{x}_p) \times m_p N_{\mathbf{i}} \mathbf{A}_p (\mathbf{x_i} - \mathbf{x}_p) \\ &= \frac{\Delta x^2}{4} m_p \boldsymbol{\epsilon} : \mathbf{A}_p^T, \end{aligned} \tag{4.89}$$

where $(\boldsymbol{\epsilon} : \mathbf{B})_\alpha := \varepsilon_{\alpha\beta\gamma} B_{\gamma\beta}$. To derive the last equality, we first rewrite $\mathbf{l}_p$ using summation notation:

$$\begin{aligned}\mathbf{l}_p &= \sum_{\mathbf{i}} (\mathbf{x_i} - \mathbf{x}_p) \times m_p N_{\mathbf{i}} \mathbf{A}_p (\mathbf{x_i} - \mathbf{x}_p) \\ &= \sum_{\mathbf{i}} \epsilon_{\alpha\beta\gamma} (\mathbf{x_i} - \mathbf{x}_p)_\beta m_p N_{\mathbf{i}}(\mathbf{x}_p) A_{p\gamma\delta} (\mathbf{x_i} - \mathbf{x}_p)_\delta \\ &= m_p \epsilon_{\alpha\beta\gamma} A_{p\gamma\delta} \sum_{\mathbf{i}} (\mathbf{x_i} - \mathbf{x}_p)_\beta (\mathbf{x_i} - \mathbf{x}_p)_\delta N_{\mathbf{i}}(\mathbf{x}_p). \end{aligned} \tag{4.90}$$

We now show that the inertia tensor

$$\mathbf{D}_p := \sum_{\mathbf{i}} (\mathbf{x_i} - \mathbf{x}_p)(\mathbf{x_i} - \mathbf{x}_p)^T N_{\mathbf{i}}(\mathbf{x}_p) \tag{4.91}$$

is a constant multiple of the identity.

Consider an arbitrary component $D_{p\alpha\beta}$. Then, using the fact that $N_{\mathbf{i}}$ is defined as a dyadic product of 1D B-splines, we obtain

$$\begin{aligned} D_{p\alpha\beta} &= \sum_{\mathbf{i}} (\mathbf{x_i} - \mathbf{x}_p)_\alpha (\mathbf{x_i} - \mathbf{x}_p)_\beta N_{\mathbf{i}}(\mathbf{x}_p) \\ &= \sum_{\mathbf{i}} (\mathbf{x_i} - \mathbf{x}_p)_\alpha (\mathbf{x_i} - \mathbf{x}_p)_\beta \prod_\delta N\left(\frac{(\mathbf{x_i} - \mathbf{x}_p)_\delta}{\Delta x}\right) \\ &= \sum_{\mathbf{i}} \prod_\gamma (\mathbf{x_i} - \mathbf{x}_p)_\gamma^{n_\gamma^{\alpha,\beta}} \prod_\delta N\left(\frac{(\mathbf{x_i} - \mathbf{x}_p)_\delta}{\Delta x}\right) \\ &= \sum_{\mathbf{i}} \prod_\gamma (\mathbf{x_i} - \mathbf{x}_p)_\gamma^{n_\gamma^{\alpha,\beta}} N\left(\frac{(\mathbf{x_i} - \mathbf{x}_p)_\gamma}{\Delta x}\right), \end{aligned} \tag{4.92}$$

where $n_\gamma^{\alpha,\beta} = \delta_{\alpha\gamma} + \delta_{\beta\gamma}$. Writing the multi-index $\mathbf{i}$ as (i_1, i_2, i_3), we note that $x_{\mathbf{i}\gamma}$ depends only on the component index i_γ. We use this observation to switch the order of the sum and product:

$$\begin{aligned} D_{p\alpha\beta} &= \sum_{\mathbf{i}} \prod_{\gamma} (\mathbf{x_i} - \mathbf{x}_p)_\gamma^{n_\gamma^{\alpha,\beta}} N\left(\frac{(\mathbf{x_i} - \mathbf{x}_p)_\gamma}{\Delta x}\right) \\ &= \sum_{i_1,i_2,i_3} \prod_{\gamma} (x_{i_\gamma} - x_{p\gamma})^{n_\gamma^{\alpha,\beta}} N\left(\frac{x_{i_\gamma} - x_{p\gamma}}{\Delta x}\right) \\ &= \prod_{\gamma} \sum_{i_\gamma} (x_{i_\gamma} - x_{p\gamma})^{n_\gamma^{\alpha,\beta}} N\left(\frac{x_{i_\gamma} - x_{p\gamma}}{\Delta x}\right). \end{aligned} \tag{4.93}$$

We have therefore reduced this to a product of 1D sums. There are three cases to consider, corresponding to the possible values 0, 1, and 2 for $n_\gamma^{\alpha\beta}$. When $n_\gamma^{\alpha\beta} = 0$, this becomes a sum over all B-splines, which is 1. When $n_\gamma^{\alpha\beta} = 1$, the linear reproduction property of B-splines implies that the sum is 0. In particular, this implies that $\mathbf{D}_p$ is diagonal since $n_\gamma^{\alpha\beta} = 1$ occurs for some γ precisely when $\alpha \neq \beta$. The diagonal values $D_{p\mu\mu}$ are then equal to the sum

$$\sum_{i_\mu} (x_{i_\mu} - x_{p\mu})^2 N\left(\frac{x_{i_\mu} - x_{p\mu}}{\Delta x}\right), \tag{4.94}$$

since the sums in the other dimensions are 1 from the $n_\gamma^{\alpha\beta} = 0$ case. This sum will be independent of the value of $x_{p\mu}$, making $\mathbf{D}_p$ a multiple of the identity.

As this is now a 1D calculation, we write x_i and x_p in place of x_{i_μ} and $x_{p\mu}$, respectively. Next, define the knot sequence

$$a_i = x_i - \frac{3}{2}\Delta x. \tag{4.95}$$

With these knots, define the degree 0 B-spline N_i^0 by

$$N_i^0(x) = \begin{cases} 1 & x \in [a_i, a_{i+1}) \\ 0 & \text{otherwise} \end{cases}, \tag{4.96}$$

and the higher degree B-splines N_i^1 and N_i^2 are defined using the standard recursion formula. The sum is expressed in this notation as

$$\sum_i (x_i - x_p)^2 N_i^2(x_p), \tag{4.97}$$

since N_i^2 is equal to N with a shifted and scaled argument. We use Marsden's identity [PBP02]

$$\sum_i (a_{i+1} - a)(a_{i+2} - a)N_i^2(x_p) = (x_p - a)^2 \tag{4.98}$$

with $a = x_p - \frac{\Delta x}{2}$ to compute it. Note that $x_i - x_p$ can be rewritten in terms of the knot sequence a_i as $x_i - x_p = a_{i+1} - a$. Then,

$$\begin{aligned}
&\sum_i (x_i - x_p)^2 N_i^2(x_p) \\
&= \sum_i (a_{i+1} - a)^2 N_i^2(x_p) \\
&= \sum_i (a_{i+1} - a)(a_{i+2} - a - \Delta x)N_i^2(x_p) \\
&= \sum_i (a_{i+1} - a)(a_{i+2} - a)N_i^2(x_p) + \Delta x \sum_i (a_{i+1} - a)N_i^2(x_p) \\
&= (x_p - a)^2 \\
&= \frac{\Delta x^2}{4},
\end{aligned} \tag{4.99}$$

where the second term is 0 by the linear reproduction property since $a_{i+1} - a = x_i - x_p$.

Hence, $\mathbf{D}_p = \frac{\Delta x^2}{4}\mathbf{I}$ and

$$\begin{aligned}
\mathbf{l}_p &= \frac{\Delta x^2}{4} m_p \epsilon_{\alpha\beta\gamma} A_{p\gamma\delta} \delta_{\beta\delta} \\
&= \frac{\Delta x^2}{4} m_p \epsilon_{\alpha\beta\gamma} A_{p\gamma\beta} \\
&= \frac{\Delta x^2}{4} m_p \boldsymbol{\epsilon} : \mathbf{A}_p^T.
\end{aligned} \tag{4.100}$$

This value is specific to the case where N is quadratic. The inertia tensor calculation with splines of arbitrary degree is presented in the following subsection.

4.2.1 Inertia Tensor

In this section we show that the inertia tensor

$$\mathbf{D}_p = \sum_{\mathbf{i}} (\mathbf{x_i} - \mathbf{x}_p)(\mathbf{x_i} - \mathbf{x}_p)^T N_{\mathbf{i}}(\mathbf{x}_p) \tag{4.101}$$

is a constant multiple of the identity for the dyadic product $N_{\mathbf{i}}$ of cardinal B-splines of degree $n \geq 2$ with spacing Δx, and compute this constant. It is shown above that

$$D_{p\alpha\beta} = \prod_{\gamma} \sum_{i_\gamma} (x_{i_\gamma} - x_{p\gamma})^{n_\gamma^{\alpha,\beta}} N\left(\frac{x_{i_\gamma} - x_{p\gamma}}{\Delta x}\right) \tag{4.102}$$

where $n_\gamma^{\alpha,\beta} = \delta_{\alpha\gamma} + \delta_{\beta\gamma}$, $x_{i_\gamma} = x_{\mathbf{i}\gamma}$ is the γth component of $\mathbf{x_i}$ (i.e. each component of $\mathbf{x_i}$ depends only on one component index of $\mathbf{i}$; namely the γth index), and N is the cardinal B-spline of degree n centered at 0 unit spacing. As $n_\gamma^{\alpha,\beta}$ can take on the values 0, 1, and 2, there are three cases to consider for each sum in the product. As before, we drop the γ subscript throughout the rest of this section for notational convenience.

The sums in these three cases are:

$$\sum_i N\left(\frac{x_i - x_p}{\Delta x}\right) = 1, \tag{4.103}$$

$$\sum_i (x_i - x_p) N\left(\frac{x_i - x_p}{\Delta x}\right) = 0, \tag{4.104}$$

$$\sum_i (x_i - x_p)^2 N\left(\frac{x_i - x_p}{\Delta x}\right) = \frac{n+1}{12}\Delta x^2. \tag{4.105}$$

The first follows from the fact that B-splines of any degree form a partition of unity. The second follows from the linear reproduction property, which we also prove in this section. We used the value of this sum above to deduce that $\mathbf{D}_p$ is diagonal. The third is the main focus of this section.

4.2.1.1 Background & Notation

We now introduce the notation and recursive formula for B-splines of varying degrees over multiple knot sequences.

For a given knot sequence $(a_i)_{i\in\mathbb{Z}}$ with $a_i < a_{i+1}$ for all i, the 0th degree B-splines N_i^0 are defined by

$$N_i^0(x) = \begin{cases} 1 & x \in [a_i, a_{i+1}) \\ 0 & \text{otherwise} \end{cases}, \tag{4.106}$$

and the general kth degree B-splines N_i^k are defined using the standard recursive formula

$$N_i^k(x) = \frac{x - a_i}{a_{i+k} - a_i} N_i^{k-1}(x) + \left(1 - \frac{x - a_{i+1}}{a_{i+1+k} - a_{i+1}}\right) N_{i+1}^{k-1}(x). \tag{4.107}$$

These B-splines are called cardinal B-splines if $a_{i+1} - a_i$ is a constant for all i. We will need the following properties.

Proposition 1. *The following hold for all x.*

(a) **(Partition of Unity)** $\sum_i N_i^k(x) = 1$ *for all degrees k.*

(b) **(Affine Invariance)** *Let $(b_i)_{i\in\mathbb{Z}}$ be a sequence of knots defined by $b_i = c_1 a_i + c_2$, and let $\hat{N}_i^k$ denote the B-splines defined over these knots. Then*

$$N_i^k(x) = \hat{N}_i^k(c_1 x + c_2) \tag{4.108}$$

for all degrees k.

(c) For $k \geq 1$,

$$\sum_i c_i N_i^k(x) = \sum_i \left[c_i \frac{x - a_i}{a_{i+k} - a_i} + c_{i-1}\left(1 - \frac{x - a_i}{a_{i+k} - a_i}\right)\right] N_i^{k-1}(x). \tag{4.109}$$

In the case of cardinal B-splines where $a_{i+1} - a_i = h$ for all i, this takes a simpler form:

$$\sum_i c_i N_i^k(x) = \sum_i \left[c_i \frac{x - a_i}{kh} + c_{i-1}\left(1 - \frac{x - a_i}{kh}\right)\right] N_i^{k-1}(x). \tag{4.110}$$

(d) For a cardinal B-spline N_i^k of any degree k, let $M = \frac{1}{2}(a_i + a_{i+k+1})$ be the midpoint of its support. Then

$$N_i^k(x) = N_i^k(M - (x - M)). \tag{4.111}$$

Property (a) is a standard property, while properties (b) and (d) can be proved by induction. Property (c) is simply an application of the recursive formula followed by a re-indexing of the second term in the series.

We shall work with the knot sequences $(a_i^n)_{i\in\mathbb{Z}}$ defined by

$$a_i^n = x_i - \frac{n+1}{2}\Delta x \tag{4.112}$$

for $n \in \mathbb{N}$. We use the notation $N_i^{n,k}$ to denote the B-splines of degree k defined by these knots. Note that these are cardinal B-splines with spacing Δx since x_i is uniform. The motivation for this choice of knots is the following identity:

$$N_i^{n,n}(x) = N\left(\frac{x_i - x}{\Delta x}\right), \tag{4.113}$$

where N is the degree n B-spline defined over the knots $t_j = j - \frac{n+1}{2}$ for $j = 0, 1, \ldots, n+1$. This is a consequence of properties (b) and (d). Since $\frac{a_{i+j}-x_i}{\Delta x} = t_j$ for $j = 0, 1, \ldots, n+1$, property (b) implies

$$N_i^{n,n}(x) = N\left(\frac{x - x_i}{\Delta x}\right). \tag{4.114}$$

Next, since N is supported on the interval $(-\frac{n+1}{2}, \frac{n+1}{2})$ whose midpoint is 0, we have

$$N_i^{n,n}(x) = N\left(\frac{x - x_i}{\Delta x}\right) = N\left(\frac{x_i - x}{\Delta x}\right) \tag{4.115}$$

from property (d).

We may therefore write our earlier sums in the form

$$\sum_i (x_i - x_p)^\alpha N\left(\frac{x_i - x_p}{\Delta x}\right) = \sum_i (x_i - x_p)^\alpha N_i^{n,n}(x_p). \tag{4.116}$$

We record one further useful consequence of this choice of knots, combined with property (b).

Corollary 2. *For any $n \geq 1$ and any degree k,*

$$N_i^{n,k}(x) = N_i^{n-1,k}\left(x + \frac{\Delta x}{2}\right). \tag{4.117}$$

4.2.1.2 Linear Reproduction Property of B-splines

Before we prove the linear reproduction property, we first prove an intermediate result.

Lemma 3. *For $n \geq 1$, the following identity holds for all x:*

$$\sum_i x_i N_i^{n,n}(x) = \frac{x}{n} - \frac{n-1}{2n}\Delta x + \frac{n-1}{n}\sum_i x_i N_i^{n,n-1}(x). \tag{4.118}$$

Proof. Applying property (c) yields

$$\begin{aligned}
\sum_i x_i N_i^{n,n}(x) &= \sum_i \left[x_i \frac{x - a_i^n}{n\Delta x} + (x_i - \Delta x)\left(1 - \frac{x - a_i^n}{n\Delta x}\right)\right] N_i^{n,n-1}(x) \\
&= \sum_i \left[x_i - \Delta x \left(1 - \frac{x - a_i^n}{n\Delta x}\right)\right] N_i^{n,n-1}(x) \\
&= \sum_i \left[x_i - \Delta x \frac{\frac{n-1}{2}\Delta x + (x_i - x)}{n\Delta x}\right] N_i^{n,n-1}(x) \\
&= \sum_i \left[\frac{n-1}{n} x_i + \frac{x}{n} - \frac{n-1}{2n}\Delta x\right] N_i^{n,n-1}(x) \\
&= \frac{x}{n} - \frac{n-1}{2n}\Delta x + \frac{n-1}{n}\sum_i x_i N_i^{n,n-1}(x),
\end{aligned} \tag{4.119}$$

where the last line follows from the partition of unity. □

Proposition 4. *For $n \geq 1$, the following identity holds for all x:*

$$\sum_i x_i N_i^{n,n}(x) = x. \tag{4.120}$$

Proof. We shall proceed by induction. The base case $n = 1$ follows immediately from Lemma 3.

Now suppose that

$$\sum_i x_i N_i^{n,n}(x) = x \tag{4.121}$$

for all x for some $n \geq 1$. Then, applying Lemma 3 and the inductive hypothesis, we get

$$\begin{aligned}
\sum_i x_i N_i^{n+1,n+1}(x) &= \frac{x}{n+1} - \frac{n}{2(n+1)}\Delta x + \frac{n}{n+1}\sum_i x_i N_i^{n+1,n}(x) \\
&= \frac{x}{n+1} - \frac{n}{2(n+1)}\Delta x + \frac{n}{n+1}\sum_i x_i N_i^{n,n}\left(x + \frac{\Delta x}{2}\right) \\
&= \frac{x}{n+1} - \frac{n}{2(n+1)}\Delta x + \frac{n}{n+1}\left(x + \frac{\Delta x}{2}\right) \\
&= \frac{n+1}{n+1}x + \left(\frac{n}{2(n+1)} - \frac{n}{2(n+1)}\right)\Delta x \\
&= x, \qquad (4.122)
\end{aligned}$$

where we have used Corollary 2 on the second line. The result thus follows for all $n \geq 1$ by induction. □

The sum 4.104 is an immediate consequence of Proposition 4.

4.2.1.3 Diagonal Terms of $\mathbf{D}_p$

We now verify the sum 4.105. We first perform a calculation similar to that of Lemma 3.

Lemma 5. *For $n \geq 2$, the following identity holds for all x:*

$$\sum (x_i - x)^2 N_i^{n,n}(x) = \frac{1}{2n}\Delta x^2 + \frac{n-2}{n}\sum_i (x_i - x)^2 N_i^{n,n-1}(x). \qquad (4.123)$$

Proof. Using the same strategy as in Lemma 3, we get

$$\begin{aligned}
\sum_i (x_i - x)^2 N_i^{n,n}(x) &= \sum_i \left[(x_i - x)^2 \frac{x - a_i^n}{n\Delta x} + (x_i - x - \Delta x)^2 \left(1 - \frac{x - a_i^n}{n\Delta x}\right) \right] N_i^{n,n-1}(x) \\
&= \sum_i \left[(x_i - x)^2 + (\Delta x^2 - 2(x_i - x)\Delta x) \left(1 - \frac{x - a_i^n}{n\Delta x}\right) \right] N_i^{n,n-1}(x) \\
&= \sum_i \left[(x_i - x)^2 + \frac{(\Delta x - 2(x_i - x)) \left(\frac{n-1}{2}\Delta x + (x_i - x)\right)}{n} \right] N_i^{n,n-1}(x) \\
&= \sum_i \left[\frac{n-2}{n}(x_i - x)^2 - \frac{n-2}{n}\Delta x(x_i - x) + \frac{n-1}{2n}\Delta x^2 \right] N_i^{n,n-1}(x) \\
&= \frac{n-1}{2n}\Delta x^2 + \sum_i \left[\frac{n-2}{n}(x_i - x)^2 - \frac{n-2}{n}\Delta x(x_i - x) \right] N_i^{n,n-1}(x).
\end{aligned} \tag{4.124}$$

Applying Corollary 2 and Proposition 4 (valid since $n-1 \geq 1$) to the second term, it becomes

$$\begin{aligned}
-\frac{n-2}{n}\Delta x \sum_i (x_i - x) N_i^{n,n-1}(x) &= -\frac{n-2}{n}\Delta x \sum_i (x_i - x) N_i^{n-1,n-1}\left(x + \frac{\Delta x}{2}\right) \\
&= -\frac{n-2}{n}\Delta x \left(x + \frac{\Delta x}{2} - x\right) \\
&= -\frac{n-2}{2n}\Delta x^2.
\end{aligned} \tag{4.125}$$

Using this we can simply the previous expression to get

$$\begin{aligned}
\sum_i (x_i - x)^2 N_i^{n,n}(x) &= \frac{n-1}{2n}\Delta x^2 - \frac{n-2}{2n}\Delta x^2 + \frac{n-2}{n}\sum_i (x_i - x)^2 N_i^{n,n-1}(x) \\
&= \frac{1}{2n}\Delta x^2 + \frac{n-2}{n}\sum_i (x_i - x)^2 N_i^{n,n-1}(x).
\end{aligned} \tag{4.126}$$

□

Proposition 6. *For $n \geq 2$, the following identity holds for all x:*

$$\sum_i (x_i - x)^2 N_i^{n,n}(x) = \frac{n+1}{12}\Delta x^2. \tag{4.127}$$

Proof. We proceed again by induction. The base case $n = 2$ is true by an application of Marsden's identity and the linear reproduction property; alternatively, it also follows from Lemma 5.

Now suppose that

$$\sum_i (x_i - x)^2 N_i^{n,n}(x) = \frac{n+1}{12}\Delta x^2 \tag{4.128}$$

for all x for some $n \geq 2$. Consider the $n+1$ term. Applying Lemma 5, we obtain

$$\begin{aligned}\sum_i (x_i - x)^2 N_i^{n+1,n+1}(x) &= \frac{1}{2(n+1)}\Delta x^2 + \frac{n-1}{n+1}\sum_i (x_i - x)^2 N_i^{n+1,n}(x) \\ &= \frac{1}{2(n+1)}\Delta x^2 + \frac{n-1}{n+1}\sum_i (x_i - x)^2 N_i^{n,n}\left(x + \frac{\Delta x}{2}\right).\end{aligned} \tag{4.129}$$

We then expand the coefficient of $N_i^{n,n}$ in the second term and apply the inductive hypothesis to get

$$\begin{aligned}\sum_i (x_i - x)^2 N_i^{n,n}\left(x + \frac{\Delta x}{2}\right) &= \sum_i \left(x_i - \left(x + \frac{\Delta x}{2}\right) + \frac{\Delta x}{2}\right)^2 N_i^{n,n}\left(x + \frac{\Delta x}{2}\right) \\ &= \frac{1}{4}\Delta x^2 + \sum_i \left(x_i - \left(x + \frac{\Delta x}{2}\right)\right)^2 N_i^{n,n}\left(x + \frac{\Delta x}{2}\right) \\ &= \frac{1}{4}\Delta x^2 + \frac{n+1}{12}\Delta x^2 \\ &= \frac{n+4}{12}\Delta x^2,\end{aligned} \tag{4.130}$$

where the sum over the coefficients $\Delta x\left(x_i - \left(x + \frac{\Delta x}{2}\right)\right)$ is 0 by the linear reproduction property. Then,

$$\begin{aligned}\sum_i (x_i - x)^2 N_i^{n+1,n+1}(x) &= \frac{1}{2(n+1)}\Delta x^2 + \frac{n-1}{n+1}\frac{n+4}{12}\Delta x^2 \\ &= \frac{n+2}{12}\Delta x^2.\end{aligned} \tag{4.131}$$

Thus, the result holds for all $n \geq 2$ by induction. □

On every diagonal of $\mathbf{D}_p$, all sums except for one in the product evaluate to 1, and the final sum is of the third case. Hence $\mathbf{D}_p = \frac{n+1}{12}\Delta x^2 \mathbf{I}$.

While it is not recommended to use linear B-splines, we include the calculation of the diagonal entries of $\mathbf{D}_p$ for the case $n = 1$ for completeness. While $\mathbf{D}_p$ remains diagonal in this case, it will not be a constant multiple of the identity.

Proposition 7.

$$\sum_i (x_i - x)^2 N_i^{1,1}(x) = (x_{i'} - x)(x - x_{i'-1}), \tag{4.132}$$

where i' is the unique index satisfying

$$x_{i'-1} \leq x < x_{i'}. \tag{4.133}$$

Proof. Following the proof of Lemma 5, we have

$$\begin{aligned}
\sum_i (x_i - x)^2 N_i^{1,1}(x) &= \sum_i \left[(x_i - x)^2 \frac{x - a_i^1}{\Delta x} + (x_i - x - \Delta x)^2 \left(1 - \frac{x - a_i^1}{\Delta x}\right) \right] N_i^{1,0}(x) \\
&= \sum_i \left[(x_i - x)^2 + (\Delta x^2 - 2(x_i - x)\Delta x) \left(1 - \frac{x - a_i^1}{\Delta x}\right) \right] N_i^{1,0}(x) \\
&= \sum_i \left[(x_i - x)^2 + (\Delta x - 2(x_i - x))(x_i - x) \right] N_i^{1,0}(x) \\
&= \sum_i \left[\Delta x (x_i - x) - (x_i - x)^2 \right] N_i^{1,0}(x) \\
&= \sum_i (x_i - x)(\Delta x - (x_i - x)) N_i^{1,0}(x) \\
&= \sum_i (x_i - x)(x - x_{i-1}) N_i^{1,0}(x).
\end{aligned} \tag{4.134}$$

Every term in this sum is 0 except for the i' term, where i' is the unique index such that $x \in \left[a_{i'}^1, a_{i'+1}^1\right)$. For this term, $N_i^{1,0}(x)$ simply evaluates to 1. Hence,

$$\sum_i (x_i - x)^2 N_i^{1,1}(x) = (x_{i'} - x)(x - x_{i'-1}). \tag{4.135}$$

Lastly, $a_i^1 = x_i - \Delta x = x_{i-1}$ and $x \in \left[a_{i'}^1, a_{i'+1}^1\right)$ implies that

$$x_{i'-1} \leq x < x_{i'}. \tag{4.136}$$

We note that there are simpler ways of performing this calculation since the B-splines $N_i^{1,1}$ have a simple form, but this calculation matches the style of the rest of this section. □

Hence, $D_{p\gamma\gamma}$ will depend on $x_{p\gamma}$.

REFERENCES

[ACW06] A. Angelidis, M.-P. Cani, G. Wyvill, and S. King. "Swirling-sweepers: Constant-volume modeling." *Graph. Models*, **68**(4):324–332, 2006.

[AN05] A. Angelidis and F. Neyret. "Simulation of smoke based on vortex filament primitives." In *Proc 2005 ACM SIGGRAPH/Eurographics Symp Comp Anim*, pp. 87–96. ACM, 2005.

[Att10] M. Attene. "A lightweight approach to repairing digitized polygon meshes." *The visual computer*, **26**(11):1393–1406, 2010.

[ATW13] R. Ando, N. Thürey, and C. Wojtan. "Highly adaptive liquid simulations on tetrahedral meshes." *ACM Trans Graph*, **32**(4):103:1–103:10, 2013.

[BB99] T. Belytschko and T. Black. "Elastic crack growth in finite elements with minimal remeshing." *Int. J. Num. Meth. Engr.*, **45**(5):601–620, 1999.

[BB08] C. Batty and R. Bridson. "Accurate viscous free surfaces for buckling, coiling, and rotating liquids." *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 219–228, 2008.

[BB12] L. Boyd and R. Bridson. "MultiFLIP for energetic two-phase fluid simulation." *ACM Trans Graph*, **31**(2):16:1–16:12, 2012.

[BBB07] C. Batty, F. Bertails, and R. Bridson. "A fast variational framework for accurate solid-fluid coupling." *ACM Trans Graph*, **26**(3), 2007.

[BDS18] G. Barill, N. Dickson, R. Schmidt, D. Levin, and A. Jacobson. "Fast winding numbers for soups and clouds." *ACM Trans. Graph.*, **37**(4):1–12, 2018.

[Bez70] P. Bézier. "Numerical control: mathematics and applications." 1970.

[BFG20] H. Brönnimann, A. Fabri, G.-J. Giezeman, S. Hert, M. Hoffmann, L. Kettner, S. Pion, and S. Schirra. "2D and 3D Linear Geometry Kernel." In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.2 edition, 2020.

[BGV09] T. Belytschko, R. Gracie, and G. Ventura. "A review of extended/generalized finite element methods for material modeling." *Mod Sim Mat Sci Eng*, **17**(4):043001, 2009.

[BH07] Y. Bazilevs and T. Hughes. "Weak imposition of Dirichlet boundary conditions in fluid mechanics." *Comp Fluids*, **36**(1):12–26, 2007.

[BL03] D.C. Banks and S. Linton. "Counting cases in marching cubes: toward a generic algorithm fo rproducing substitopes." In *IEEE Visualization, 2003. VIS 2003.*, pp. 51–58, 2003.

[Bla67] S. Blank. *Extending immersions of the circle.* PhD thesis, Brandeis University, Waltham, Mass., 1967.

[BM82] J. Bates and A. McDonald. "Multiply-upstream, semi-Lagrangian advective schemes: Analysis and application to a multi-level primitive equation model." *Monthly Weather Review*, **110**(12):1831–1842, 1982.

[Bot02] O. Botella. "On a collocation B-spline method for the solution of the Navier–Stokes equations." *Comp Fl*, **31**(4-7):397–420, 2002.

[Bre10] A. Bressan. "Isogeometric regular discretization for the Stokes problem." *IMA J Num Anal*, **31**(4):1334–1356, 2010.

[Bri08] R. Bridson. *Fluid simulation for computer graphics.* Taylor & Francis, 2008.

[BSM20] M. Botsch, D. Sieger, P. Moeller, and A. Fabri. "Surface Mesh." In *CGAL User and Reference Manual.* CGAL Editorial Board, 5.2 edition, 2020.

[BWC04] P. Bhaniramka, R. Wenger, and R. Crawfis. "Isosurface construction in any dimension using convex hulls." *IEEE Transactions on Visualization and Computer Graphics*, **10**(2):130–141, 2004.

[BWS09] A. Brunton, S. Wuhrer, C. Shu, P. Bose, and E. Demaine. "Filling holes in triangular meshes by curve unfolding." In *2009 IEEE International Conference on Shape Modeling and Applications*, pp. 66–72, 2009.

[CBE15] M. Cong, M. Bao, J. E, K. Bhat, and R. Fedkiw. "Fully automatic generation of anatomical face simulation models." In *Proc ACM SIGGRAPH/Eurographics Symp Comp Anim*, pp. 175–183, 2015.

[CBF16] M. Cong, L. Bhat, and R. Fedkiw. "Art-Directed Muscle Simulation for High-End Facial Animation." In *Proc 2016 ACM SIGGRAPH/Eurographics Symp Comp Anim*, pp. 119–127. Eurographics Association, 2016.

[Cha77] G. Chartrand. *Introductory graph theory.* Courier Corporation, 1977.

[Che95] E. Chernyaev. "Marching Cubes 33: Construction of Topologically Correct Isosurfaces." Technical report, Institute for High Energy Physics, 1995.

[Cho67] A. Chorin. "A numerical method for solving incompressible viscous flow problems." *J Comp Phys*, **2**(1):12–26, 1967.

[CIR52] R. Courant, E. Isaacson, and M. Rees. "On the solution of nonlinear hyperbolic differential equations by finite differences." *Comm Pure App Math*, **5**(3):243–255, 1952.

[CK12] W. Cheney and D. Kincaid. *Numerical mathematics and computing*. Cengage Learning, 2012.

[CKP16] A. Chern, F. Knöppel, U. Pinkall, P. Schröder, and S. Weißmann. "Schrödinger's smoke." *ACM Trans Graph (TOG)*, **35**(4):77, 2016.

[CMK15] N. Chentanez, M. Müller, and T. Kim. "Coupling 3D eulerian, heightfield and particle methods for interactive simulation of large scale liquid phenomena." *IEEE Trans Vis Comp Graph*, **21**(10):1116–1128, 2015.

[CWB01] F. Cheng, X. Wang, and B. Barsky. "Quadratic B-spline curve interpolation." *Comp Math App*, **41**(1-2):39–50, 2001.

[DCB13] C. Doran, A. Chang, and R. Bridson. "Isosurface Stuffing Improved: Acute Lattices and Feature Matching." In *ACM SIGGRAPH 2013 Talks*, SIGGRAPH '13, New York, NY, USA, 2013. Association for Computing Machinery.

[Dem19] D. Demidov. "AMGCL: An Efficient, Flexible, and Extensible Algebraic Multigrid Implementation." *Lobachevskii Journal of Mathematics*, **40**(5):535–546, 5 2019.

[EB14] E. Edwards and R. Bridson. "Detailed water with coarse grids: combining surface meshes and adaptive Discontinuous Galerkin." *ACM Trans Graph*, **33**(4):136:1–136:9, 2014.

[EFW20] P. Evans, B. Fasy, and C. Wenk. "Combinatorial Properties of Self-Overlapping Curves and Interior Boundaries." In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 41:1–41:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[EM09] D. Eppstein and E. Mumford. "Self-overlapping curves revisited." In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 160–169. SIAM, 2009.

[ENG03] D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw. "Using the particle level set method and a second order accurate pressure boundary condition for free surface flows." In *ASME/JSME 2003 4th Joint Fluids Summer Engineering Conference*, pp. 337–342. ASMEDC, 2003.

[Eva10] L. Evans. *Partial differential equations.* American Mathematical Society, Providence, R.I., 2010.

[FAW16] F. Ferstl, R. Ando, C. Wojtan, R. Westermann, and N. Thuerey. "Narrow Band FLIP for Liquid Simulations." In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics*, EG '16, p. 225–232, Goslar, DEU, 2016. Eurographics Association.

[FF98] M. Falcone and R. Ferretti. "Convergence analysis for a class of high-order semi-Lagrangian advection schemes." *SIAM J Num Anal*, **35**(3):909–940, 1998.

[FF01] N. Foster and R. Fedkiw. "Practical animation of liquids." In *Proc 28th SIGGRAPH*, pp. 23–30. ACM, 2001.

[FGG17] C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran. "A Polynomial Particle-in-cell Method." *ACM Trans Graph*, **36**(6):222:1–222:12, November 2017.

[FM96] N. Foster and D. Metaxas. "Realistic animation of liquids." *Graph Mod Imag Proc*, **58**:471–483, 1996.

[FSJ01] R. Fedkiw, J. Stam, and H. Jensen. "Visual simulation of smoke." In *SIGGRAPH*, pp. 15–22. ACM, 2001.

[FTS06] W. Von Funck, H. Theisel, and H.-P. Seidel. "Vector field based shape deformations." *ACM Trans. Graph.*, **25**(3):1118–1125, 2006.

[FWD14] F. Ferstl, R. Westermann, and C. Dick. "Large-scale liquid simulation on adaptive hexahedral grids." *IEEE Trans Vis Comp Graph*, **20**(10):1405–1417, 2014.

[GC11] J. Graver and G. Cargo. "When Does a Curve Bound a Distorted Disk?" *SIAM Journal on Discrete Mathematics*, **25**(1):280–305, 2011.

[GCX20] J. Gao, W. Chen, T. Xiang, A. Jacobson, M. McGuire, and S. Fidler. "Learning Deformable Tetrahedral Meshes for 3D Reconstruction." In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pp. 9936–9947. Curran Associates, Inc., 2020.

[GD01] J. Gain and N. Dodgson. "Preventing self-intersection under free-form deformation." *IEEE Trans Viz Comp Grap*, **7**(4):289–298, 2001.

[GHM20a] S. Gagniere, D. Hyde, A. Marquez-Razon, C. Jiang, Z. Ge, X. Han, Q. Guo, and J. Teran. "A Hybrid Lagrangian/Eulerian Collocated Velocity Advection and Projection Method for Fluid Simulation." *Computer Graphics Forum*, **39**(8):1–14, 2020.

[GHM20b] S. Gagniere, D. Hyde, A. Marquez-Razon, C. Jiang, Z. Ge, X. Han, Q. Guo, and J. Teran. "Supplementary Technical Document." Technical report, 2020.

[GS08] O. Gonzalez and A. M. Stuart. *A First Course in Continuum Mechanics*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2008.

[GTG14] M. T. Goodrich, R. Tamassia, and M. H. Goldwasser. *Data Structures and Algorithms in Java*. Wiley Publishing, 6th edition, 2014.

[HCB05] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement." *Comp Meth App Mech Eng*, **194**(39,41):4135–4195, 2005.

[HGM20] D.A.B. Hyde, S.W. Gagniere, A. Marquez-Razon, and J. Teran. "An Implicit Updated Lagrangian Formulation for Liquids with Large Surface Energy." *ACM Trans Graph*, **39**(6), November 2020.

[HL95] Z.-J. Hu and Z.-K. Ling. "Geometric modeling of a moving object with self-intersection." In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 17162, pp. 141–148. American Society of Mechanical Engineers, 1995.

[HPS11] D. Harmon, D. Panozzo, O. Sorkine, and D. Zorin. "Interference-aware geometric modeling." *ACM Transactions on Graphics (TOG)*, **30**(6):1–10, 2011.

[HSW20] Y. Hu, T. Schneider, B. Wang, D. Zorin, and D. Panozzo. "Fast tetrahedral meshing in the wild." *ACM Trans. Graph.*, **39**(4):117–1, 2020.

[HT89] W. Horn and D. Taylor. "A theorem to determine the spatial containment of a point in a planar polyhedron." *Comp Vis Graph Imag Proc*, **45**(1):106–116, 1989.

[Hua94] C. Huang. "Semi-Lagrangian advection schemes and Eulerian WKL algorithms." *Monthly Weather Review*, **122**(7):1647–1658, 1994.

[Hug12] T. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[HZG18] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo. "Tetrahedral Meshing in the Wild." *ACM Trans. Graph.*, **37**(4):60:1–60:14, July 2018.

[IAA12] M. Ihmsen, N. Akinci, G. Akinci, and M. Teschner. "Unified spray, foam and air bubbles for particle-based fluids." *Vis Comp*, **28**(6-8):669–677, 2012.

[JAY15] C. Jamin, P. Alliez, M. Yvinec, and J.-D. Boissonnat. "CGALmesh: a generic framework for delaunay mesh generation." *ACM Trans. Math. Soft.*, **41**(4):1–24, 2015.

[JKS13] A. Jacobson, L. Kavan, and O. Sorkine-Hornung. "Robust inside-outside segmentation using generalized winding numbers." *ACM Trans. Graph.*, **32**(4):1–12, 2013.

[JSS15] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. "The Affine Particle-In-Cell Method." *ACM Trans Graph*, **34**(4):51:1–51:10, 2015.

[JST16] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. "The Material Point Method for Simulating Continuum Materials." In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery.

[KBT17] D. Koschier, J. Bender, and N. Thuerey. "Robust eXtended Finite Elements for complex cutting of deformables." *ACM Trans Graph*, **36**(4):55:1–55:13, 2017.

[Kim98] E. Kim. "A mixed Galerkin method for computing the flow between eccentric rotating cylinders." *Int J Num Meth Fl*, **26**(8):877–885, 1998.

[KLL05] B. Kim, Y. Liu, I. Llamas, and J. Rossignac. "FlowFixer: Using BFECC for Fluid Simulation." In *Proc Eurograph Conf Nat Phen*, pp. 51–56. Eurographics Association, 2005.

[KLL06] B. Kim, Y. Liu, I. Llamas, and J. Rossignac. "Advections with significantly reduced dissipation and diffusion." *IEEE Trans Viz Comp Graph*, **13**(1):135–144, 2006.

[KMS99] A. Kravchenko, P. Moin, and K. Shariff. "B-spline method and zonal grids for simulations of complex turbulent flows." *J Comp Phys*, **151**(2):757–789, 1999.

[KSB12] M. Kazhdan, J. Solomon, and M. Ben-Chen. "Can Mean-Curvature Flow Be Modified to Be Non-Singular?" *Comput. Graph. Forum*, **31**(5):1745–1754, August 2012.

[KSK08] D. Kim, O. Song, and H. Ko. "A Semi-Lagrangian CIP Fluid Solver without Dimensional Splitting." *Computer Graphics Forum*, **27**(2):467–475, 2008.

[KT10] H.-J. Kim and T. Tautges. "EBMesh: An Embedded Boundary Meshing Tool." In Suzanne Shontz, editor, *Proceedings of the 19th International Meshing Roundtable*, pp. 227–242, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[KW90] H. Kuo and R. Williams. "Semi-Lagrangian solutions to the inviscid Burgers equation." *Monthly Weather Review*, **118**(6):1278–1288, 1990.

[LB18] Y. Li and J. Barbič. "Immersion of Self-Intersecting Solids and Surfaces." *ACM Trans. Graph.*, **37**(4), July 2018.

[LBB17] E. Larionov, C. Batty, and R. Bridson. "Variational stokes: A unified pressure-viscosity solver for accurate viscous liquids." *ACM Trans Graph*, **36**(4), 2017.

[LC87] W. Lorensen and H. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm." *SIGGRAPH Comput. Graph.*, **21**:163–169, August 1987.

[Leo79] B. Leonard. "A stable and accurate convective modelling procedure based on quadratic upstream interpolation." *Computer methods in applied mechanics and engineering*, **19**(1):59–98, 1979.

[Li11] W. Li. "Detecting Ambiguities in 3D Polygons with Self-Intersecting Projections." In *2011 12th International Conference on Computer-Aided Design and Computer Graphics*, pp. 11–16, 2011.

[LLV03] T. Lewiner, H. Lopes, A.W. Vieira, and G. Tavares. "Efficient Implementation of Marching Cubes' Cases with Topological Guarantees." *Journal of Graphics Tools*, **8**(2):1–15, 2003.

[LRT20] S. Loriot, M. Rouxel-Labbé, J. Tournois, and I. Yaz. "Polygon Mesh Processing." In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.2 edition, 2020.

[LS07] F. Labelle and J. Shewchuk. "Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles." In *ACM SIGGRAPH 2007*, SIGGRAPH '07, pp. 57–es, New York, NY, USA, 2007. ACM.

[Mar74] M. Marx. "Extensions of normal immersions of $\mathcal{S}^1$ into $\mathcal{R}^2$." *Transactions of the American Mathematical Society*, **187**:309–326, 1974.

[MAS15] N. Mitchell, M. Aanjaneya, R. Setaluri, and E. Sifakis. "Non-Manifold Level Sets: A Multivalued Implicit Surface Representation with Applications to Self-Collision Processing." *ACM Trans. Graph.*, **34**(6), October 2015.

[MBF03] N. Molino, R. Bridson, and R. Fedkiw. "Tetrahedral mesh generation for deformable bodies." In *Proc. Symposium on Computer Animation*, p. 8, 2003.

[MBF04] N. Molino, Z. Bao, and R. Fedkiw. "A virtual node algorithm for changing mesh topology during simulation." *ACM Trans Graph*, **23**(3):385–392, 2004.

[MBT03] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. "A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra." In *Int Mesh Round*, pp. 103–114. Citeseer, 2003.

[MCP08] J. Molemaker, J. Cohen, S. Patel, and J. Noh. "Low viscosity flow simulations for animation." In *Proc 2008 ACM SIGGRAPH/Eurographics Symp Comp Anim*, pp. 9–18. Eurographics Association, 2008.

[MCP09] P. Mullen, K. Crane, D. Pavlov, Y. Tong, and M. Desbrun. "Energy-Preserving Integrators for Fluid Animation." *ACM Trans. Graph.*, **28**(3), jul 2009.

[MK96] P. Makar and S. Karpik. "Basis-spline interpolation on the sphere: Applications to semi-Lagrangian advection." *Monthly Weather Review*, **124**(1):182–199, 1996.

[Muk14] U. Mukherjee. "Self-overlapping curves: Analysis and applications." *Computer-Aided Design*, **46**:227–232, 2014.

[NSB18] M. B. Nielsen, K. Stamatelos, M. Bojsen-Hansen, D. Brinsmead, Y. Pomerleau, M. Nordenstam, and R. Bridson. "A Collocated Spatially Adaptive Approach to Smoke Simulation in Bifrost." In *ACM SIGGRAPH 2018 Talks*, SIGGRAPH '18, New York, NY, USA, 2018. Association for Computing Machinery.

[NY06] T.S. Newman and H. Yi. "A survey of the marching cubes algorithm." *Computers & Graphics*, **30**(5):854–879, 2006.

[NZT19] R. Narain, J. Zehnder, and B. Thomaszewski. "A Second-order advection-reflection solver." *Proc ACM Comput Graph Interact Tech*, **2**(2), July 2019.

[OF03] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces.* Applied mathematical science. Springer, New York, N.Y., 2003.

[PBP02] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-spline techniques*, volume 6. Springer, 2002.

[PK05] S. Park and M. Kim. "Vortex fluid for gaseous phenomena." In *Proc 2005 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 261–270. ACM, 2005.

[PS84] J. Pudykiewicz and A. Staniforth. "Some properties and comparative performance of the semi-Lagrangian method of Robert in the solution of the advection-diffusion equation." *Atmosphere-Ocean*, **22**(3):283–308, 1984.

[QZG19] Z. Qu, X. Zhang, M. Gao, C. Jiang, and B. Chen. "Efficient and conservative fluids using bidirectional mapping." *ACM Trans. Graph.*, **38**(4), 2019.

[RC12] T. Rüberg and F. Cirak. "Subdivision-stabilised immersed b-spline finite elements for moving boundary flows." *Comp Meth App Mech Eng*, **209**:266–283, 2012.

[RCL98] L. Riishøjgaard, S. Cohn, Y. Li, and R. Ménard. "The use of spline interpolation in semi-Lagrangian transport models." *Monthly Weather Review*, **126**(7):2008–2016, 1998.

[RH99] J. C. Roberts and S. Hill. "Piecewise linear hypersurfaces using the marching cubes algorithm." In Robert F. Erbacher, Philip C. Chen, and Craig M. Wittenbrink, editors, *Visual Data Exploration and Analysis VI*, volume 3643, pp. 170 – 181. International Society for Optics and Photonics, SPIE, 1999.

[Rob81] A. Robert. "A stable numerical integration scheme for the primitive meteorological equations." *Atm Ocean*, **19**(1):35–46, 1981.

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, USA, 2nd edition, 2003.

[Saw63] J. Sawyer. "A semi-Lagrangian method of solving the vorticity advection equation." *Tellus*, **15**(4):336–342, 1963.

[SB09] J.-H. Song and T. Belytschko. "Cracking node method for dynamic fracture with finite elements." *Int. J. Num. Meth. Engr.*, **77**(3):360–385, 2009.

[SB12] E. Sifakis and J. Barbic. "FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction." In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, pp. 20:1–20:50, New York, NY, USA, 2012. ACM.

[SBI18] T. Sato, C. Batty, T. Igarashi, and R. Ando. "Spatially adaptive long-term semi-Lagrangian method for accurate velocity advection." *Comp Vis Med*, **4**(3):223–230, 2018.

[SC91] A. Staniforth and J. Côté. "Semi-Lagrangian integration schemes for atmospheric models-A review." *Monthly weather review*, **119**(9):2206–2223, 1991.

[SDF07] E. Sifakis, K. Der, and R. Fedkiw. "Arbitrary cutting of deformable tetrahedralized objects." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 73–80, 2007.

[SDG19] T. Schneider, J. Dumas, X. Gao, M. Botsch, D. Panozzo, and D. Zorin. "Poly-Spline Finite-Element Method." *ACM Trans Graph*, **38**(3):1–16, 2019.

[SFK08] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac. "An unconditionally stable MacCormack method." *J Sci Comp*, **35**(2-3):350–371, 2008.

[SHQ18] C. Shah, D. Hyde, H. Qu, and P. Levis. "Distributing and Load Balancing Sparse Fluid Simulations." *Computer Graphics Forum*, **37**(8):35–46, 2018.

[Si15] H. Si. "TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator." *ACM Trans. Math. Softw.*, **41**(2), February 2015.

[SIB17] T. Sato, T. Igarashi, C. Batty, and R. Ando. "A long-term semi-Lagrangian method for accurate velocity advection." In *SIGGRAPH Asia 2017 Tech Briefs*, p. 5. ACM, 2017.

[SJP13] L. Sacht, A. Jacobson, D. Panozzo, C. Schüller, and O. Sorkine-Hornung. "Consistent Volumetric Discretizations inside Self-Intersecting Surfaces." In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, SGP '13, p. 147–156, Goslar, DEU, 2013. Eurographics Association.

[SKK09] O. Song, D. Kim, and H. Ko. "Derivative particles for simulating detailed movements of fluids." *IEEE Trans Vis Comp Graph*, pp. 247–255, 2009.

[SOS04] C. Shen, J. O'Brien, and J. Shewchuk. "Interpolating and Approximating Implicit Surfaces from Polygon Soup." In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, p. 896–904, New York, NY, USA, 2004. Association for Computing Machinery.

[SP86] T. Sederberg and S. Parry. "Free-form deformation of solid geometric models." In *Proc. 13th Ann. Conf. Comp. Graph. Interactive Techniques*, pp. 151–160, 1986.

[SRF05] A. Selle, N. Rasmussen, and R. Fedkiw. "A Vortex Particle Method for Smoke, Water and Explosions." In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, p. 910–914, New York, NY, USA, 2005. Association for Computing Machinery.

[SSH14] C. Schroeder, A. Stomakhin, R. Howes, and J. Teran. "A second order virtual node algorithm for Navier-Stokes flow problems with interfacial forces and discontinuous material properties." *J Comp Phys*, **265**:221 – 245, 2014.

[Sta99] J. Stam. "Stable Fluids." In *Siggraph*, volume 99, pp. 121–128, 1999.

[STK07] E. Sharif, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun. "Stable, circulation-preserving, simplicial fluids." *ACM Trans Graph (TOG)*, **26**(1):4, 2007.

[SU94] J. Steinhoff and D. Underhill. "Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings." *Phys Fl*, **6**(8):2738–2744, 1994.

[SV92] P. Shor and C. Van Wyk. "Detecting and decomposing self-overlapping curves." *Computational Geometry*, **2**(1):31–50, 1992.

[SWT18] T. Sato, C. Wojtan, N. Thuerey, T. Igarashi, and R. Ando. "Extended narrow band FLIP for liquid simulations." *Comp Graph For*, 2018.

[TBF19] M. Tao, C. Batty, E. Fiume, and D. Levin. "Mandoline: Robust Cut-Cell Generation for Arbitrary Triangle Meshes." *ACM Trans. Graph.*, **38**(6), November 2019.

[TH73] C. Taylor and P. Hood. "A numerical solution of the Navier-Stokes equations using the finite element technique." *Comp & Fl*, **1**(1):73–100, 1973.

[The20] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.2 edition, 2020.

[Tit61] C. Titus. "The combinatorial topology of analytic functions of the boundary of a disk." *Acta Mathematica*, **106**(1-2):45–64, 1961.

[TP11] J. Tessendorf and B. Pelfrey. "The characteristic map for fast and efficient vfx fluid simulations." In *Computer Graphics International Workshop on VFX, Computer Animation, and Stereo Movies. Ottawa, Canada*, 2011.

[TSB05] J. Teran, E. Sifakis, S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. "Creating and simulating skeletal muscle from the visible human data set." *IEEE Trans Vis Comp Graph*, **11**(3):317–328, 2005.

[WDG19] S. Wang, M. Ding, T. Gast, L. Zhu, S. Gagniere, C. Jiang, and J. Teran. "Simulation and Visualization of Ductile Fracture with the Material Point Method." *Proc. ACM Comput. Graph. Interact. Tech.*, **2**(2), jul 2019.

[WJS14] Y. Wang, C. Jiang, C. Schroeder, and J. Teran. "An adaptive virtual node algorithm with robust mesh cutting." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 77–85. Eurographics Association, 2014.

[WL10] J. Wang and A. Layton. "New numerical methods for Burgers' equation based on semi-Lagrangian and modified equation approaches." *App Num Math*, **60**(6):645–657, 2010.

[WP10] S. Weißmann and U. Pinkall. "Filament-Based Smoke with Vortex Shedding and Variational Reconnection." In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA, 2010. Association for Computing Machinery.

[WWD15] J. Wu, R. Westermann, and C. Dick. "A survey of physically based simulation of cuts in deformable bodies." *Comp Graph Forum*, **34**(6):161–187, 2015.

[XK01] D. Xiu and G. Karniadakis. "A semi-Lagrangian high-order method for Navier–Stokes equations." *J Comp Phys*, **172**(2):658–684, 2001.

[YXU01] T. Yabe, F. Xiao, and T. Utsumi. "The constrained interpolation profile method for multiphase analysis." *J Comp Phys*, **169**:556–593, 2001.

[ZB05] Y. Zhu and R. Bridson. "Animating sand as a fluid." *ACM Trans Graph*, **24**(3):965–972, 2005.

[ZBG15] X. Zhang, R. Bridson, and C. Greif. "Restoring the missing vorticity in advection-projection fluid solvers." *ACM Trans Graph (TOG)*, **34**(4):52, 2015.

[ZDZ18] J. Zhang, F. Duan, M. Zhou, D. Jiang, X. Wang, Z. Wu, Y. Huang, G. Du, S. Liu, P. Zhou, and X. Shang. "Stable and realistic crack pattern generation using a cracking node method." *Front. Comp. Sci.*, **12**(4):777–797, 2018.

[Zha05] H. Zhao. "A fast sweeping method for eikonal equations." *Math Comp*, **74**(250):603–627, 2005.

[ZNT18] J. Zehnder, R. Narain, and B. Thomaszewski. "An advection-reflection solver for detail-preserving fluid simulation." *ACM Trans Graph (TOG)*, **37**(4):85, 2018.

[ZZS17] F. Zhang, X. Zhang, Y. Sze, Y. Lian, and Y. Liu. "Incompressible material point method for free surface flow." *J Comp Phys*, **330**(C):92–110, 2017.