Generation and Conditioning of Metric Tensor Fields for Design, Analysis, and Manufacturing Applications

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Mechanical Engineering

Ved Vyas

Carnegie Mellon University
Pittsburgh, PA

August 2018

Abstract

Automatic mesh generation is a challenging problem that is motivated by the need for efficient and accurate simulations of physical phenomena. Many demanding problems require mesh elements to have appropriate anisotropy (size and stretching), orientation, and quality to ensure that a solver can converge reasonably and accurately capture the physics involved. One example of this is large deformation structural analysis, where the progressive distortion of mesh elements in many cases causes issues with solution accuracy and convergence — including premature termination. Another example domain is high speed fluid flow. Here, potentially high aspect ratios and suitable orientation are necessary to efficiently capture strong flow features such as shocks, boundary layers, wakes, and free shear layers. These and other mesh characteristics may also help mitigate the formation of numerical artifacts in the solution.

Unfortunately, exact mesh requirements are generally not known *a priori*, but user input, knowledge of the geometry, and *a posteriori* solution-derived information can all be considered in order to control orientation and anisotropy. It turns out that other applications, such as pattern placement for product design and structure generation for additive manufacturing, can similarly benefit from this.

This thesis presents a framework using Riemannian metric tensor fields for controlling orientation and anisotropy information over surfaces and volumes, as well as recipes for employing this framework to solve problems in various application domains: design, analysis, and manufacturing. Within the framework, metric tensors are interpolated (to form metric fields), generated from constraints, analyzed, and conditioned as necessary. Furthermore, we present novel mesh generation schemes that exploit metric fields to produce boundary-aligned or

anisotropic meshes in 2D and 3D. This is all embedded in an anisotropic mesh adaptation process to iteratively improve fluid dynamics simulations.

Finally, the proposed methods are demonstrated through applications in graphic & product design, structural & fluid dynamics, and additive manufacturing.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Professor Kenji Shimada, for his guidance and support both throughout and preceding this thesis work. He introduced me to the interesting problems that can be found in computer aided design and engineering (CAD/CAE), and the chance to work with him is the reason I pursued studies and research along these avenues.

I am indebted to my committee members for their helpful feedback as well as the selfless accommodations they made for me as this thesis work concluded. I would especially like to thank Dr. Bil Kleb for his mentoring, collaboration, and discussion as we embarked on and continued to develop solutions to challenging fluid dynamics problems. On that note, many thanks are also due to the FUN3D team at the NASA Langley Research Center for their insightful discussions and helpful input.

To the researchers and my friends in the Computational Engineering and Robotics Lab: thank you! I learned a lot from all of you and will cherish the good times we've had together over the years. For fear of forgetting anyone, I haven't compiled a "complete" list; but please know you are all appreciated. However, special thanks go to Dr. Soji Yamakawa. His versatile geometry, meshing, and graphics class library provided a solid foundation for many of the methods developed in this thesis.

Last, but not least, I would like to thank my family for their constant support and encouragement throughout my doctoral studies. They have all been there for me over this long journey. It is particularly important to acknowledge the profound impact my parents, Neema Vyas and Dr. Subhash Vyas, have had on me. Together, they introduced me to the world of research as a child and inspired me to do my best to make a meaningful academic contribution.

This work was supported in part by phases I, II, and II-E (enhancement) of a NASA Small Business Technology Transfer (STTR) program: proposal number T8.01-9986 (09-1 for Phase I and 09-2 for Phases II & II-E) and contract numbers NNX10CF76P (Phase I) and NNX11CC71C (Phases II & II-E). Portions of this work were self-supported.

The doctoral committee for this thesis comprised the following members:

Dr. Kenji Shimada, Chair

Department of Mechanical Engineering, Carnegie Mellon University

Dr. Jack Beuth

Department of Mechanical Engineering, Carnegie Mellon University

Dr. Satbir Singh

Department of Mechanical Engineering, Carnegie Mellon University

Dr. Bil Kleb

NASA Langley Research Center

Contents

List of Figures

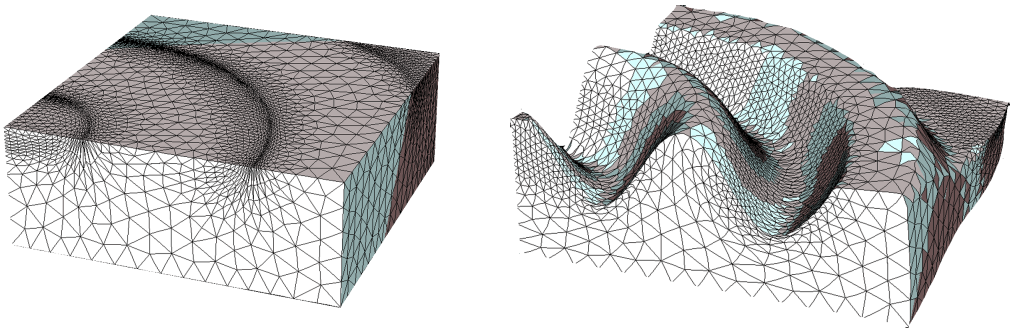
List of Tables

Chapter 1

Introduction

Automatic mesh generation is a challenging problem that is motivated by the need for efficient and accurate simulations of physical phenomena. Many demanding problems require mesh elements to have appropriate anisotropy (size and stretching), orientation, and quality to ensure that a solver can converge reasonably and accurately capture the physics involved.

An example of this is large deformation structural analysis, where the progressive distortion of mesh elements during an analysis may cause issues with solution accuracy and convergence — including premature termination. One approach to this problem is iterative inverse pre-deformation as presented by Dheeravongkit and Shimada [DS05]. That work considers strain error estimates and deformation during a pre-analysis solution run to generate new anisotropy and orientation requirements for an initial mesh, which then undergoes a large deformation process (*e.g.*, metal forming). The solution process is restarted with a newly generated mesh based on the updated requirements. Even one iteration of this method will result in better mesh quality at termination, however it can be repeated to prolong the coverage of the simulated event [DS07]. An example of an inversely pre-deformed mesh in the reference configuration and the final mesh in the deformed configuration is shown in Figure 1.1. In this example, a rectangular blank is driven into a rigid, sinusoidal die. This



(a) Inversely pre-deformed mesh on reference configuration

(b) Mesh in deformed configuration after analysis

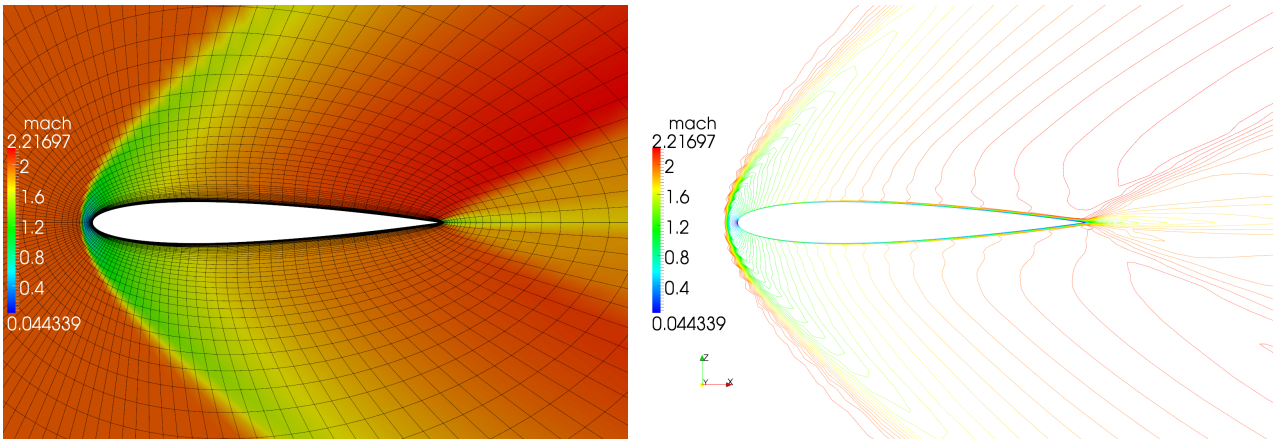
Figure 1.1: Results of inverse pre-deformation (adapted from [DS07])

particular case involves tetrahedral meshes, but the method is also effective on hexahedral-dominant (hex-dominant) meshes in 3D.

Another example concerns supersonic fluid flow. Here, potentially high aspect-ratio elements with suitable orientation are necessary to efficiently capture strong flow features such as shocks, boundary layers, wakes, and free shear layers. Many of these features are demonstrated for supersonic (Mach 2) flow past a NACA 0012 airfoil in Figure 1.2a. That Mach number plot is complemented by a contour plot of the same solution variable in Figure 1.2b. The original mesh for the problem is an *O-type* mesh (as described in [TWM85]) that has been refined near the airfoil boundary. One of the strong flow features is a bow, or detached, shock that is upwind of the airfoil's leading edge. The shock is actually a very thin (on the order of the mean free path of the fluid molecules) "layer" across which fluid properties rapidly change [TAP97]. Note how the bow shock is aliased (exhibits jagged contours) due to misalignment of the mesh. In terms of solution discretization, the large element size across the shock means that the represented change in fluid properties will be less pronounced.

To illustrate the effectiveness of some of the proposed methods and the importance of good mesh alignment and sizing, a series of mesh adaptation iterations were run on this example problem. Solution-based orientation and anisotropy information was conditioned (to treat

noise due to numerical approximations and other factors) and used to generate a specialized quadrilateral-dominant (quad-dominant) mesh that is combined with a quad-dominant mesh generated using BubbleMesh [VSI00]. As seen in Figure 1.3a and Figure 1.3b, the enhanced capture of orientation and anisotropy in the new mesh results in improved resolution of the shock profile. Depending on the target application, this can improve the fidelity of the solution downwind as well — this will be revisited in Section 2.3. Of course, improved resolution of the Mach number alone is a somewhat synthetic benchmark. Ultimately, solution outputs such as drag and lift coefficients or other quantities are of interest to an analyst. The key point is that appropriate mesh orientation and anisotropy (nominal size and aspect ratio) are critical for capturing important flow features, assisting solution convergence, improving accuracy in engineering outputs (*e.g.*, lift-to-drag ratio), and mitigating the formation of numerical artifacts. Anisotropy can also lead to more efficient use of computational resources by focusing "attention" on directions in which the solution is changing more rapidly, as opposed to isotropy, where the mesh may be gratuitously fine in all directions.

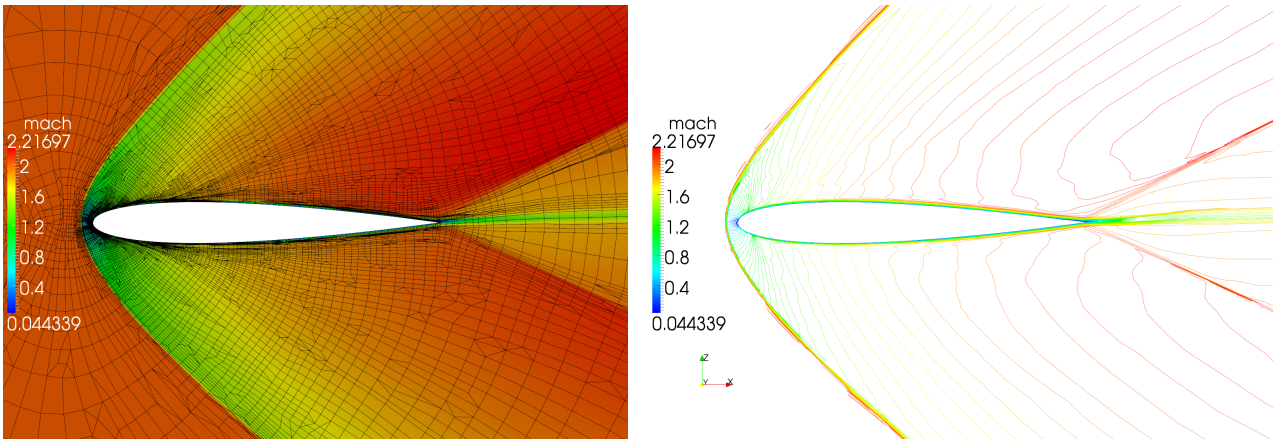


(a) O-type mesh: Mach number plot

(b) O-type mesh: Mach number contour plot

Figure 1.2: Mach plots of a supersonic flow past a 2D airfoil on an O-type mesh

Mesh orientation and anisotropy will be better explained later in the thesis. For now, it should suffice to say that anisotropy indicates preferred directions in which mesh elements are



(a) Adapted mesh using the proposed methods: Mach number plot

(b) Adapted mesh using the proposed methods: Mach number contour plot

Figure 1.3: Mach plots of a supersonic flow past a 2D airfoil on an adapted quad-dominant mesh

stretched and thus implies an orientation for mesh elements as well. Besides these concepts, it is also important to consider other mesh characteristics that go hand-in-hand with mesh generation techniques and that have an effect on the choice of solver and quality of the resulting solution:

- Element type: quadrilateral & hexahedral, simplicial (triangular & tetrahedral), wedges, pyramids, knifes, polyhedral
- Global and local structure: see 2.2.1 for further explanation
- *A priori* (based on element shape alone) and *a posteriori* (based on knowledge of solution information) quality measures

Furthermore, consider that these characteristics may vary over the domain just as orientation and anisotropy do. Variation in mesh orientation and anisotropy can indirectly affect characteristics like local & global structure and quality. However, this work focuses on the former two characteristics.

In this thesis, we recognize that there are other types of problems where control over local characteristics is similarly advantageous. As a concrete example, take pattern placement for

graphic and mechanical design. Here, the goal is to place patterns (which may be as simple as holes) on the boundary surface of a model in an aesthetic and functional manner. One may desire an arrangement of speaker holes that exhibits certain inter-hole alignment and smooth variation in hole diameters over the surface, while enhancing (or at least, not impeding) sound transmission. Whether pattern primitives are holes or something more complex, this can be achieved to some extent solely through control of the variation of orientation and anisotropy of these primitives over the model boundary. Patterns for ceramic tiling, mosaic arrangements, and fabric/upholstery design are other potential applications.

Another example falls in the domain of additive manufacturing (AM). AM is not encumbered by many of the limitations in capability of traditional manufacturing processes. Furthermore, AM cost models usually promote minimization of material but do not penalize geometric complexity as traditional methods might [TG14]. One way to exploit this is by designing truss, webbed, or cellular internal structures (which may be exposed on the "boundary") that provide adequate structural support while minimizing cost, something not feasible with traditional manufacturing processes [ROCV16]. Even if the input material used in an AM process is fixed, it is possible to effect local variation in mechanical response by changing the local structure's characteristics in some manner. Once again, this is a problem that can be addressed through orientation and anisotropy control, provided that such information is taken into account while generating structures.

A related application is the design of medical devices. Hip prosthetics, orthotic insoles, and other devices may benefit from localized control over the support they provide. Again, prosthesis designs may be designed using truss/webbed/cellular structures with variable orientation and anisotropy. A suitable method for generating the geometry can then produce designs tailored to patients' specific needs.

In all these examples, some effort has been made to suggest decoupling the specification of orientation and anisotropy information from the mesh, pattern, or structure generator. In

this way, many applications can benefit from a general set of tools that deal with generating and specifying this information.

1.1 Problem Statement

The presented examples prompt several questions:

- What is a good representation for orientation and anisotropy, and the variation of this information over a domain?
- How can we prescribe (generate) orientation and anisotropy over an entire domain, given constraints from various sources (user input, input geometry, physics) over portions of the domain?
- How can we condition (*e.g.*, de-noise and fair) pre-existing orientation and anisotropy information to make it more suitable for applications?
- What kind of problem-specific strategies are needed to effectively apply these methods in various application domains?

1.2 Proposed Solution

This thesis presents a framework for controlling orientation and anisotropy information over surfaces and volumes as well as recipes for using this framework to solve problems in various application domains: design, analysis, and manufacturing.

A key concept in the proposed solution is to decouple specification of orientation and anisotropy from the algorithms that can use such information to generate meshes, patterns, or geometric structures.

Riemannian metric tensor fields (metric fields) have been selected to provide a convenient abstraction of orientation and anisotropy as they vary over the domain. Metric fields can

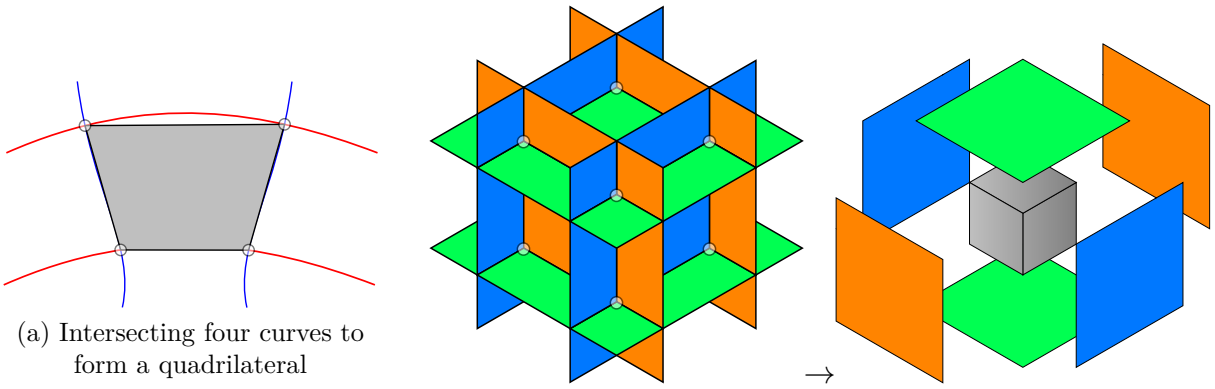
be generated from various source data: the input geometry, user input, and solution data. Their mathematical definition will be elaborated on later in this thesis, but for now it should suffice to indicate that one can go back and forth between a metric tensor and explicit representations of the orientation and anisotropy that it encodes. Furthermore, a rich set of operations can be performed to generate, condition, query, and analyze them. The present work applies many of these operations to prepare suitable fields for target applications.

With these fundamental building blocks, we compose recipes to target specific problems in various application domains. This usually involves:

- Generating a metric field from individual constraints over the domain
- Additional processing of new and existing metric fields to make them more suitable for subsequent steps
- Using the metric field to place features, shape mesh elements, and generate structural primitives in such a way that orientation and anisotropy prescribed by the field is respected

The recipes may be single-shot or involve iteration, as is the case with adaptive remeshing for CFD problems.

Besides metric field processing, the proposed solution also consists of mesh generation algorithms that exploit field "structure" to generate meshes in tandem with existing meshers. These meshing algorithms are based on the premise that elements can be formed from the intersections of a set of curves (in 2D) or surfaces (in 3D). In Figure 1.4, it is shown how non-degenerate configurations of both cases can be used to form elements. Circles are used to mark the points of intersection that are converted to mesh nodes. Then a valid node ordering yields an element. This provides a mechanism for element formation; it does not address the creation of elements with desired orientation and anisotropy. If, for instance, the curves and surfaces are aligned everywhere with the local orientation requirements and



(a) Intersecting four curves to form a quadrilateral

(b) Intersecting six planes to form a hexahedron

Figure 1.4: Intersecting curves and surfaces to form elements

spaced according to the local anisotropy requirements, then the resulting element will also approximately satisfy the requirements. This leaves a few open issues:

1. How to generate curves and surfaces that respect orientation requirements
2. How to space these entities based on anisotropy requirements
3. Adjusting this strategy to generate more than one element, *i.e.*, a complete mesh

The present work addresses the first point by integrating curves and surfaces from the directional vector fields just as streamlines and streamsurfaces are generated from a steady flow field. Different approaches to the second and third points are detailed in Chapter 4 and Chapter 5. One approach is to grow a mesh element-by-element like an advancing front scheme [WK97], but without the need to start at the boundaries. Another is to simultaneously create a complete set of elements, then subject it to anisotropic refinement as needed.

Finally, the proposed methods are flexible enough to be embedded in an anisotropic mesh adaptation process to iteratively improve resulting solutions. The ultimate goal is to utilize metric fields to produce anisotropic quadrilateral- and hexahedral-dominant meshes, thereby leveraging many of the strengths of hexahedral and tetrahedral meshes while circumventing some of their disadvantages.

A graphical depiction of one path, solution-based adaptation of quad-dominant meshes

in 2D, is given in Figure 1.5. Figures 1.5a and 1.5b show a visualization of raw metric data that has been obtained from a computational fluid dynamics (CFD) solver. Conditioning is performed to remove isolated noise and fair global orientation trends. The mesher generates an arrangement of metric streamlines (Figure 1.5c) from which a quad-dominant mesh (Figure 1.5d) is derived and then anisotropically refined. This process is only in certain critical portions of the domain (subdomains), as described in Section 6.2.2. The complementary portions of the domain are treated using BubbleMesh [VSI00] (Figure 1.5e) and the metric field, and the two meshes are combined to produce the result in Figure 1.5f. As necessary, the adaptation framework (Chapter 5) passes this mesh along with boundary conditions and solver parameters to the solver and repeats these steps.

As alluded to earlier, all of this machinery can be applied to other application domains as well. User input and knowledge of geometry can be used to prescribe orientation and anisotropy constraints, from which a metric field covering the whole domain is generated. Meshes produced from this field can be further processed to produce designs and geometries as desired.

1.3 Contributions

The proposed framework for processing metric fields consists of the following contributions:

- Metric field generation
    1. Anisotropic field generation on surfaces and volumes (with limitations)
    2. Boundary-aligned orientation field (pseudo-isotropic) generation for surfaces and volumes
- Employing conditioning (noise removal, fairing, blending, and smoothing) to address issues with existing and generated metric fields

- Novel mesh generation algorithms that exploit metric field structure
- Successful application of these techniques to design, analysis, and manufacturing problems

1.4 Organization of the Thesis

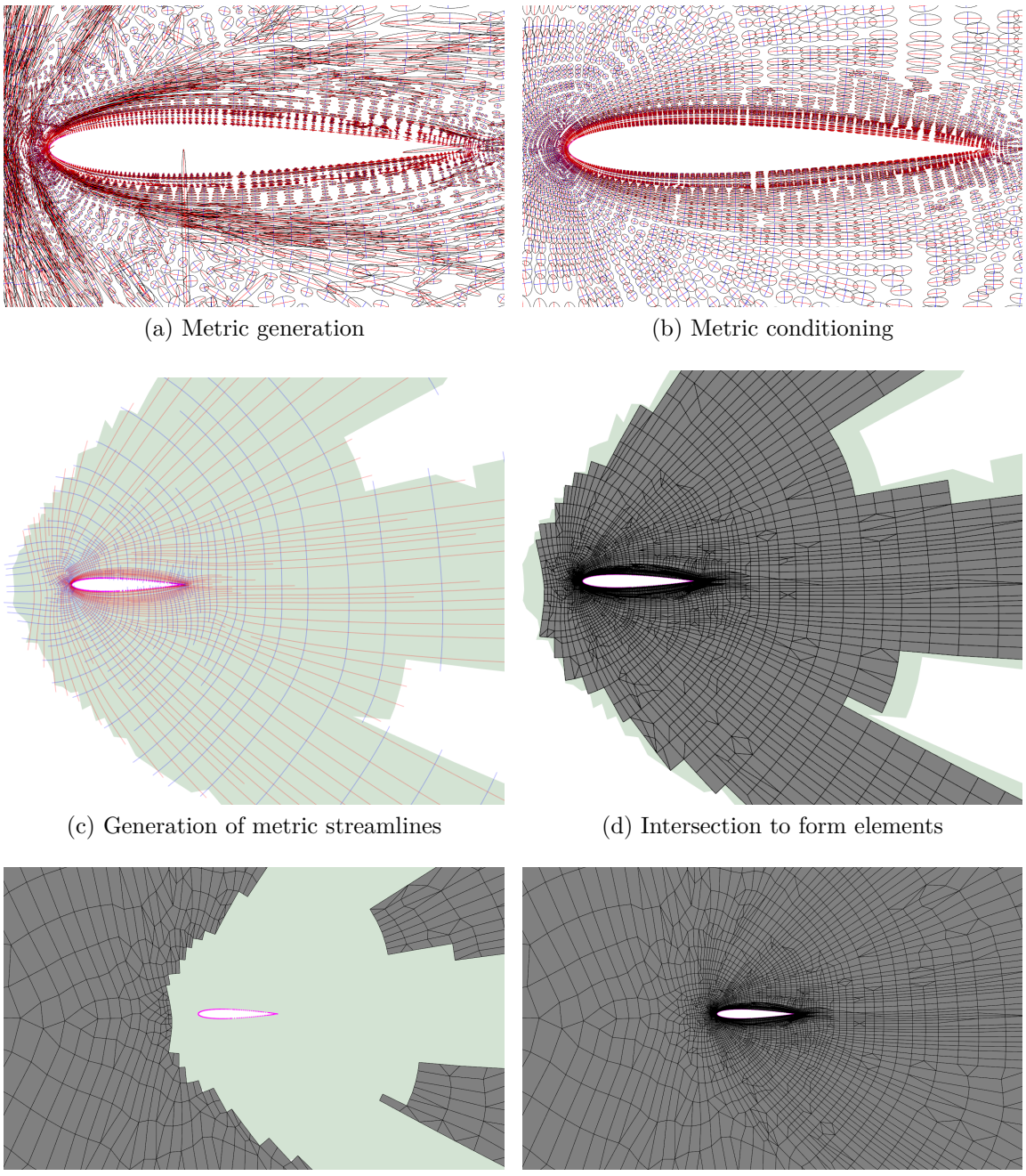
The thesis begins with a review of relevant techniques in the areas of metric field processing, mesh generation, anisotropic meshing generation and adaptation, pattern design, and geometric modeling for additive manufacturing (Chapter 2).

The first part of the thesis work (Chapter 3) is concerned with the representation and manipulation of metric tensor fields. Although primarily in the context of mesh generation, it will become evident that the other considered applications share a lot of this machinery. This includes the problems of constructing a continuous metric field from discrete data (interpolation), generating metric fields based on geometry and user input, extracting structures from metric fields, and visualization of tensor data. Regularization and similar operations on tensor fields are investigated for preparing suitable fields for mesh generation.

The second part of this thesis develops a class of mesh generation schemes that directly exploit the structure of metric fields to generate elements. Initial investigations in two and three dimensions (Sections 4.2, 4.3, and 4.4) yield novel field-guided meshing schemes. This work also lays the foundation for future work on 3D anisotropic mesh generation (Section 7.1).

An adaptation controller that coordinates these components and the processing of metric and other data between iterations is developed in the third part of this work (Chapter 5). Finally, the proposed methods are applied to problems in design, analysis, and manufacturing domains in Chapter 6. This thesis work is then concluded in Chapter 7. Observations drawn from the strengths and weaknesses of the proposed methods and their applications lead to

thoughts on future work in Section 7.1.



(a) Metric generation

(b) Metric conditioning

(c) Generation of metric streamlines

(d) Intersection to form elements

(e) Meshing complementary regions

(f) Complete mesh

Figure 1.5: Anisotropic adaptation of quadrilateral-dominant meshes in 2D

Chapter 2

Background

2.1 Representing Orientation and Anisotropy

There are many ways to represent the orientation of a body with respect to a reference frame in two and three dimensions. Traditional methods include:

- Axis-angle (2D and 3D): all rotations can be equivalently described as a single rotation by an angle θ about an axis $\mathbf{u}$. This can be used to describe the orientation of a body relative to a reference frame.
- Euler angles (3D): a composition of three rotations (represented by a triplet of angles) about reference axes (*extrinsic*) or updated local axes (*intrinsic*) can be used to reach a target orientation. There are many variations on Euler angles, one of which is used to describe aircraft attitude in terms of yaw, roll, and pitch. It is possible to reach configurations where a rotational degree of freedom is lost — this is commonly known as *gimbal lock*.
- Rotation matrices (2D and 3D): a rotation matrix (transformation) can be formed by choosing a set of orthogonal unit vectors (two in 2D and three in 3D) as its columns (in an order that produces a positive determinant). The action of rotation matrices on po-

sition vectors in the reference frame brings them into the rotated frame. Furthermore, rotations can be composed through matrix multiplication.

- Unit quaternions (3D): quaternions are an extension of complex numbers, and unit quaternions in particular represent rotations using only four components that correspond to an axis-angle description of rotation. Quaternions are attractive for numerical and performance reasons when composing rotations. They are also useful for interpolating two rotations/orientations (see spherical-linear interpolation, or "Slerp"). Finally, conversion to rotation matrices or axis-angle allows rotating reference vectors into the orientation described by a unit quaternion.

Descriptions of contemporary methods for representing directions and orientation can be found in the survey paper by Vaxman *et al.* [VCD+16].

Anisotropy is generally defined as the directional dependence of some quantity. For instance, the appearance of a surface at a point may depend on the direction from which it is viewed and the direction of incoming light. Or, a material may respond differently when subjected to axial loading in different directions. Again, a general description of anisotropy may be a function f of direction (a unit vector, angles, *etc.*) and other parameters that evaluates to a response in that direction. Directional dependence suggests that anisotropy may involve orientation; this is explored further in the following section.

Metric Fields

The topic of metric fields is both central to this work and tightly coupled with the proposed techniques. This section provides a basic definition and some background; a more complete treatment in the mesh generation context is given in Chapter 3.

A Riemannian metric tensor $\boldsymbol{\mathcal{M}}$ is a second-order tensor that acts on two vectors to produce a scalar; it defines an inner product. This may be written as $\boldsymbol{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y})$ for two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$. Riemannian metrics encode both orientation (principal directions) and

anisotropy (stretching in these directions). Since inner products are used to measure the lengths of vectors and the angles between vectors, a key point is that these measurements are appropriately biased by the orientation and anisotropy from a metric. Metrics are symmetric operators since the action of $\boldsymbol{\mathcal{M}}$ on two vectors does not depend on their order (just like regular inner products), and positive-definite operators since $\boldsymbol{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}) > 0, \forall \boldsymbol{x} \neq 0$. Thus, they are symmetric positive-definite (SPD). When $\boldsymbol{\mathcal{M}}$ is a function of location in the domain, it is said to be a *metric field*, which is sometimes written as $\boldsymbol{\mathcal{M}}(\boldsymbol{x})$. It will be assumed that $\boldsymbol{\mathcal{M}}$ varies spatially, unless specifically noted otherwise.

Tensors are prevalent in mechanics and many other fields. It is instructive and relevant to consider symmetric (if not also positive-definite) tensors in other domains:

Continuum Mechanics A common example of a symmetric, but not positive-definite, tensor is the Cauchy stress $\boldsymbol{\sigma}$. It is a second-order tensor that acts on a normal vector n to a plane in the deformed configuration of a body. This action produces a traction vector in the deformed configuration. The traction vector can be decomposed into components normal to the plane (normal stresses) and tangent to the plane (shear stresses).

Another example is the right Cauchy-Green deformation tensor $\underline{\underline{\mathbf{C}}}$ [Ogd97]. Its action on an infinitesimal "fiber" in the reference configuration produces the corresponding fiber (that undergoes rotation and stretching) in the deformed configuration. This, in turn, facilitates the calculation of changes in curve lengths and angles after deformation has taken place.

We now draw a useful analogy to the present mesh generation problem. Just as $\underline{\underline{\mathbf{C}}}$ maps infinitesimal vectors from the reference to deformed configuration, $\boldsymbol{\mathcal{M}}$ maps infinitesimal vectors from our physical space to a computational space. This happens in such a way that an isotropic (equilateral) mesh in the computational space corresponds to an appropriately anisotropic mesh (this includes orientation) in the physical space. In fact, this is the unit-mesh [FG08] approach to anisotropic mesh generation: generate meshes such that edge

lengths are as close to unity as possible according to the local metrics. Since the computational space (*i.e.*, deformed configuration) is not present, meshing algorithms can operate in the physical space while relying on metrics to evaluate lengths and angles.

Medical Imaging: Diffusion-Tensor MRI Diffusion-Tensor MRI (DT-MRI) imaging is a technique to measure the anisotropic water diffusion properties of tissue [WH06]. The output is a symmetric, positive-definite tensor field that is called a diffusion tensor field. Water diffusivity is closely tied to tissue microstructure, and it is the case that the principal diffusion directions approximate local tissue fiber directions. In tractography, streamlines are integrated along the direction corresponding to the largest principal diffusion (*i.e.*, the major eigenvector of the diffusion tensor).

Diffusion tensor fields as used in tractography share many characteristics with the metric tensor fields considered in this work:

1. Derived from observations along with a mathematical model
2. Symmetric positive-definite (SPD)
3. May be noisy and need to be conditioned
4. Integral manifolds (*e.g.*, streamlines) are necessary

Furthermore, DT-MRI researchers have worked extensively on problems such as interpolation, noise removal & fairing by image and signal processing, and visualization. In particular, techniques usually reserved for scalar and vector fields have been extended to the general tensor case.

2.2 Mesh Generation

2.2.1 Hexahedral Mesh Generation

Hexahedral (hex) meshes consist of polyhedral elements that have six quadrilateral faces and therefore eight nodes (see Figure 1.4). Note that *higher-order* elements possess curvilinear edges and additional *mid-nodes*. Hex meshes are often required to be *conformal*, in which case internal element faces are used by exactly two elements and exterior element faces are used by exactly one element. For the purposes of this thesis, the following discussion will be further restricted to *boundary conformal* meshes that respect the vertices, curves, and surfaces of the geometry being meshed. Furthermore, *overset* (overlapping) meshes are also excluded. Finally, there is also a notion of *constrained* mesh generation schemes that respect a prescribed surface mesh on portions or the entire boundary.

The use of all-hexahedral meshes is generally preferred over all-tetrahedral meshes in the structural finite element analysis community. This is in part due to experience with various problems, solvers, material models, and element stiffness formulations. Several researchers have attempted to objectively study the differences between the two elements types, while factoring in the behavior of higher-order versions of them [BPM+95]. In fluids problems that may be solved using finite volume techniques, such as high-speed aeroheating, it is beneficial to have orthogonality of control volume boundaries to solution discontinuities such as shocks and other strong features [Gno10]. Properly crafted hexahedral meshes are capable of doing this. The difficulty that will become clear in this section is that high quality, fully-automated hex meshing is an open problem. On top of that, generating hex meshes with spatially varying anisotropy **and** orientation control is intractable in most cases.

Hex meshes can be classified as structured, block-structured, and unstructured; this depends on the node-element connectivity. This is both a consequence of and motivation for the choice of a mesh generation algorithm. Thus, each type is discussed along with

corresponding meshing algorithms in the following sections.

Structured and Block-Structured The term *structured* implies a regular node-element connectivity in the mesh. For hexahedral meshes, this means that the mesh is topologically equivalent to a Cartesian grid, *i.e.*, each node is incident on eight hexes. This makes the storage and access of structured meshes simple and efficient.

Traditional methods for structured mesh generation include: conformal mapping, PDE-based, and algebraic (*e.g.*, transfinite interpolation or TFI) methods [FG08, TWM85]. A geometry that is to be meshed needs to be topologically equivalent to a hexahedron to apply these methods. Otherwise, a special topological mapping is necessary. For example, consider a pentagonal prism. If two of the quadrilateral sides are logically considered to be one face, then a structured mesh can be generated via mapping — this is a TFI method. For principally "blocky" volumes, techniques such as submapping can break up a volume into map-able chunks and enforce interval linear constraints to produce a complete mesh. In other cases, manual decomposition of the geometry into multiple volumes is necessary. To ensure a conformal mesh at the end of the meshing process, interval linear constraints have to be enforced between adjacent volumes. Note that most of these techniques are equally applicable in 2D and 3D.

Block-structured meshes partially circumvent the topological restrictions of fully structured methods. The key difference is that a block structure, or high level decomposition of the volume into logical cubes, is first defined. Each block is then meshed with appropriate constraints between adjacent blocks. This results in a mesh that is structured everywhere except perhaps at edges of the block structure.

Whether structured or block-structured, it is also possible to obtain nicely layered meshes. However, this is subject to obtaining the desired orientation in the mesh first.

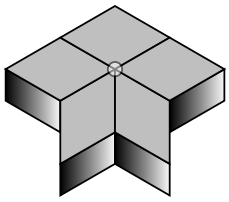


Figure 2.1: Unstructured hex configuration around a node in a mesh

Unstructured Unstructured hex meshes have a non-uniform node-element connectivity throughout the mesh. Figure 2.1 shows an unstructured configuration of hexes around a node in the mesh.

Current practical methods for unstructured hexahedral mesh generation are at most semi-automatic, requiring model decomposition and application of topology specific methods such as sweeping [LGT99]. Other methods, such as the grid based methods of Schneiders et al. [SSW96] or Zhang et al. [ZHB07], are only subject to topology constraints if boundary features (curves and vertices) need to be captured. Although they do provide some degree of anisotropy and size gradation using templates, there is no mechanism to control arbitrary orientation. Furthermore, 3D advancing front methods such as plastering [BM93] start at the boundary while placing elements. Unfortunately, this leads to unmeshable voids and stitching problems between fronts. Unconstrained plastering is a promising extension that aims to avoid these difficulties, but is still under development [SKOB06].

Robust and automatic all-hexahedral mesh generation has not yet been achieved with the guarantees, quality, and control of arbitrary anisotropy/orientation provided tetrahedral and hex-dominant meshing schemes.

2.2.2 Tetrahedral Mesh Generation

Tetrahedral meshes are widely used for fully automatic mesh generation of complex geometries. One reason is that tetrahedra and triangles are simplices. A simplex is an n-dimensional polytope that is the convex hull of $n+1$ vertices. Thus, a triangle is a 2-simplex

and a tetrahedron is a 3-simplex. Unlike polytopes with a greater number of vertices (*e.g.*, a quadrilateral or hexahedron), there is only one possible connectivity of the vertices for a desired orientation; this considers an isolated simplex only.

Another reason is that simplicial meshes can be constructed using a single concept: the Delaunay criterion [FG08]. With the exception of numerical issues in extreme cases and degenerate cases such as co-circular/spherical points, this is another reason that simplicial meshes are attractive. In 2D, Delaunay methods are both robust and guaranteed to maximize the minimum angles of the generated elements. However, the output meshes generally do not conform to the boundaries. Boundary recovery operates through edge flips; technically, this means that the criterion is violated to ensure that boundary features are appropriately captured. The 3D case is similar in spirit but tends to produce ill-shaped tetrahedra. Boundary recovery is also more difficult. So, topological (node connectivity) and geometric (node position) mesh improvement is necessary.

An important point is that Delaunay methods allow us to decouple the problems of deciding node locations and node connectivity. Therefore, most of the control on the mesh can be exerted by carefully planning node locations. One strategy considered in this work is Bubble Packing, in which optimal node locations are determined from a physically-based force relaxation scheme on a set of bubbles [YS00]. The geometry of the bubbles follows from the local orientation and anisotropy requirements, so the final bubble distribution reflects these as well. Nodes are placed at bubble centers, and can be passed through the remaining steps for Delaunay-based methods, such as: node insertion into the mesh, boundary recovery, and quality improvement.

2.2.3 Hexahedral-Dominant Mesh Generation

Hexahedral-dominant meshing schemes have gained traction for fully automatic mesh generation. These schemes generate meshes that consist of hexes and tets in the non-conformal

case, and pyramids and triangular prisms can be added for a conformal result. Rectangular bubble-packing, by Yamakawa and Shimada [YS03a], is an indirect hex-dominant method that is capable of generating high quality hex-dominant meshes. The resulting meshes conform to a specified tensor field and have good hexahedral volume ratios. The method begins by packing rectangular solid cells and then generating nodes at the centers of the cells. Next, a constrained tetrahedral mesh is constructed from the nodes and the original boundary information. This is followed by merging groups of tetrahedra into good hexahedral elements. Optionally, tets can be merged into prisms and possibly pyramids to produce a fully conformal mesh. That is, hanging edges which are present in a mixed hex-tet mesh will be eliminated through the use of prisms and pyramids. Alternatively, special subdivision templates have been developed to produce conformal meshes without pyramids [YS08].

2.3 Anisotropic Mesh Generation and Adaptation

Mesh adaptation is commonly employed in conjunction with CFD and other solvers to improve solution accuracy, convergence, and resolution of various physical phenomena. It proceeds by modifying an initial mesh or completely remeshing the domain according to *a posteriori* meshing requirements computed from solution-based error estimates, which may be represented with Riemannian metric tensors. In particular, we consider gradient-based and adjoint-based methods that rely on Hessian information from a solution variable to determine orientation and aspect ratio, which is then combined with a scalar intensity that is specific to each method [BGPJ06, Par03].

Many existing methods utilize local modification on simplicial meshes to attain metric-conformity during adaptation [Par03]. Simplicial meshes are flexible enough to represent certain spatial variation in anisotropy and orientation. Furthermore, generation and adaptation of simplicial meshes are fairly robust in 2D. However, not all edges or faces can be

aligned with the local metric directions and flow features.

Quadrilateral meshes, on the other hand, can achieve better metric alignment while possessing orthogonality to flow structures. In problems such as high-speed aeroheating, it is important to have orthogonality of control volume boundaries to solution discontinuities such as shocks and other strong features [Gno10]. Quadrilateral meshes are more advantageous in this respect, but are difficult to create and adapt to a metric field. Capturing gradation in anisotropy and orientation is also limited when compared to simplicial meshes.

Chapter 3

Riemannian Metric Tensor Fields

3.1 Overview

Riemannian metrics and metric fields were first introduced in Section 2.1. This chapter begins with the mathematical definition and practical representation of metrics for mesh generation and other applications. More importantly, this is accompanied by several reasons for selecting Riemannian metrics as a practical means of representing anisotropy and orientation information. The discussion then progresses to metric fields and operations that query continuous fields and discrete tensors. This includes interpolation, topological analysis, and visualization. Techniques for generating geometry-based metric fields are then developed and followed by a brief consideration of existing methods for solution-based metric computation. Finally, conditioning operations on metric fields are presented.

3.2 Riemannian Metrics

Riemannian metric tensors are a popular means of anisotropic mesh control [YS00, SLI98, BH96]. Conceptually, a Riemannian metric $\mathcal{M}$ defines an inner product between two vectors

$\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^n$. That is, $\boldsymbol{x}\cdot\boldsymbol{y}$ is replaced by the action of $\boldsymbol{\mathcal{M}}$ on $\boldsymbol{x}$ and $\boldsymbol{y}$: $\boldsymbol{\mathcal{M}}(\boldsymbol{x},\boldsymbol{y})$ (expressions for this action will be revealed later on). Since inner products can be used to measure the Euclidean length of a vector as well as the angle between two vectors, the idea is that $\boldsymbol{\mathcal{M}}$ somehow affects these measurements based on the anisotropy and orientation it represents. To better understand how that happens, let's begin with an exploration of Riemannian metrics in the context of differential geometry, computer graphics, and continuum mechanics. Consider a surface S immersed in $\mathbb{R}^n$ that is mapped to another surface $\tilde{S}$ immersed in $\mathbb{R}^m$ by $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Three common scenarios may be described this way:

Parametric surfaces (differential geometry) Here S is a patch in a two-dimensional ($n = 2$) parameter space that is mapped to a parametric surface patch $\tilde{S}$ immersed in $\mathbb{R}^3$ ($m = 3$), as shown in Figure 3.1.

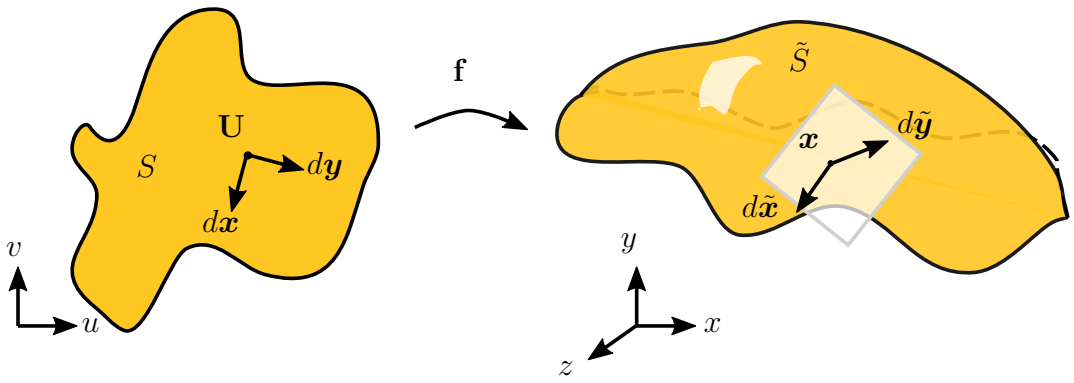


Figure 3.1: Parametric surface

The coordinates of a point $\boldsymbol{x}$ on $\tilde{S}$ may be expressed as:

$$\boldsymbol{x} = \mathbf{f}(\mathbf{U}) = \mathbf{f}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix},$$

where $\mathbf{U}$ is a point in the parameter space with components u and v and $\boldsymbol{x}$ has components x, y, and z. As seen in the figure, two differential directions $d\boldsymbol{x}$ and $d\boldsymbol{y}$ in the parameter

space are mapped to corresponding directions in the tangent space of $\tilde{S}$ by the Jacobian $\mathbf{J}$ of the mapping $\mathbf{f}$ as follows:

$$\begin{aligned} d\tilde{\boldsymbol{x}} &= \frac{d\mathbf{f}}{d\mathbf{U}} d\boldsymbol{x} \\ &= \mathbf{J} d\boldsymbol{x}, \text{ and similarly,} \\ d\tilde{\boldsymbol{y}} &= \mathbf{J} d\boldsymbol{y}. \end{aligned}$$

Thus, the Jacobian is responsible for any changes in direction and length to $d\boldsymbol{x}$ and $d\boldsymbol{y}$. How does the inner product of $d\tilde{\boldsymbol{x}}$ and $d\tilde{\boldsymbol{y}}$ relate to their preimages in the parameter space? Well:

$$\begin{aligned} d\tilde{\boldsymbol{x}} \cdot d\tilde{\boldsymbol{y}} &= (\mathbf{J} d\boldsymbol{x}) \cdot (\mathbf{J} d\boldsymbol{y}) \\ &= (\mathbf{J} d\boldsymbol{x})^{\mathrm{T}} (\mathbf{J} d\boldsymbol{y}) \\ &= d\boldsymbol{x}^{\mathrm{T}} \left(\mathbf{J}^{\mathrm{T}} \mathbf{J} \right) d\boldsymbol{y} \\ &= d\boldsymbol{x}^{\mathrm{T}} \, g \, d\boldsymbol{y} \\ &= d\boldsymbol{x} \cdot g \cdot d\boldsymbol{y}. \end{aligned}$$

g is the referred to as the implied metric of the mapping $\mathbf{f}$ and is also known as the *first fundamental form*. Since g tells us how to compute the inner products of $d\boldsymbol{x}$ and $d\boldsymbol{y}$ once mapped to $\tilde{S}$, it also allows us to compute the angle $\tilde{\theta}$ between $d\tilde{\boldsymbol{x}}$ and $d\tilde{\boldsymbol{y}}$ as well as the length of $d\tilde{\boldsymbol{x}}$ in terms of $d\boldsymbol{x}$ and $d\boldsymbol{y}$. When g is positive-definite — that is, when $d\boldsymbol{x} \cdot g \cdot d\boldsymbol{x} > 0$, $\forall d\boldsymbol{x} \neq 0$ — it is a Riemannian metric and in this work denoted by $\boldsymbol{\mathcal{M}}$.

$$\begin{aligned} \|d\tilde{\boldsymbol{x}}\| = \|d\boldsymbol{x}\|_{\boldsymbol{\mathcal{M}}} &= \left(d\boldsymbol{x}^{\mathrm{T}} \, g \, d\boldsymbol{x} \right)^{1/2}, \quad \text{and} \\ \cos\tilde{\theta} &= \frac{d\boldsymbol{x}^{\mathrm{T}} \, g \, d\boldsymbol{y}}{\|d\boldsymbol{x}\|_{\boldsymbol{\mathcal{M}}} \, \|d\boldsymbol{y}\|_{\boldsymbol{\mathcal{M}}}}, \end{aligned}$$

where the L^2 (Euclidean) norm $\|d\tilde{\boldsymbol{x}}\|$ is given by $(d\tilde{\boldsymbol{x}} \cdot d\tilde{\boldsymbol{x}})^{1/2}$ and the norm under the metric

$\boldsymbol{\mathcal{M}}$ is denoted by $\|d\boldsymbol{x}\|_{\boldsymbol{\mathcal{M}}}$.

Alternatively, in the language of exterior calculus, we may instead refer to the Jacobian as the push-forward df. df is the exterior derivative of the vector-valued 0-form f, and it acts on a single vector such as $\boldsymbol{x}$ or $\boldsymbol{y}$ (rather than differential directions as in the traditional vector calculus treatment). Just as before, df maps parameter-space directions to directions in the tangent space of the parametric surface. The metric is defined in terms of the action of df on $\boldsymbol{x}$ and $\boldsymbol{y}$: $g(\boldsymbol{x}, \boldsymbol{y}) = df(\boldsymbol{x}) \cdot df(\boldsymbol{y})$.

Surface parameterization (computer graphics) Here S is a parametric surface patch embedded in $\mathbb{R}^3$ ($n = 3$) for which a parameterization is found; in practice, an *atlas* of *charts* act as a mapping from S to one or more $\tilde{S}$ patches in the parameter space(s) (where $m = 2$). Each chart may map a portion of S to its own $\tilde{S}$, and charts may overlap in their coverage of S (in which regions there are additional mappings between charts). Altogether, this constitutes the atlas [SW13]. This is illustrated in Figure 3.2.

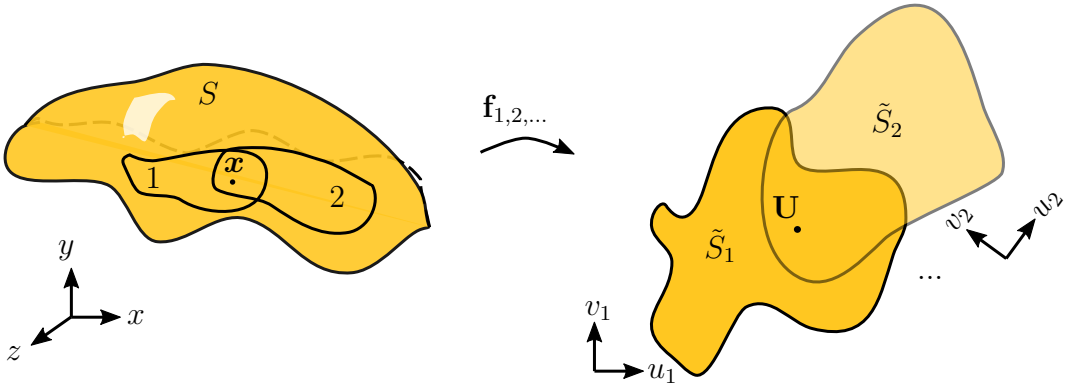


Figure 3.2: Parameterization of a surface S via an atlas of charts

Continuum mechanics Here S corresponds to an undeformed configuration (geometry) that undergoes a deformation to reach $\tilde{S}$. Through this deformation, a *material* point X in S is mapped by ϕ (previously $\mathbf{f}$) to reach a deformed position $\boldsymbol{x}$ in $\tilde{S}$, most commonly in 3D

($n = m = 3$). Sometimes this is expressed as $\boldsymbol{x} = \phi(\mathbf{X}, t)$ or $\boldsymbol{x}(\mathbf{X}, t)$. For the remainder of the discussion, time-dependence is dropped.

Aside from choice of variable names and notation, this is very similar to the parametric surface case, except that the preimage and image of ϕ are both in $\mathbb{R}^3$. Here, the Jacobian $\mathbf{J}$ from before is known as the deformation gradient tensor $\mathbf{F}$, and the metric is the right Cauchy-Green deformation tensor $\underline{\underline{\mathbf{C}}}$.

Metrics have several key properties, some of which can be gathered from the exploration above:

- They are symmetric: $\boldsymbol{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y}) \equiv \boldsymbol{\mathcal{M}}(\boldsymbol{y}, \boldsymbol{x})$.
- They are positive-definite: $\boldsymbol{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}) > 0, \forall \boldsymbol{x} \neq 0$.
- They are bilinear: $\boldsymbol{\mathcal{M}}(\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2, \boldsymbol{y}) = \alpha_1 \boldsymbol{\mathcal{M}}(\boldsymbol{x}_1, \boldsymbol{y}) + \alpha_2 \boldsymbol{\mathcal{M}}(\boldsymbol{x}_2, \boldsymbol{y})$, and similar for the second argument.

We now arrive at the practical mathematical representation for metrics: they are second-order symmetric positive-definite tensors. In the standard basis (Cartesian tensors) we have $\boldsymbol{\mathcal{M}} = \mathcal{M}_{ij}\, \mathbf{e}_i \otimes \mathbf{e}_j$, where $\mathcal{M}_{ij}$ are the components of $\boldsymbol{\mathcal{M}}$ in the basis defined by vectors $\mathbf{e}_{1,2,3}$. Given a metric tensor $\boldsymbol{\mathcal{M}}$, the dot (inner) product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y} \in \mathbb{R}^3$ can be written as $\boldsymbol{x} \cdot \boldsymbol{\mathcal{M}} \cdot \boldsymbol{y} = \mathcal{M}_{ij} x_i y_j$ [LTU99]. This can be used to obtain the L_2 norm of a vector under the metric via $(\boldsymbol{x} \cdot \boldsymbol{\mathcal{M}} \cdot \boldsymbol{x})^{1/2}$, as well as the angle θ between two vectors:

$$\theta = \arccos \frac{\boldsymbol{x} \cdot \boldsymbol{\mathcal{M}} \cdot \boldsymbol{y}}{(\boldsymbol{x} \cdot \boldsymbol{\mathcal{M}} \cdot \boldsymbol{x})^{1/2} (\boldsymbol{y} \cdot \boldsymbol{\mathcal{M}} \cdot \boldsymbol{y})^{1/2}} .$$

Suppose we have a curve $\boldsymbol{r}(t)$ in the domain being meshed, and a spatial metric field $\boldsymbol{\mathcal{M}}(\boldsymbol{x})$. Then, the metric length l of $\boldsymbol{r}(t)$ over an interval $t \in [t_0, t_1]$ can be expressed as:

$$l = \int_{t_0}^{t_1} \left[\mathcal{M}_{ij} \frac{dr_i}{dt} \frac{dr_j}{dt} \right]^{1/2} dt = \int_{t_0}^{t_1} \left[\dot{\boldsymbol{r}}(t) \cdot \boldsymbol{\mathcal{M}}\left(\boldsymbol{r}(t)\right) \cdot \dot{\boldsymbol{r}}(t) \right]^{1/2} dt \,, \tag{3.1}$$

where $i, j = 1, 2, 3$. Note that all of these formulas yield the standard Euclidean length and angle formulas when $\boldsymbol{\mathcal{M}}(\boldsymbol{x}) = \mathbf{I}$.

In practice, the Riemannian metrics are stored as symmetric positive-definite (SPD) 3×3 matrices. Orientation and anisotropy information are explicitly obtained from the eigen-decomposition ([Str88]) by the Jacobi rotation method (also see Section 3.5). Equation (3.2) is used to go back to the SPD matrix form.

$$\boldsymbol{\mathcal{M}} = \mathbf{Q} \cdot \boldsymbol{\Lambda} \cdot \mathbf{Q}^{\mathrm{T}}, \quad \mathbf{Q} = \begin{pmatrix} | & | & | \\ \boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 \\ | & | & | \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} h_1^{-2} & & \\ & h_2^{-2} & \\ & & h_3^{-2} \end{pmatrix}, \tag{3.2}$$

where h_1, h_2, and h_3 are the locally desired sizes in the orthonormal directions $\boldsymbol{q}_1$, $\boldsymbol{q}_2$, and $\boldsymbol{q}_3$. It is sometimes practical to maintain the decomposed form (vectors and sizes) as well because some operations explicitly require it. Also note that the components of $\boldsymbol{\mathcal{M}}$ vary with the inverse square of the desired sizes. An alternate, "linear-size" representation given by $\boldsymbol{\mathcal{M}}^{-1/2}$ has components that vary with h_i.

3.3 Common Operations with Metrics

During meshing it is often necessary to query eigenvector directions, find the best eigenvector corresponding to an input direction, and to relate the eigenvectors of two closely aligned metrics. In this work, eigenvectors are specified by an index ($e = 1, 2, 3$) and sign ($\sigma = \pm 1$) with respect to a given metric tensor. The eigenvector (index-sign pair with respect to a specific metric) that best corresponds to an input direction is taken to be the one that has the largest dot product with it. For example, if the input direction is the normal of a patch of mesh faces, this facilitates finding the best eigenvector to trace streamlines in the normal direction.

When relating two metrics, one of two methods is used. The first consistently orders eigenvectors based on eigenvalue magnitudes, so correspondence between the eigenvectors of both metrics is determined by eigenvalue magnitude. This is used to define the three eigenvector fields which are traced during tiling. The second is employed during field generation to snap the eigenvectors of one tensor to the eigenvectors of another. It compares directions as mentioned before to determine the correspondence.

3.4 Representation of Metric Fields

3.4.1 Mesh-supported Fields

Techniques like regression or interpolation are useful when constructing a continuous field from discrete data samples. The metric fields in this work have a piecewise-continuous representation obtained by interpolating on an unstructured mesh. The support is a convex, constrained Delaunay background tet mesh. Metrics tensors are stored at the nodes of the support. An advantage of this choice is that boundaries of the input geometry are explicitly captured and that the support can easily be adjusted to add or remove data (refinement/coarsening). With known surface normals, a flood fill is used to mark interior and exterior tets. The first step to field evaluation at a position $\boldsymbol{x}$ is to locate the element in the background mesh that encloses $\boldsymbol{x}$ — this is referred to as point location. Point location in the background mesh is performed by walking [SS03]. The convexity requirement prevents walks from falsely terminating at non-convex boundaries. This is also somewhat optimal for the access patterns of the proposed meshers. These meshers perform groups of queries on positions that are relatively close to one another rather than jumping around the domain. Another benefit is that the barycentric coordinates used during the walk can be reused during interpolation.

3.4.2 Interpolation Schemes

Here, fields are continuously interpolated over tetrahedral elements using linear or log-Euclidean (geometric) interpolation schemes. Given a location identified by barycentric coordinates $(\alpha_1, \alpha_2, \alpha_3)$ and nodal tensors $\{\boldsymbol{\mathcal{M}}_1, \boldsymbol{\mathcal{M}}_2, \boldsymbol{\mathcal{M}}_3\}$ of the enclosing triangle, several schemes for the interpolated metric $\boldsymbol{\mathcal{M}}$ are given as:

$$\boldsymbol{\mathcal{M}} = \sum_{i=1}^{d+1} \alpha_i \boldsymbol{\mathcal{M}}_i, \qquad \text{Linear}$$

$$\boldsymbol{\mathcal{M}} = \prod_{i=1}^{d+1} (\boldsymbol{\mathcal{M}}_i)^{\alpha_i} = \exp\left[\sum_{i=1}^{d+1} \alpha_i \ln\left[\boldsymbol{\mathcal{M}}_i\right]\right] = \exp\left[\sum_{i=1}^{d+1} \alpha_i \, \mathbf{Q}_i \cdot \ln\left[\boldsymbol{\Lambda}_i\right] \cdot \mathbf{Q}_i^{\mathrm{T}}\right], \text{ or} \qquad \text{Log-Euclidean}$$

$$\boldsymbol{\mathcal{M}} = \left[\sum_{i=1}^{d+1} \alpha_i \boldsymbol{\mathcal{M}}_i^{-1/2}\right]^{-2} = \left[\sum_{i=1}^{d+1} \alpha_i \, \mathbf{Q}_i \cdot \boldsymbol{\Lambda}_i^{-1/2} \cdot \mathbf{Q}_i^{\mathrm{T}}\right]^{-2}. \qquad \text{Linear-size}$$

In general, these barycentric or linear (in position) interpolation schemes can be visualized

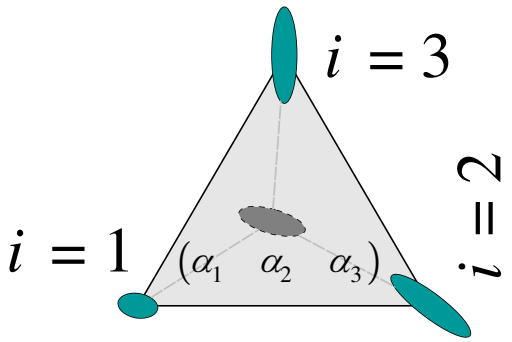


Figure 3.3: Metric interpolation over a simplex

as shown in Figure 3.3. There, a metric ellipse (covered in the next section) is shown at each vertex $(i = 1, 2, 3)$ of a triangle. Then the interpolated metric is shown at the desired position with barycentric coordinates $(\alpha_1, \alpha_2, \alpha_3)$. The linear scheme, while computationally fast, exhibits undesired phenomena such as swelling ([PFA06]) and non-linear size/anisotropy gradients when applied to nodal metrics with varying orientation and anisotropy, respectively. These issues are addressed to some extent by the log-Euclidean scheme [AFPA06]. Finally,

the linear-size scheme is equivalent to performing linear interpolation on the linear-size representation from the previous section, then transforming back to the standard form.

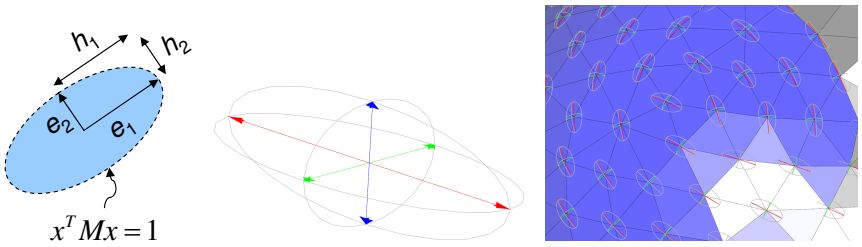
3.5 Methods for Metric Field Visualization

There are several ways to visualize planar and volumetric SPD tensor fields. One of the most elementary methods is that of tensor ellipsoids. This concept draws on the use of metrics to evaluate distance. In Euclidean space we can define a unit circle or sphere (*unit-ball* in general) as the locus of positions $\boldsymbol{x}$ that satisfy $\boldsymbol{x} \cdot \boldsymbol{x} = 1$. That is, all $\boldsymbol{x}$ that are of unit length. When given a metric $\boldsymbol{\mathcal{M}}$ at the origin, we can write $\boldsymbol{x} \cdot \boldsymbol{\mathcal{M}} \cdot \boldsymbol{x} = 1$ to visualize the unit-ball according to the metric. Note that the square root has been simplified out in both cases. This directly reflects the orientation and anisotropy encoded in the metric in the form of an ellipsoid. In fact, it can be shown that:

1. The major, medium, and minor axes of the ellipsoid directly correspond to the eigenvectors of $\boldsymbol{\mathcal{M}}$
2. The major, medium, and minor radii of the ellipsoid correspond to the sizes encoded by $\boldsymbol{\mathcal{M}}$ (where size $h_i = \lambda_i^{-1/2}$)

Tensor ellipsoids are an example of visualization using glyphs. They can be drawn as wireframes that are supplemented by arrows indicating the principle directions (hedgehogs), or as solid/transparent surfaces. Alternatively, one can draw rectangular solid cells (hexahedra) that use the local metric eigenvectors as a basis and the locally desired sizes as their edge lengths or half edge lengths. Examples of all of these are shown in Figure 3.4, along with superquadrics that incorporate more information about the relative anisotropy [Kin04].

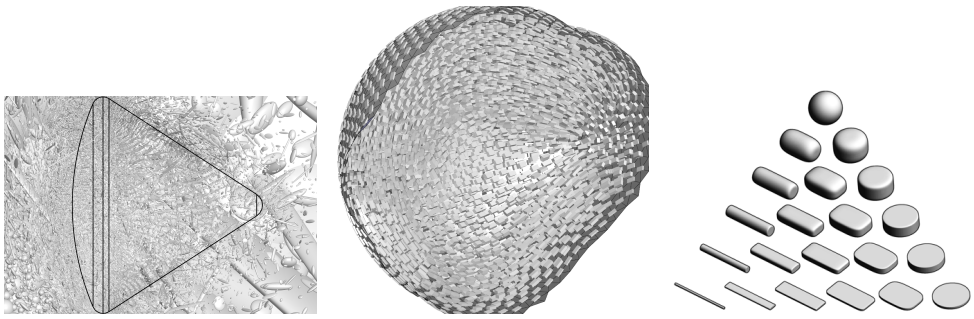
Glyphs depict the orientation and anisotropy of the field at sample points only. For an understanding of global trends in orientation (and perhaps anisotropy), we resort to integral



(a) Connection to unit-ball

(b) Hedgehogs and wireframe ellipsoids

(c) Surface field with wireframe ellipsoids



(d) Shaded ellipsoids from a volumetric dataset

(e) Hexahedral glyphs from another volumetric dataset

(f) Superquadrics (reproduced from [Kin04])

Figure 3.4: Glyph representations of metric tensor fields on surfaces and volumes

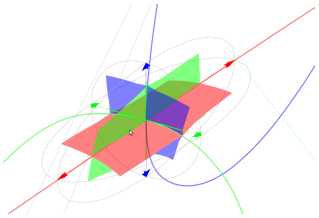


Figure 3.5: Streamlines and streamsurfaces of a metric field

manifolds. That is, curves and surfaces that are integrated from the eigenvector-fields of the metric field in consideration. A collection of integral manifolds starting from a single point is shown in Figure 3.5. Discussion of surfaces is deferred to Section 4.3, where they are employed for mesh generation. Curves, however, are introduced in the following.

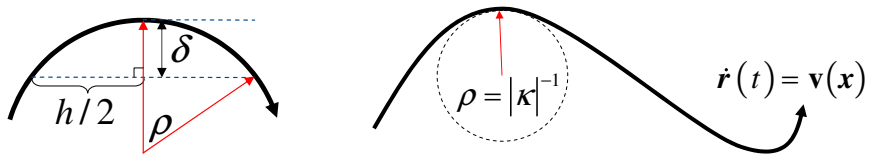
Metric streamlines For a steady flow field $\boldsymbol{v}(\boldsymbol{x})$ in $\mathbb{R}^2$ or $\mathbb{R}^3$, a streamline is a curve in the domain that is everywhere tangent to $\boldsymbol{v}(\boldsymbol{x})$. Mathematically, the equation for a streamline

$\boldsymbol{r}(t)$ is the following ordinary differential equation:

$$\dot{\boldsymbol{r}}(t) = \boldsymbol{v}(\boldsymbol{x})\,, \tag{3.3}$$

where t is a pseudo-time and $\dot{\boldsymbol{r}}(t)$ is the tangent to the streamline at time t. A starting position $\boldsymbol{x}_0$ is provided as the initial condition.

In our case, there is a continuous metric field $\boldsymbol{\mathcal{M}}(\boldsymbol{x})$ rather than vector fields. Say that the operations of evaluating $\boldsymbol{\mathcal{M}}(\boldsymbol{x})$ and then performing an eigen-decomposition are composed. That indirectly provides the eigenvector-fields $\boldsymbol{q}_{1,2,3}(\boldsymbol{x})$. Schemes to integrate discrete streamlines of eigenvector-fields have previously been detailed; refer to [ACSD+03, TGDC05] for details. This section describes a new scheme for adaptive step-sizing during integration of vector fields, and more specifically, eigenvector-fields. The problem is that while commonly used Runge-Kutta (RK) integrators (*e.g.*, fourth-order) exhibit good global error behavior so new points lie very close to the "true streamline," segments connecting points may deviate more significantly. The proposed scheme adjusts the step size to meet a chordal deviation criterion for the streamline segments rather than by an error estimate. If the true streamline geometry were known *a priori*, then the curve could be directly tessellated while controlling deviation.



(a) Chordal deviation for a constant-curvature arc

(b) Osculating circle for an integrated metric-streamline

Figure 3.6: Concepts of deviation-based step size control

A simple model for chordal deviation control, as illustrated in Figure 3.6 and described in [ACSD+03], considers a curve with locally constant curvature (*i.e.*, a circular arc) and

provides the maximum chord length that satisfies the allowable deviation. If the local curve tangents are unit-length, the maximum chord length corresponds to the integration step size. In that case, the step size h is given by

$$h \leq 2\left[\delta_{\text{allow}}\left(2\rho - \delta_{\text{allow}}\right)\right]^{1/2} , \tag{3.4}$$

where δ_{allow} is the allowable chordal deviation and the local radius of curvature is ρ. Since this is scale-dependent, a more appropriate (non-dimensional) measure of allowable deviation relative to radius of curvature, $\delta^{\star}_{\text{allow}} \in (0, 1]$, is defined such that $\delta_{\text{allow}} = \delta^{\star}_{\text{allow}}\rho$. Then Equation (3.4) becomes

$$h \leq 2\rho\left[\delta^{\star}_{\text{allow}}\left(2 - \delta^{\star}_{\text{allow}}\right)\right]^{1/2} ,$$

from which it is clear that $\max h \propto \rho$, and with the exception of ρ the right-hand side is constant and can be precomputed.

The next task is to quantify ρ without *a priori* knowledge of the fully integrated curve. For this purpose, consider again the definition of a streamline given in Equation (3.3). In the rest of this discussion, $\boldsymbol{v}$ will be taken to mean one of the eigenvector fields $\boldsymbol{q}_{1,2,3}$.

The magnitude of curvature $|\kappa|$ of a parametric curve $\boldsymbol{r}\left(t\right)$ is given by $\left\|\dot{\boldsymbol{r}} \times \ddot{\boldsymbol{r}}\right\|_2 \cdot \left\|\dot{\boldsymbol{r}}\right\|_2^{-3}$. Recognizing that $\ddot{\boldsymbol{r}}\left(t\right)$ is the material derivative of the spatial field $\boldsymbol{v}\left(\boldsymbol{x}\right)$ and substituting Equation (3.3) in yields

$$|\kappa| = \left\|\boldsymbol{v} \times \frac{\mathcal{D}\boldsymbol{v}}{\mathcal{D}t}\right\|_2 \cdot \left\|\boldsymbol{v}\right\|_2^{-3} , \tag{3.5}$$

from which the radius of curvature ρ is computed as $|\kappa|^{-1}$. In Equation (3.5), the material derivative $\frac{\mathcal{D}\boldsymbol{v}}{\mathcal{D}t}$ simplifies to $\cancelto{0}{\frac{\partial \boldsymbol{v}}{\partial t}} + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \boldsymbol{v}$, which is simply the convective acceleration of a particle advected along $\boldsymbol{v}(\boldsymbol{x})$.

The final piece is to obtain the spatial gradient of $\boldsymbol{v}\left(\boldsymbol{x}\right)$. Since the discussion up to this point considers $\boldsymbol{v}\left(\boldsymbol{x}\right)$ to be a vector field, the application to integration of metric fields poses

two challenges:

1. Eigenvector fields are not vector fields because they do not possess magnitude or direction, only orientation. Furthermore, they do not vanish but are not uniquely defined at umbilics.
2. $\boldsymbol{v}$ at a point $\boldsymbol{x}$ is computed from the eigensystem of the metric $\boldsymbol{\mathcal{M}}$ evaluated at $\boldsymbol{x}$ and does not have a straightforward representation.

The first point is addressed by normalizing eigenvectors and flipping them for directional consistency, *e.g.*, when integrating metric-streamlines and when sub-sampling for RK4. Umbilics occur in 2D wherever the field is locally isotropic. Thus all vectors in $\mathbb{E}^2$ are eigenvectors there as well (no unique solution). This is safely circumvented by simply returning the minimum allowable step size if the local tensor is isotropic but the nodal tensors of the support triangle are not; isotropic or nearly so regions should be specially treated since the numerical decomposition will return flips and 90° rotations of the standard basis that are inconsistent from point to point.

The second point is due to the fact that the local metric is computed and expressed in component form and then decomposed to get $\boldsymbol{v}(\boldsymbol{x})$. A simple solution is to employ a central difference approximation to compute spatial derivatives of the eigenvector-fields. Another approach we have formulated for 2D is to decouple the dependence on the derivatives of $\boldsymbol{v}(\boldsymbol{x})$ from the spatial derivatives of the metric field. The remainder of this section presents an analytical derivation of $\nabla_{\boldsymbol{x}} \boldsymbol{v}(\boldsymbol{x})$ that conveniently generalizes to any fully 2D metric interpolation scheme in component space.

Metric Field Derivatives in 2D Given a 2D metric field $\boldsymbol{\mathcal{M}}(\boldsymbol{x})$, the goal is to compute the spatial gradients of the eigenvectors $\boldsymbol{v}_i$ and corresponding eigenvalues λ_i of $\boldsymbol{\mathcal{M}}$. The present work considers various forms of piecewise-continuously interpolated fields defined over a triangular background mesh, as presented in Section 3.4.

The interpolated field over an element is a function of the nodal (and midnode) positions $\{\mathbf{p}_e\}$ and nodal metrics $\left\{\boldsymbol{\mathcal{M}}^{(e)}\right\}$, and is evaluated at a value of $\boldsymbol{x}$ contained by the element. Thus the spatial gradients of λ_i and $\boldsymbol{v}_i$ are given by the chain-rule as (i subscript dropped for clarity):

$$\nabla_{\boldsymbol{x}}\,\lambda = \frac{\partial\lambda}{\partial\boldsymbol{x}_l}\mathbf{e}_l \qquad = \frac{\partial\lambda}{\partial\boldsymbol{\mathcal{M}}_{jk}}\frac{\partial\boldsymbol{\mathcal{M}}_{jk}}{\partial\boldsymbol{x}_l}\mathbf{e}_l \qquad = \nabla^{\mathrm{T}}_{\boldsymbol{\mathcal{M}}}\lambda :\ \nabla^{\mathrm{T}}_{\boldsymbol{x}}\boldsymbol{\mathcal{M}} \tag{3.6}$$

$$\nabla_{\boldsymbol{x}}\,(\boldsymbol{v})_m = \frac{\partial\,(\boldsymbol{v})_m}{\partial\boldsymbol{x}_l}\,\mathbf{e}_m\otimes\mathbf{e}_l \qquad = \frac{\partial\,(\boldsymbol{v})_m}{\partial\boldsymbol{\mathcal{M}}_{jk}}\frac{\partial\boldsymbol{\mathcal{M}}_{jk}}{\partial\boldsymbol{x}_l}\,\mathbf{e}_m\otimes\mathbf{e}_l \qquad = \nabla^{\mathrm{T}}_{\boldsymbol{\mathcal{M}}}\,(\boldsymbol{v})_m : \nabla^{\mathrm{T}}_{\boldsymbol{x}}\boldsymbol{\mathcal{M}}\,. \tag{3.7}$$

Closed-form expressions for $\lambda_i\,(\boldsymbol{\mathcal{M}})$ and $\boldsymbol{v}_i\,(\boldsymbol{\mathcal{M}})$ are available in the 2D case by studying the characteristic polynomial of $\boldsymbol{\mathcal{M}}$ and the kernels of $\boldsymbol{\mathcal{M}}-\lambda_i\mathbf{I}$. The forms used in elementary stress analysis to obtain the principal stresses and directions of the Cauchy stress tensor are particularly suitable. They can be stated as:

$$\lambda_{\{1,2\}} = \frac{\boldsymbol{\mathcal{M}}_{11}+\boldsymbol{\mathcal{M}}_{22}}{2} \pm \tau_{\max}, \quad \tau_{\max} = \left[\left(\frac{\boldsymbol{\mathcal{M}}_{11}-\boldsymbol{\mathcal{M}}_{22}}{2}\right)^2 + \boldsymbol{\mathcal{M}}_{12}^2\right]^{1/2} \tag{3.8}$$

$$\boldsymbol{v}_{\{1,2\}} = \begin{pmatrix}\cos\theta_{\boldsymbol{v}} + \{0,\pi/2\}\\ \sin\theta_{\boldsymbol{v}} + \{0,\pi/2\}\end{pmatrix}, \quad \tan 2\theta_{\boldsymbol{v}} = \frac{2\boldsymbol{\mathcal{M}}_{12}}{\boldsymbol{\mathcal{M}}_{11}-\boldsymbol{\mathcal{M}}_{22}}\,, \tag{3.9}$$

where it is assumed in the following that the direction of $\boldsymbol{v}_{\{1,2\}}\,(\boldsymbol{\mathcal{M}}\,(\boldsymbol{x}))$ is consistent over a neighborhood ϵ around $\boldsymbol{x}$.

The previous chain-rule decompositions of $\nabla_{\boldsymbol{x}}\,\lambda_i$ and $\nabla_{\boldsymbol{x}}\,\boldsymbol{v}_i$ are convenient because the derivatives of the quantities with respect to $\boldsymbol{\mathcal{M}}$ are decoupled from the interpolation-dependent gradient $\nabla_{\boldsymbol{x}}\,\boldsymbol{\mathcal{M}}$. Therefore, $\nabla_{\boldsymbol{\mathcal{M}}}\,\lambda_i$ and $\nabla_{\boldsymbol{\mathcal{M}}}\,\boldsymbol{v}_i$ can be derived once for all cases and then combined with $\nabla_{\boldsymbol{x}}\,\boldsymbol{\mathcal{M}}$, as determined for a specific choice of $\boldsymbol{\mathcal{M}}\,(\boldsymbol{x})$. The general

derivatives of $\lambda_{\{1,2\}}$ and $\boldsymbol{v}_{\{1,2\}}$ are obtained first:

$$\nabla^{\mathrm{T}}_{\boldsymbol{\mathcal{M}}} \lambda_{\{1,2\}} = \frac{1}{2}\mathbf{I} \pm \tau^{-1}_{\max} \left[\frac{\boldsymbol{\mathcal{M}}_{11} - \boldsymbol{\mathcal{M}}_{22}}{4} \begin{pmatrix} 1 & \\ & -1 \end{pmatrix} + \boldsymbol{\mathcal{M}}_{12} \begin{pmatrix} & 1 \\ 1 & \end{pmatrix} \right]. \tag{3.10}$$

Similarly, we have:

$$\nabla^{\mathrm{T}}_{\boldsymbol{\mathcal{M}}} \mathrm{v}_{\{1,2\}} = \pm \frac{\mathrm{v}_{\{2,1\}}}{\left(\boldsymbol{\mathcal{M}}_{11} - \boldsymbol{\mathcal{M}}_{22}\right)^2 + 4\boldsymbol{\mathcal{M}}^2_{12}} \begin{pmatrix} \boldsymbol{\mathcal{M}}_{12} & -\left(\boldsymbol{\mathcal{M}}_{11} - \boldsymbol{\mathcal{M}}_{22}\right) \\ \text{sym.} & -\boldsymbol{\mathcal{M}}_{12} \end{pmatrix}, \tag{3.11}$$

$$\nabla^{\mathrm{T}}_{\boldsymbol{\mathcal{M}}} \boldsymbol{v}_{\{1,2\}} = \nabla^{\mathrm{T}}_{\boldsymbol{\mathcal{M}}} \boldsymbol{v}\Big|_{\boldsymbol{v}=\boldsymbol{v}_{\{1,2\}}}. \tag{3.12}$$

Note that Equation (3.11) is for the components 1 and 2 of a vector $\boldsymbol{v}$. It is then evaluated at $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Also note that $\pm$ and $\{1,2\}$ should be permuted simultaneously in the above expressions.

We now consider interpolation schemes that are based on Barycentric coordinates in 2D. In these cases, $\boldsymbol{\mathcal{M}}$ varies directly with the Barycentric coordinates $\eta_{\{1,2,3\}}$ at the position $\boldsymbol{x}$ and the nodal tensors $\boldsymbol{\mathcal{M}}_{\{1,2,3\}}$. Selecting η_1 and η_2 as the independent coordinates ($\eta_3 = 1 - \eta_1 - \eta_2$), the mapping from positions to coordinates is:

$$\begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{p}_1 - \mathbf{p}_3 & \mathbf{p}_2 - \mathbf{p}_3 \end{pmatrix}^{-1}}_{\mathcal{A}^{-1}} (\boldsymbol{x} - \mathbf{p}_3). \tag{3.13}$$

Then $\frac{\partial \boldsymbol{\mathcal{M}}_{jk}}{\partial x_l} = \frac{\partial \boldsymbol{\mathcal{M}}_{jk}}{\partial \eta_m} \frac{\partial \eta_m}{\partial x_l} = \frac{\partial \boldsymbol{\mathcal{M}}_{jk}}{\partial \eta_m} \mathcal{A}^{-1}_{ml}$. For the linear interpolation scheme, this is simply:

$$\frac{\partial \boldsymbol{\mathcal{M}}_{jk}}{\partial x_l} = \sum_{e=1}^{2} \tilde{\boldsymbol{\mathcal{M}}}^{(e)}_{jk} \mathcal{A}^{-1}_{el}, \quad \tilde{\boldsymbol{\mathcal{M}}}^{(e)} = \boldsymbol{\mathcal{M}}^{(e)} - \boldsymbol{\mathcal{M}}^{(3)}.$$

3.6 Metric Field Measures

In this work, we often use measures of a metric field $\boldsymbol{\mathcal{M}}$ at a specific point in order to characterize anisotropy, understand how well a mesh captures the requirements of a metric field, or perhaps identify regions of interest in the field. A simple measure is aspect ratio, which is usually taken as the ratio of the largest size to smallest size represented by a metric ($h_{\max}/h_{\min}$).

Another useful measure is fractional anisotropy (FA), which is often used for DT-MRI data [KTW07]. FA is based on the deviatoric decomposition of a tensor:

$$\tilde{\mathbf{A}} = \mathbf{A} - \frac{\operatorname{tr}(\mathbf{A})}{n}\mathbf{I}, \quad \mathbf{A} \in \mathbb{R}^{n\times n}.$$

The deviator is trace-less by construction, as we subtract the isotropic portion of the original tensor. Thus, the deviator vanishes for isotropic tensors. For a metric $\boldsymbol{\mathcal{M}}$ with deviator $\tilde{\boldsymbol{\mathcal{M}}}$, FA can be expressed as:

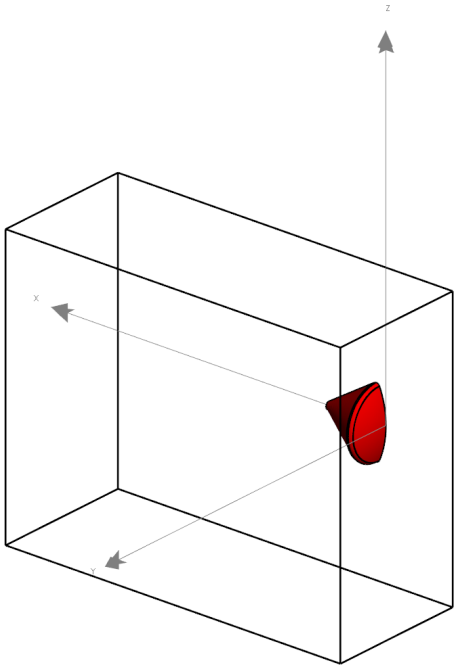
$$\mathrm{FA} = \sqrt{\frac{3}{2}}\frac{\left\|\tilde{\boldsymbol{\mathcal{M}}}\right\|_2}{\left\|\boldsymbol{\mathcal{M}}\right\|_2}.$$

As such, it is closely related to J_2 and the von Mises stress in solid mechanics. Figure 3.7 shows several visualizations of FA computed from solution-based metrics for a re-entry capsule problem.

Metric non-conformity is a measure that assesses how well a given mesh conforms to a metric field. In [LDV+04], it is defined as:

$$\epsilon = \left\|\boldsymbol{\mathcal{M}}_{\text{implied}}^{-1}\boldsymbol{\mathcal{M}}_{\text{field}} + \boldsymbol{\mathcal{M}}_{\text{field}}^{-1}\boldsymbol{\mathcal{M}}_{\text{implied}} - 2\mathbf{I}\right\|_2, \tag{3.14}$$

where $\boldsymbol{\mathcal{M}}_{\text{field}}$ is the representative metric from the field, and $\boldsymbol{\mathcal{M}}_{\text{implied}}$ is the metric implied from the mesh element under consideration. Implied metrics can be uniquely found for simplices by solving for components of a metric in which the edges of a simplex are

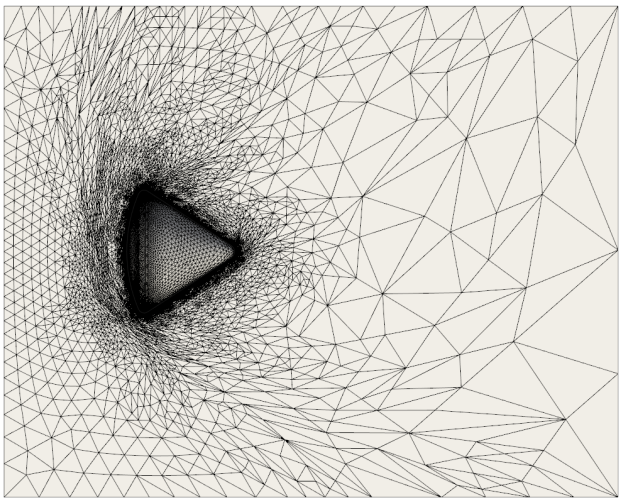


(a) Re-entry capsule geometry

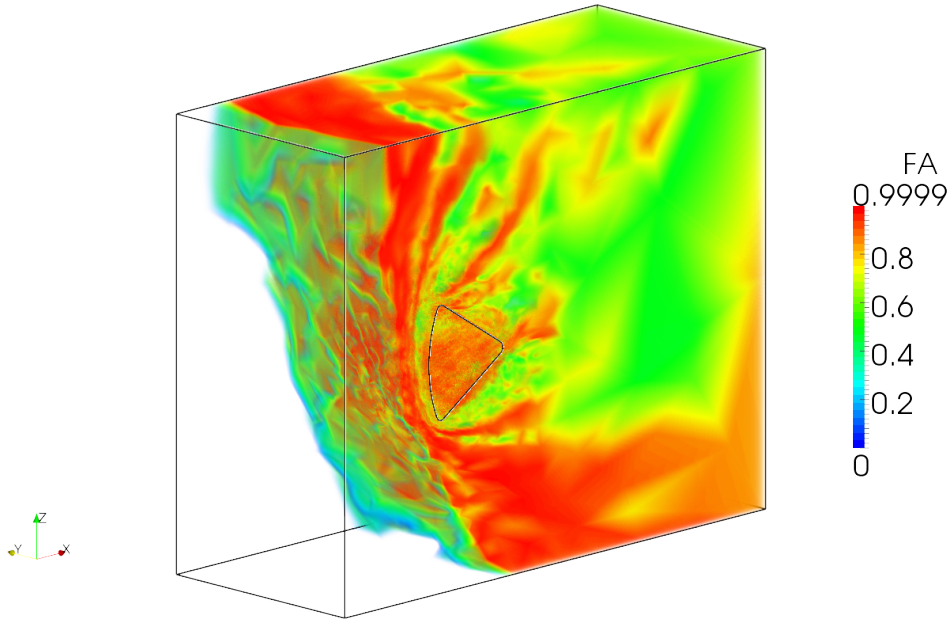
unit-length. This provides as many constraints as we have unknown independent metric components (three in 2D, six in 3D). Refer to [LDV+04] for details on the approximation of implied metrics for non-simplicial elements. The non-conformity measure is 0 when the field and implied metrics agree: $\epsilon = \left\| \boldsymbol{\mathcal{M}}^{-1}\boldsymbol{\mathcal{M}} + \boldsymbol{\mathcal{M}}^{-1}\boldsymbol{\mathcal{M}} - 2\mathbf{I} \right\|_2 = \|\mathbf{I} + \mathbf{I} - 2\mathbf{I}\|_2 = 0$. For a mesh adaptation iteration for the capsule problem, Figure 3.8 shows elements that exceed a non-conformity threshold ($\epsilon \geq 10$) with respect to solution-based metrics for generating the next iteration mesh.

3.7 Generation of Metric Fields

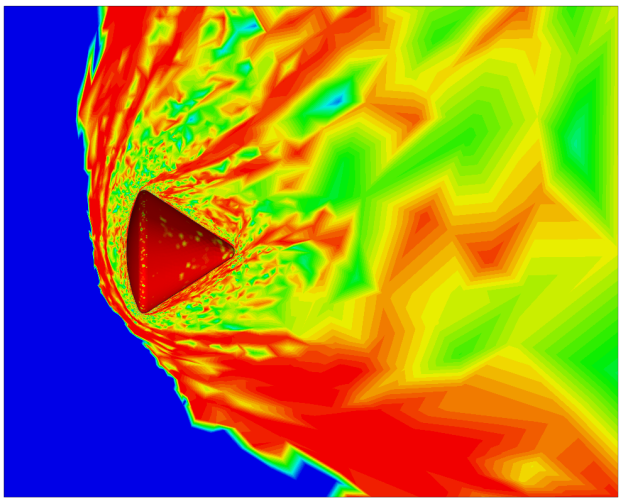
This section covers a proposed approach to the generation of metric tensor fields that are aligned with the boundaries of an input geometry. The generation of metric fields for solution-based adaptation is also briefly discussed.



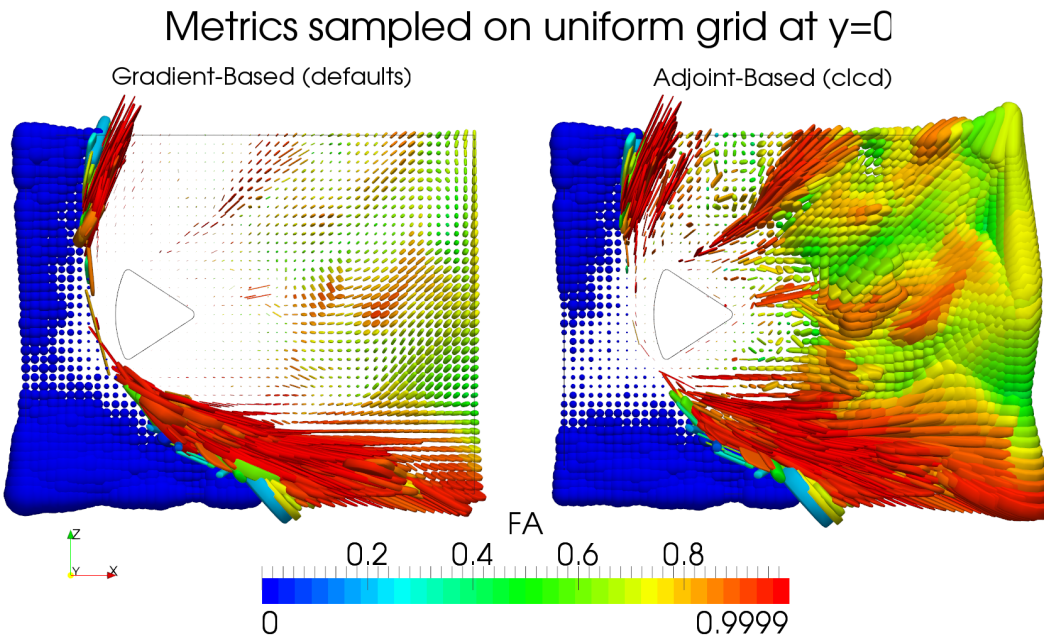
(b) Adapted mesh on which metrics were calculated



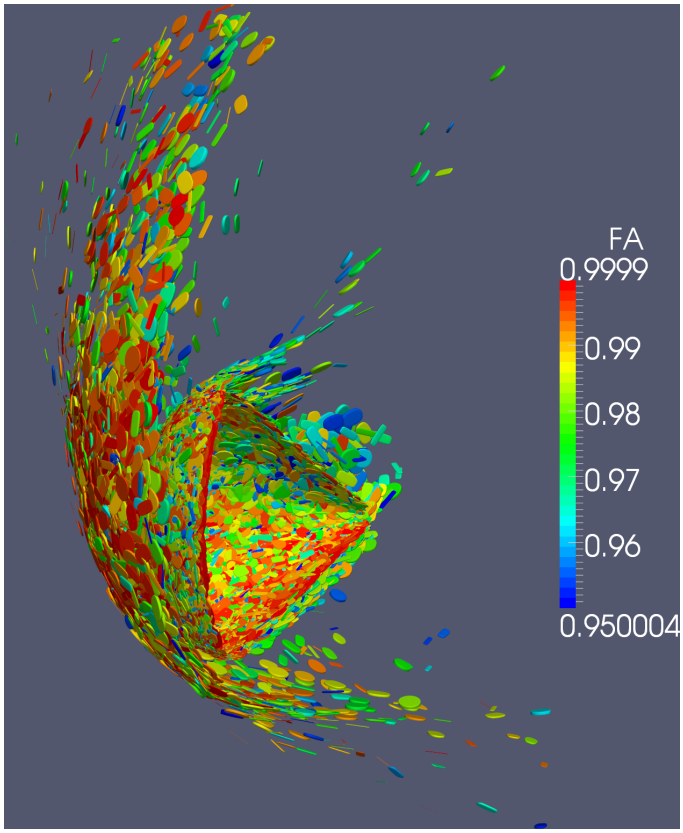
(c) Volume rendering of FA



(d) Surface plot of FA on $y = 0$ face

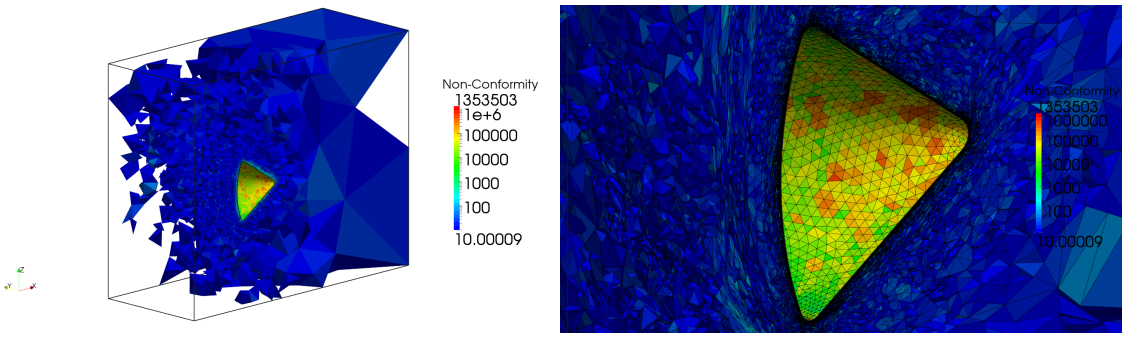


(e) Superquadric glyph visualization, colormapped by FA, of gradient-based and Adjoint-based metrics from FUN3D



(f) Superquadric glyphs, colormapped by FA, for very anisotropic regions where FA ≥ 0.95. This captures the bow shock, boundary layer region, and perhaps where flow separation occurs.

Figure 3.7: Visualizing gradient-based and adjoint-based metrics using fractional anisotropy (FA)



(a) Cells from the capsule problem that exceed a metric non-conformity threshold

(b) Zoomed-in view of metric non-conformity near capsule boundary

Figure 3.8: Visualizing metric non-conformity

3.7.1 Geometry-based

In two-dimensional problems, the goal is to construct a field that aligns with the boundary curves. If tensors are first specified on background mesh nodes that lie on boundary curves, the interior field can be populated by solving a partial differential equation (PDE). A simple choice is Laplace's equation, which is used to describe steady-state temperature distributions with spatially constant thermal conductivity and no internal heat sources. In the tensor case, the following equation is solved over the domain:

$$\nabla_{\boldsymbol{x}}^2 \mathcal{M}_{ij} = 0 \qquad \text{on} \quad \Omega \tag{3.15}$$

$$\mathcal{M}_{ij} = \mathcal{M}_{ij}^0(\boldsymbol{x}) \qquad \text{on} \quad \partial\Omega\,, \tag{3.16}$$

where Ω is the domain, $\partial\Omega$ is the boundary, and $\mathcal{M}_{ij}^0(\boldsymbol{x})$ is the prescribed field behavior on the boundaries (boundary conditions). Several discretizations of Equation (3.15) are possible. One of the choices made in the present work is the following stencil:

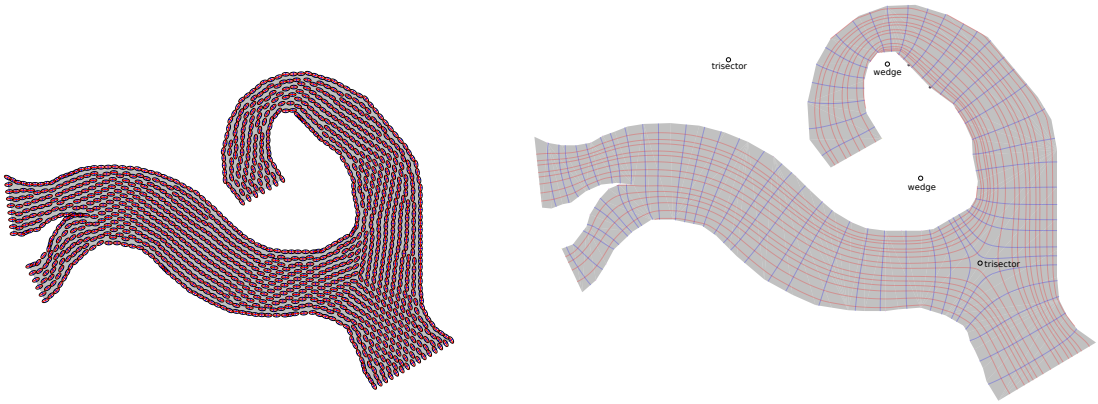
$$\boldsymbol{\mathcal{M}}^{k+1} = (1-\omega)\,\boldsymbol{\mathcal{M}}^k + \omega \left[\left(\sum_{j=1}^{N} l_j^{-1} \right)^{-1} \sum_{j=1}^{N} l_j^{-1} \boldsymbol{\mathcal{M}}_j^k \right], \tag{3.17}$$

where $\boldsymbol{\mathcal{M}}^k$ is the metric at the current node, $\boldsymbol{\mathcal{M}}^{k+1}$ is the updated metric at the current node, $\boldsymbol{\mathcal{M}}_j^k$ is the metric at the j-th neighbor of the current node ($j = 1, \ldots, N$), l_j is the length of the edge connecting the current node to its j-th neighbor node, and ω is a relaxation parameter. If $0 < \omega < 1$, this is an under-relaxed scheme [TAP97] (also used to smooth data), and $1 < \omega < 2$ allows for over-relaxation to accelerate convergence. The steps to apply this stencil are as follows:

1. Boundary conditions: tensors on boundary curve nodes are generated such that their eigenvectors match the local boundary curve tangents as best as possible. If isotropic sizing is desired, pseudo-isotropic tensors are chosen to avoid umbilics where there is no preferred orientation. One of the eigenvalues is perturbed (two eigenvalues are independently perturbed in 3D) by a very small amount to achieve this. The tensors on all boundary curve nodes are then set as fixed boundary conditions.
2. Initialization: non boundary condition metrics are initialized to $\mathbf{I}$.
3. Iteration: nodes of the background mesh are visited in a Gauss-Seidel fashion. The stencil is applied to each non boundary condition metric and node in multiple passes over the entire background mesh until the maximum relative error falls below a threshold. One termination criterion is $\left|\frac{\boldsymbol{\mathcal{M}}_{ij}^{k+1} - \boldsymbol{\mathcal{M}}_{ij}^{k}}{\boldsymbol{\mathcal{M}}_{ij}^{k}}\right| \leq \epsilon$. The solution procedure can be terminated if this is satisfied by all metrics at the end of a global pass.

An example of the results of this procedure is shown in Figure 3.9. In practice, this form of iteration converges slowly, so we assemble a global linear system by applying the stencil to nodes where the metrics are unknown. Both LU decomposition and stabilized biconjugate gradient (BiCGSTAB) methods are used to solve the systems depending on whether the domain is flat (planar 2D surfaces or 3D volumes) or curved (curved surfaces in 3D). For the former, LU decomposition may be favorable because the same system can be independently solved for each independent metric component (*i.e.*, same coefficient matrix with

different right-hand sides). The stencil for curved 3D surfaces (covered later) couples different components, so we resort to iterative solvers like BiCGSTAB. If fields are repeated generated on flat domains by making incremental changes to boundary conditions (refer to form-fitting approach mentioned later on), then it may make sense to use iterative solvers with the existing field as the starting guess.



(a) Metric ellipses (displayed with exaggerated anisotropy)

(b) Metric streamlines and umbilic points

Figure 3.9: PDE-based metric field generation in 2D

A slightly different, bottom-up procedure is employed to generate 3D fields. First, preferred orientation is set on feature curves and vertices. Next, preferred orientation is established on the boundaries in a way that smoothly adjusts to the feature curves. Finally, a volume field is generated by iteratively generating boundary conditions on the surface and updating the interior metrics. This procedure is designed to match the intuitive notion of how orientation should vary over a "principally blocky" volume.

Boundary Field Form-fitting In the first step, "scaffold triads" are placed along feature curves and at feature vertices. These triads are aligned with the local features as best as is possible, and specify required orientation but not eigenvector ordering or anisotropy. Many geometric features can be captured this way, however, the orthogonality of the triads limits

capture of features such as knife edges or a vertex with more than four incident feature curves.

Then, form-fitting is applied independently to each surface. For a given surface, one of its bounding curve's nodes is selected to begin. This node's metric tensor is initialized with the directions from its scaffold triad, and the following procedure is performed: The stencil

Algorithm 1 Form-fitting a metric field on a boundary surface

while not all bounding nodes have been visited **do**
  Get the metric tensor of the current node, and **snap its directions** to the closest directions on the node's scaffold.
  Mark the node as visited and as a boundary condition (BC) node.
  Solve an approximate surface Laplacian over the current surface, with the current set of tensor BCs.
  Move to the next unvisited, adjacent node on the bounding node-loop(s) of the surface.
end while

for the approximate surface Laplacian is:

$$\boldsymbol{\mathcal{M}}^{k+1} = \left(\sum_{i=1}^{N} w_i\right)^{-1} \sum_{j=1}^{N} w_j \left(\mathbf{T}_j \cdot \boldsymbol{\mathcal{M}}_j^k \cdot \mathbf{T}_j^{\mathrm{T}}\right), \tag{3.18}$$

where $\boldsymbol{\mathcal{M}}^{k+1}$ is the updated tensor at the current node, w_j is the Floater's mean value coordinate [PZ07] associated with neighbor node $j = 1...N$, $\mathbf{T}_j$ is the transformation from the surface normal at neighbor node j to the surface normal at the current node, and $\boldsymbol{\mathcal{M}}_j^k$ is the tensor at neighbor node j.

Interior Form-fitting By this point, all surface fields have been established. The fields of adjacent surfaces may not match up in terms of ordered eigenvectors, but they are directionally-compatible due to the shared scaffold triads used while generating them. The tensors on the interior of each surface are snapped to the local surface normal to ensure alignment, and then all surface tensors are converted to scaffold triads. A similar procedure is then employed to march over unvisited boundary nodes and complete the volume field.

Algorithm 2 Form-fitting a metric field to a volume

```
Pick a starting node on the boundary and generate a complete metric tensor with directions
from its scaffold.
Set this node as visited.
Push node onto boundary-node queue.
while boundary-node queue not empty do
  Set current node to the front node of the queue.
  Pop the front of the queue.
  Snap the current node's metric tensor to its scaffold directions.
  Set the current node as a BC node.
  Solve tensor-component Laplacian over volume with BCs.
  for each node adjacent to the current node do
    if current adjacent node is on the boundary and is unvisited then
      Push onto queue.
      Set visited flag.
    end if
  end for
end while
```

This concludes the description of the form-fitting procedure. After this, all surface tensors are set as boundary conditions to solve for the exterior volume field. This allows streamlines and surfaces to extend slightly past the boundary as necessary. The nodal tensors of the completed field are modified to represent the desired anisotropy, and the umbilics of the field are extracted as previously described.

3.7.2 Solution-based

Solution-based metrics are computed based on error estimates from the current solution. The idea is to minimize and equidistribute local error estimates by selecting appropriate orientation and anisotropy. In particular, we consider gradient-based (see also: adjoint-based) methods that rely on Hessian information from a solution variable to determine orientation and aspect ratio. That is then combined with a scalar intensity [BGPJ06, Par03].

Suppose a solution has been obtained on the current mesh. We then select two scalar solution variables, which will be referred to as $f_{\mathcal{H}}$ and f_{∇}. The Hessian $\mathcal{H}$ of $f_{\mathcal{H}}$, given by

$\nabla_{\boldsymbol{x}} \otimes \nabla_{\boldsymbol{x}} f_{\mathcal{H}}$, is in general a symmetric matrix. It has an eigen-decomposition $\mathbf{Q} \cdot \mathbf{\Lambda} \cdot \mathbf{Q}^{\mathrm{T}}$, where $\mathbf{Q}$ is orthogonal and $\mathbf{\Lambda}$ is a diagonal eigenvalue matrix. For convenience later on, the eigenvalues of $\mathbf{\Lambda}$ are ordered by *magnitude* ($|\lambda_1| \geq |\lambda_2| \geq |\lambda_3|$), and this is also reflected in $\mathbf{Q}$. We then define a somewhat normalized, positive-definite eigenvalue matrix $\hat{\mathbf{\Lambda}}$:

$$\hat{\mathbf{\Lambda}} = \begin{pmatrix} 1 & & \\ & \frac{|\lambda_2|}{|\lambda_1|} & \\ & & \frac{|\lambda_3|}{|\lambda_1|} \end{pmatrix}.$$

The ratios preserve the anisotropy originally present in the Hessian. Next, we compute a scalar adaptation intensity, I as a function of $\nabla_{\boldsymbol{x}} f_{\nabla}$ and other tolerances/parameters as necessary. An estimate of the original local mesh size, h_0, is scaled by an appropriate function of I to produce a new base size h_1 with corresponding eigenvalue h_1^{-2}.

Finally, a metric tensor $\boldsymbol{\mathcal{M}}$ can be constructed as follows:

$$\boldsymbol{\mathcal{M}} = \mathbf{Q} \cdot \left(h_1^{-2} \hat{\mathbf{\Lambda}} \right) \cdot \mathbf{Q}^{\mathrm{T}}. \tag{3.19}$$

The tolerances and parameters alluded to earlier can be used to cap the sizes, aspect ratio, and change from the previous local mesh size.

3.8 Topological Analysis of Metric Fields

Flow fields have critical points where the velocity vanishes and the local flow configuration is one of several types (source, sink, *etc.*). At each critical point, curves known as *separatrices* separate different regions of the flow around the critical point. Together, a collection of critical points and separatrices of the flow field provide a complete picture of its topology. A similar notion of *umbilics* exists for the SPD tensor fields considered in this work. Umbilics

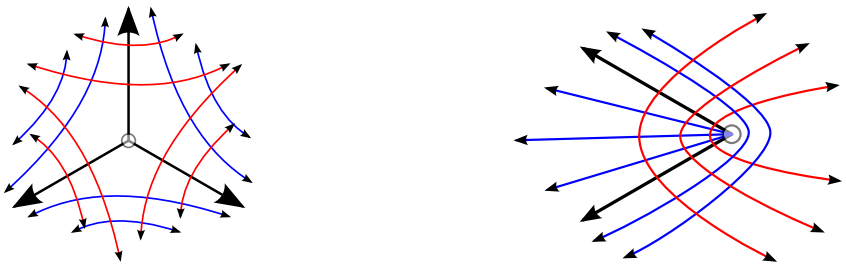
occur at points where the field becomes isotropic rather than vanishing (which cannot occur if positive-definiteness is maintained everywhere).

Umbilic extraction has been utilized extensively in the study of SPD tensor fields and their visualization [HLL97]. The following contains a brief review of the conditions for umbilic detection and a formulation for this when using piecewise-continuous metric fields on simplicial background meshes.

The derivation here begins with the most commonly occurring configuration in 2D, isolated umbilics. The goal is to search for a point in each background mesh triangle at which the interpolated metric is isotropic (scalar multiple of $\mathbf{I}$) and thus, its deviator is the null matrix. Substituting in the linear interpolation scheme for $\boldsymbol{\mathcal{M}}$ and enforcing $\sum \alpha_i = 1$ yields a linear system that applies to each triangle of the background mesh:

$$\begin{bmatrix} \left(\boldsymbol{\mathcal{M}}^1 - \boldsymbol{\mathcal{M}}^3\right)_{11} - \left(\boldsymbol{\mathcal{M}}^1 - \boldsymbol{\mathcal{M}}^3\right)_{22} & \left(\boldsymbol{\mathcal{M}}^2 - \boldsymbol{\mathcal{M}}^3\right)_{11} - \left(\boldsymbol{\mathcal{M}}^2 - \boldsymbol{\mathcal{M}}^3\right)_{22} \\ \left(\boldsymbol{\mathcal{M}}^1 - \boldsymbol{\mathcal{M}}^3\right)_{12} & \left(\boldsymbol{\mathcal{M}}^2 - \boldsymbol{\mathcal{M}}^3\right)_{12} \end{bmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = - \begin{pmatrix} \boldsymbol{\mathcal{M}}^3_{11} - \boldsymbol{\mathcal{M}}^3_{22} \\ \boldsymbol{\mathcal{M}}^3_{12} \end{pmatrix}, \quad (3.20)$$

where nodal metric indices are indicated using superscripts (not to be confused with contravariant indices). If a unique solution exists for Equation (3.20), then it simply remains to check if the point is in the triangle ($0 \leq \alpha_{1,2} \leq 1$). More pathological cases such as umbilics along open/closed curves and surface patches are possible due to the solution-based source of the metrics. From the equation it is evident that those cases correspond to a rank-deficient matrix and that the solution space will either be a line in α space or the α_1-α_2 plane. Since barycentric coordinates map linearly to positions in the 2D domain, those cases translate to lines clipped to the triangle or the entire area of the triangle, respectively. The latter is simple to detect; all nodal metrics will be isotropic. The former implies that two edges



(a) Trisector (b) Wedge

Figure 3.10: First-order umbilic types for planar metric tensor fields

of the triangle contain isotropic points and the line segment between them must be the umbilic geometry. This leads to a bottom-up procedure to locate umbilics: mark isotropic nodal metrics, then solve for isotropic metrics along edges and within triangles. The same procedure can be applied to the log-Euclidean and linear-size schemes if $ln\,\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{M}}^{-1/2}$, respectively, are substituted for the nodal metrics in Equation (3.20).

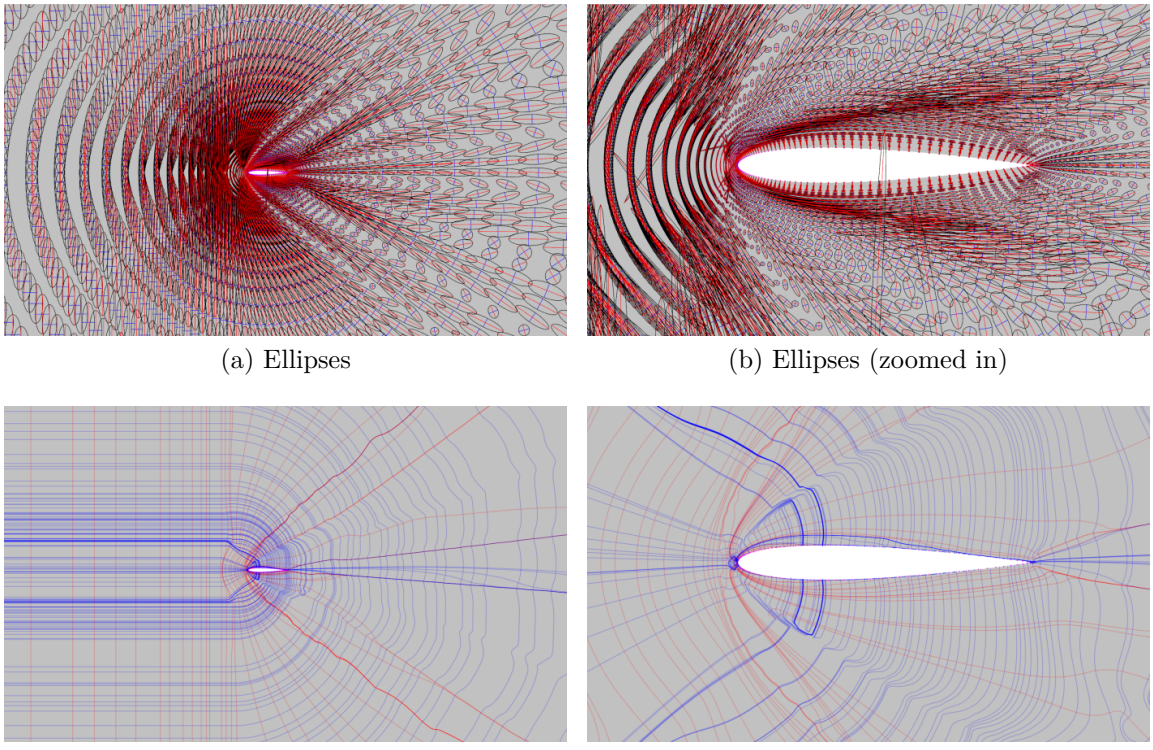
This process becomes more challenging in 3D, where the case of transverse (planar) isotropy causes the problem to become non-linear. We have adapted the second technique presented by Zheng et al. to extract connected umbilic structures (curves, surfaces, and sub-volumes) as well as discrete umbilics for full- and transverse-isotropy [ZPP05]. The theory of the extension for connected structures is similar to that described previously for 2D, however the system being solved is different.

A note on local eigenvector-field configurations around umbilic points: there are primarily two first-order umbilics in 2D, the trisector and the wedge. Both are shown in Figure 3.10, where the circles indicate the location of the umbilic, the black lines are the previously mentioned separatrices, the blue curves are streamlines of the major eigenvector-field, and the red curves are streamlines of the minor eigenvector-field. The meshing schemes presented in this work require umbilic location and type information because eigenvector directions change very rapidly in their proximity.

3.9 Conditioning of Metric Fields

Unlike the geometry-based fields discussed in Section 3.7.1, solution-based metrics are not well behaved. Simple observations of such metrics (refer to Figure 3.11) reveal two types of issues:

- Isolated, local noise: individual nodal metrics may be outliers in terms of orientation and/or anisotropy when compared to their local neighborhood. Several anisotropy outliers can be seen in Figure 3.11b.
- Global undulations in orientation: as seen in Figure 3.11c, metric streamlines around the bow shock are wavy.



(a) Ellipses

(b) Ellipses (zoomed in)

(c) Metric streamlines

(d) Metric streamlines (zoomed in)

Figure 3.11: Noise and undulations in solution-based metric data

Therefore, we seek methods that can accomplish the following:

- Removal of aspect-ratio and scaling outliers

- Removal of directional outliers
- Preservation of strong features that are present in the metric field
- Fairing of directional trends

Median filtering is a traditional image processing technique that is effective at removing noise while preserving important image features. When applied to a scalar grayscale image, for example, one replaces a grayscale pixel with the median value of the pixels in its neighborhood. In [WWB+07], Welk et al. extend the definition of a median to tensors and present two variants of the median filter for tensor-valued images.

The typical algorithm for computing a median of a set of scalar values is to sort them by value and then select the middle value. It is not as straightforward to sort SPD tensor data in this manner, so Welk et al. consider an alternative definition: the median element of a set of values is the value that has a minimum summed distance to the other elements. This can be stated as the following for a set of values $\mathbb{X} = \{x_1, \ldots, x_N\}$:

$$x_{\text{median}} := \arg\min_{x \in \mathbb{X}} \sum_{j=1}^{N} \|x - x_j\|, \tag{3.21}$$

where x_{median} is the median value and x is restricted to the set $\mathbb{X}$ of elements x_j. For scalar values, the absolute value is the same as the L^1 norm and is consistent with the sorting method for the median.

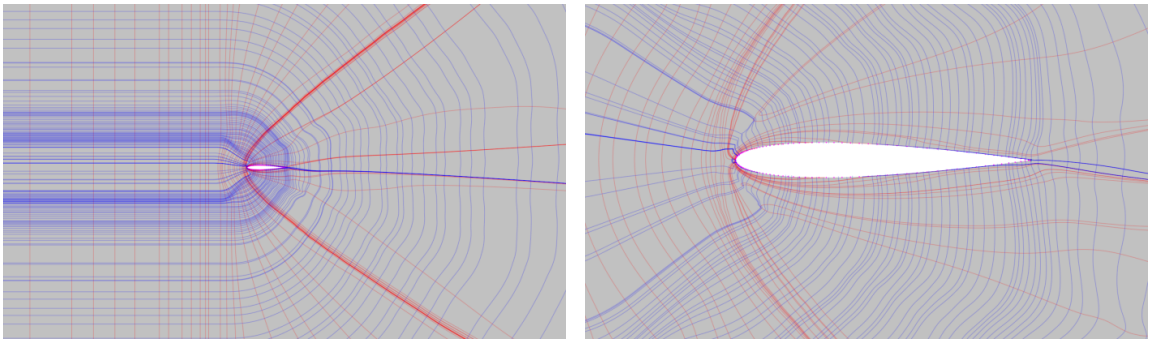
We now discuss specific choices for the filtering of solution-based metrics. Out of several possible norms, the Frobenius, or Euclidean, norm has been selected for the metric norm in Equation (3.21). It has the ideal property of being a tensor invariant. Unlike diffusion tensor data, metric data is non-uniformly sampled over space. To compensate for this, the local neighborhood around a node can be built by including all neighbors within a certain radius. Neighborhood construction has been designed to satisfy criteria such as minimum number of layers (n-ring), a neighborhood radius, and a minimum number of

neighbors. Furthermore, a Gaussian distance weighting scheme is applied to the terms of the summation in Equation (3.21) to smoothly factor in distance between the metric samples in a neighborhood. This yields the following median around a metric for a set of metrics $\mathbb{M} = \{\boldsymbol{\mathcal{M}}_1, \ldots, \boldsymbol{\mathcal{M}}_N\}$ at corresponding positions $\mathbb{X} = \{x_1, \ldots, x_N\}$:

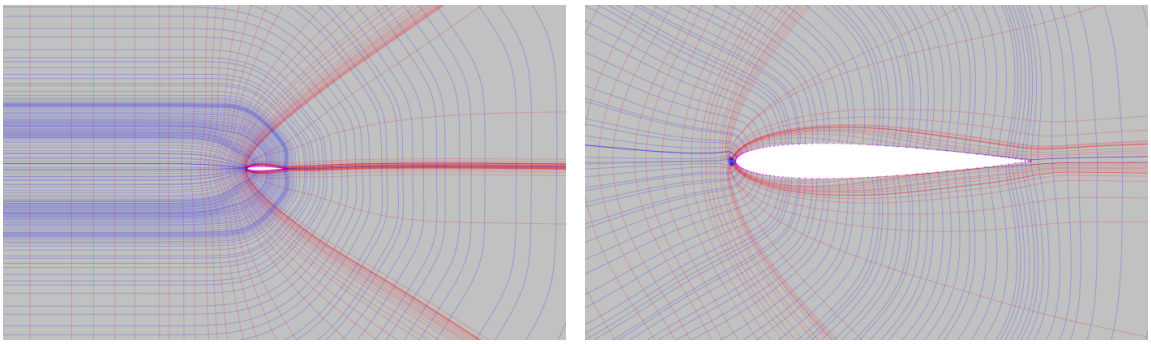
$$\begin{aligned} \boldsymbol{\mathcal{M}}_{\text{median}} &:= \arg\min_{\mathfrak{m} \in \mathbb{M}} \left[\left(\sum_i^N \omega_i \right)^{-1} \sum_i^N \omega_i \left\| \mathfrak{m} - \boldsymbol{\mathcal{M}}_i \right\|_2 \right] \\ \omega_i &= \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2}{2\sigma^2} \right) \\ \sigma^2 &= \frac{1}{N} \sum_i^N \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 , \end{aligned} \tag{3.22}$$

where $\mathfrak{m}$ is a dummy variable, and the Gaussian weights ω_i bias the computation with respect to a metric at position $\boldsymbol{x}$. This is also reflected in the variance, σ^2. Technically, the sets $\mathbb{M}$ and $\mathbb{X}$ may contain the metric that is currently being filtered, *i.e.*, $\boldsymbol{\mathcal{M}}$ at $\boldsymbol{x}$. A continuous extension is also used, where the restriction that $\mathfrak{m} \in \mathbb{M}$ is relaxed; this allows the median to take any symmetric positive-definite value (new restriction: $\mathfrak{m} \in \mathbb{S}^+_{2\times 2}$). Solving this form requires continuous optimization. We use gradient descent with adaptive step size control and the weighted mean of the metrics in $\mathbb{M}$ as an initial guess.

The discrete and continuous median filters are applied using Jacobi-style iterations over a specified number of global passes over the domain. Figure 3.12 shows the effects on the metric streamlines after one and ten passes of median filtering on the raw metrics from Figure 3.11. In practice, filtering is applied conservatively (one pass of discrete median) to produce a rectified field for length measurements. Then, a more generous application (ten passes of continuous median filtering) is used to generate a field with faired orientation. Finally, Figure 3.13 provides an overall perspective of the effects on orientation. Figure 3.14 shows the results of passes of discrete medial filtering on metrics from the 3D re-entry capsule problem from earlier.

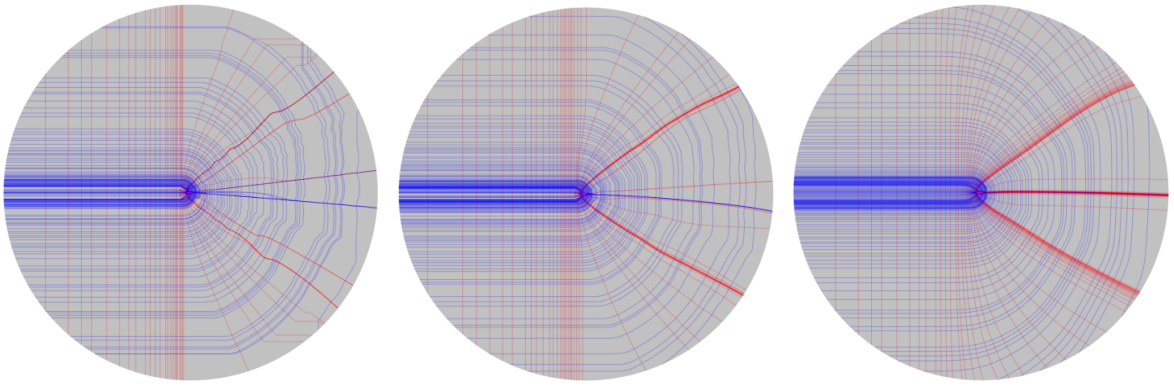


(a) One pass of continuous median (b) One pass of continuous median (zoomed in)



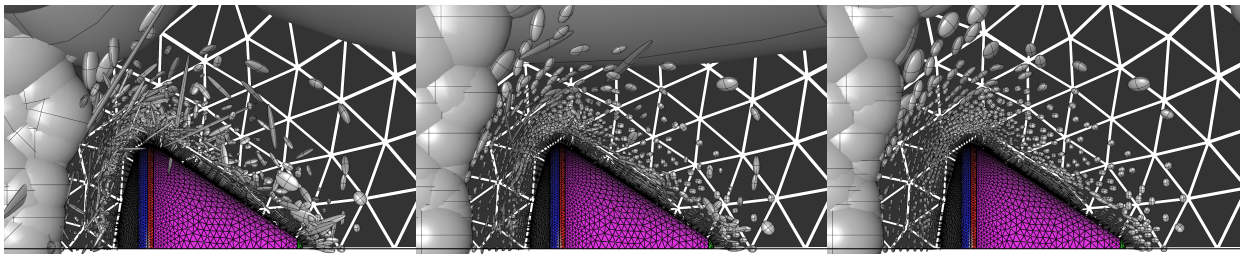
(c) Ten passes of continuous median (d) Ten passes of continuous median (zoomed in)

Figure 3.12: Discrete and continuous, Gaussian weighted median filtering of solution-based metric fields



(a) Raw (b) One pass continuous median (c) Ten passes continuous median

Figure 3.13: Global view of metric streamlines after filtering



(a) Unfiltered metrics (b) One pass discrete median (c) Five passes discrete median

Figure 3.14: Global view of metric ellipsoids after median filtering for re-entry capsule problem

Chapter 4

Metric Field Guided Mesh Generation Algorithms

In this chapter, we present a general strategy to construct unconstrained meshes that respect the anisotropy and orientation requirements from a metric field. The process utilizes the structure of the fields to define the output meshes. As input to the process, a metric field with spatial support and geometry are required. Multiple variants of the field can be constructed by applying various levels of conditioning; certain fields can be utilized for anisotropy queries, and others for orientation. If the metric field is solution-based, the solution can also be utilized. Depending on the target application, feature detection can be performed on the metric field and/or solution to extract the regions of interest to operate within.

We begin by presenting an initial approach that was developed in 2D and extended to 3D (Sections 4.2 and 4.3, respectively). In this approach, streamlines or streamsurfaces are locally generated to facilitate an advancing front technique. Elements are grown from seed points to form a complete mesh or disjoint mesh chunks which may be combined with a quad/hex-dominant meshing scheme. This approach will be referred to as "local." The local methods have primarily been applied to pseudo-isotropic fields that are generated from the

input geometry.

An analysis of the local methods in 2D and 3D leads to a "global" 2D approach that is presented in Section 4.4 (with a 3D extension suggested in Section 7.1). The main idea of this approach is to generate an arrangement of streamlines or streamsurfaces that are spaced appropriately, and then to identify the points of intersection of the arrangement. This provides an intersection graph, which may be traversed to form elements. Processing of the arrangement and resulting mesh yields valid quad-dominant and hex-dominant meshes. The global nature of this approach will become clear as the algorithms are described. An initial form of this approach has been implemented and successfully demonstrated in two-dimensions.

4.1 Rationale

From the inherent drawbacks and strengths of the methods mentioned in Section 2.2, initial observations suggest that a successful hex-dominant meshing strategy can be formulated using the following concepts:

1. Using fields to control orientation and anisotropy.
2. Tailoring such fields to be boundary aligned and to incorporate additional orientation and anisotropic size specifications.
3. Meshing from the inside out by starting at a suitable location(s).
4. Deciding marching directions for meshing ahead of time (via the field).
5. Meshing element by element in such a way that the mesh remains topologically valid and does not march into itself.
6. Complementing difficult regions for all-hex with good quality hex-dominant regions.

Some of these points, such as the need for a boundary sensitive overlay for grid-based

methods, have been noted by Blacker for all-hex meshing [Bla00]. Furthermore, field generation has been performed for application to other methods such as BubbleMesh [YS00, SLI98, VSmI00].

The first class of methods attempted in this work utilize these desired characteristics through field generation and application of topological insertion operators to insert elements and grow the mesh outward from a starting interior point. The metric field incorporates boundary normal information and user-input uniform anisotropy, and is capable of providing information on element shaping and marching directions for meshing. This allows for a relatively simple tiling algorithm based on topological insertion operators. The insertion operators maintain mesh integrity while attempting to prevent the mesh from marching into itself. They also govern the shaping and sizing of elements via specific intersections of streamsurfaces of the eigenvector fields that are obtained from the metric field.

4.2 Local Metric Field Guided Methods in 2D

Local mesh growing in 2D requires a planar geometry with feature vertices and a guiding metric field. Fields constructed as per Section 3.7.1 mostly guarantee a smooth variation in anisotropy and orientation (though not in the C^2 continuity sense, perhaps just continuous) that is boundary aligned and has few umbilic points.

Advancing front schemes usually start from boundaries. This allows boundary features to be captured, and it also provides support; marching directions for element placement start from the boundary normals and evolve as the mesher proceeds toward the interior. An important consequence is that orientation control is limited — it is primarily governed by the boundaries and marching direction. Furthermore, meshing fronts may collide as they move toward the interior. This sometimes requires many checks and stitching or cleanup operations to prevent mesh overlap and/or cracks.

On the other hand, our scheme has the benefit of operating with a field that will provide appropriate orientation at any point in the domain. We can, for instance, start from the interior and grow a mesh outward. As the mesh approaches the boundaries, it will naturally line up with the boundaries. Once the mesh actually reaches the boundaries, it can be projected or made conformal to them. Grid-based methods ([SSW96]) have a similar step. However, their interior elements are not boundary aligned, which results in poor quality of the last layer of elements at the boundary. Our method mostly circumvents this issue.

Topological Insertion Operators First, we discuss topological concerns that help organize element insertion. Suppose we have a patch of quadrilateral elements that is simply connected. The patch has an exterior loop of edges that are each used by only one element. This is called the skin of the patch and is given a positive orientation. A newly inserted element will reuse between one and three edges on the skin, and it is important to process these cases in the right order. This concept is illustrated in Figure 4.1, which shows the first three insertion operators. These are the primary operators that introduce new nodes into the mesh. Otherwise, there is one other insertion that places an element using three skin edges and their nodes.

The nodes on the skin of the patch are colored according to their classification: convex corner, side, or reversal. The classifications can be defined in terms of connectivity, where a convex corner node is used by one element, a side node by two elements, and a concave corner node by three elements. Alternatively, a skin node can be classified by the total included angle of the elements incident on it. The current node classifications can be used to determine what type of insertion is appropriate along a portion of the skin. Cracks and mesh self-intersection are almost completely avoided by applying insertion operators in the right order, from highest to lowest complexity.

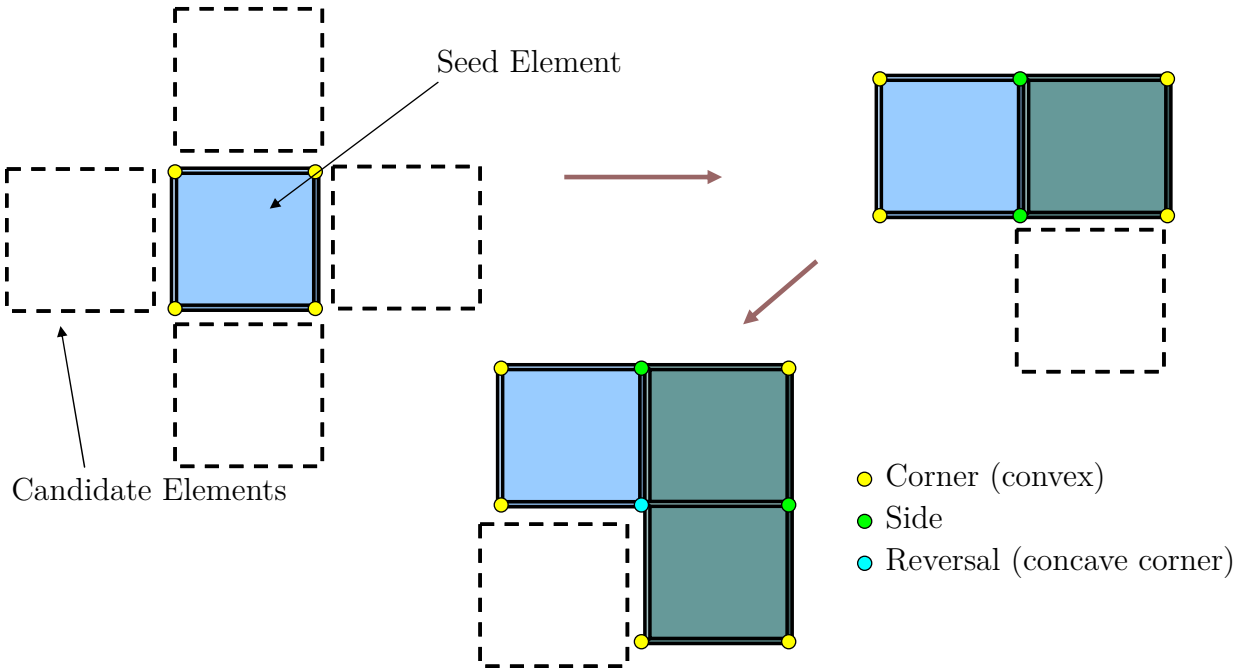


Figure 4.1: Topological insertion operators: seed, side, and reversal insertions in sequence

Element Shaping The previous discussion of insertion operators established an order to process element insertion. The next task is to determine the shape of new elements according to the metric field, and as alluded to in Section 2.1, it would be ideal to obtain a unit-mesh. The first insertion of a seed element begins with an insertion location $\boldsymbol{x}$. Four streamlines are integrated from $\boldsymbol{x}$ along both directions ($\pm$) of both eigenvectors $\boldsymbol{q}_1(\boldsymbol{\mathcal{M}}(\boldsymbol{x}))$ and $\boldsymbol{q}_2(\boldsymbol{\mathcal{M}}(\boldsymbol{x}))$, such that each streamline is of length $l = 0.5$ according to the metric field. This is numerically approximated by accumulating the length of each streamline segment. Streamlines are then integrated along the alternate eigenvectors' directions from the endpoints of the four original streamlines. This configuration is shown in Figure 4.2, where nodes are created at the four resulting intersections. For the remaining insertion types, it follows to determine the eigenvector directions that best match the current skin edge directions and normals. Shaping a *reversal* insertion is shown in Figure 4.3.

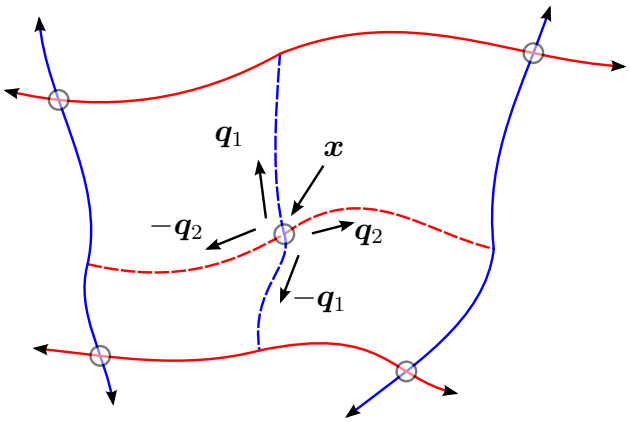


Figure 4.2: Shaping a seed element

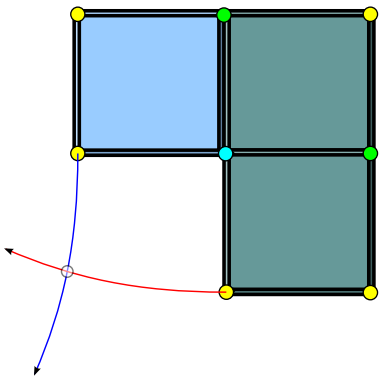
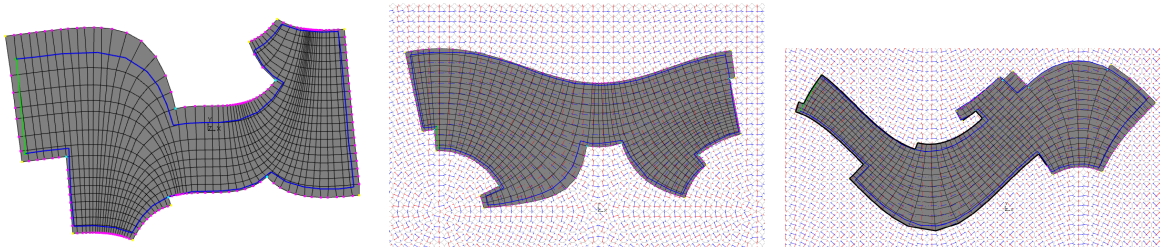


Figure 4.3: Streamline shaping of other insertion types

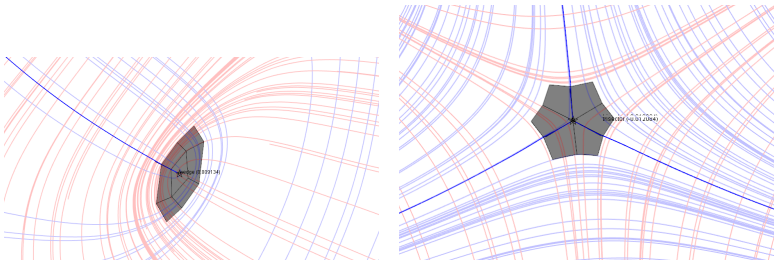
Initial Results and Observations Several structured quadrilateral mesh results are shown in Figure 4.4. Boundary conformity operations have not been used in these examples. Note that while the meshes are able to follow the gently varying orientation over the domain, they are not able to capture the requested sizes (as indicated by the metric ellipses). One reason is the convergence and divergence of the eigenvector-field directions. Furthermore, intersections often fail around umbilics because streamlines may diverge and fail to intersect. Thus, elements cannot be formed. Two solutions are introduced: umbilic meshing templates and extended insertion operators. The umbilic meshing templates let the meshing process start around umbilic points. Their natural structure is used to define a suitable element configuration (shown in Figure 4.5). However, this is still a problem if there



(a) Example 1 (b) Example 2 (c) Example 3

Figure 4.4: Basic results of local 2D method with pseudo-isotropic field

are multiple umbilics — that would require multiple mesh fronts. The extended insertion



(a) Wedge template (b) Trisector template

Figure 4.5: Mesh templates for umbilics

operators in Figure 4.6 allow size transitions to take place as marching proceeds: A selection of final results for the local 2D method are shown in Figure 4.7. The results have been post-processed with Laplacian smoothing.

4.3 Local Metric Field Guided Methods in 3D

In the following sections, we discuss the extension of the previously described local method to 3D [VS09]. Once again, the key to these methods is to separate the tasks of controlling element shape, selecting marching directions, and obtaining boundary alignment from the meshing algorithm. This allows for a simpler meshing algorithm, while incurring more effort during metric field generation.

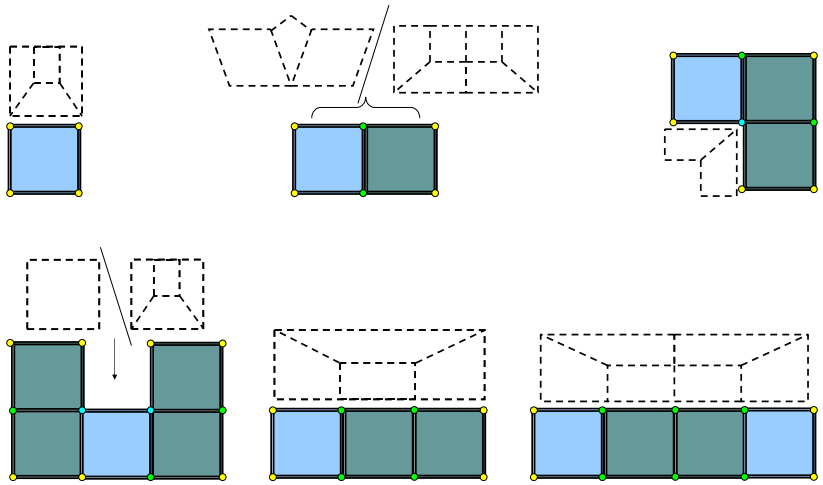
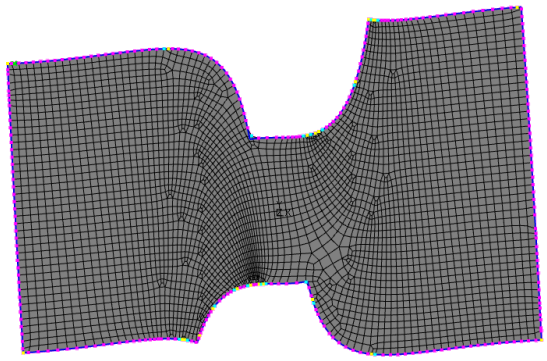


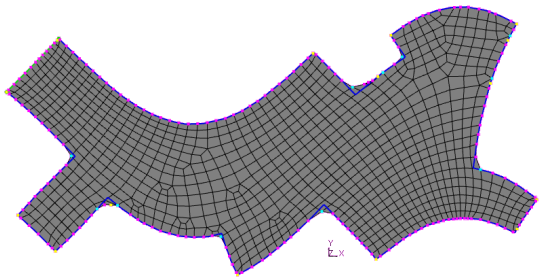
Figure 4.6: Extended insertion operators

Individual hex fronts are generated in regions that are free of umbilics and in regions of interest. A hex front is initiated by inserting a seed element on the interior or just within the boundary of the volume. This exposes six hex faces on the "skin," or collection of 2D elements on the boundary of the current hex mesh. The next elements to be inserted will use some of the current skin faces and add new ones, after which the skin is updated. As the skin grows, different combinations of existing skin faces may be used to form new elements. To avoid combinations that might lead to self-intersection or topological invalidity, insertion operators are used to plan and execute insertions that utilize groups of existing skin faces, referred to as insertion face groups (IFG). The IFGs, in turn, are identified by studying their skin nodes. When an operator is used to perform an insertion, depending on the IFG type, quarter-bands are appropriately placed and intersected to obtain the new nodes that define the target element shape. As appropriate, boundary-conformity operations may be invoked to associate proximal mesh entities with boundary features.

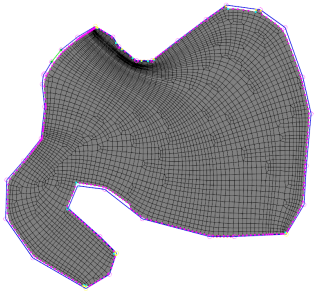
Once the fronts have been tiled in ROIs and away from field umbilics, there are two possible strategies for completing the hex-dominant mesh. The first is to perform a Boolean subtraction of the fronts from the volume, generate a hex-dominant mesh in the remaining volume, and then unite the meshes. This is similar to the Hex-Tet algorithm by Meyers



(a) Example 1



(b) Example 2



(c) Example 3

Figure 4.7: Final results for local 2D method

et al. [MTT98]. The second is to incorporate the front nodes into the rectangular bubble-packing process, which has been selected for this work. Although it does not guarantee 100% hex recovery, in practice the distribution of front nodes leads to significant recovery. Furthermore, it may allow for better overall element quality and transitions between hex and hex-dominant regions. The potential for non-conformal configurations with hanging-nodes and other complications noted in [YS03a] are also avoided by taking this route.

4.3.1 Preliminaries

The overall strategy is described in this section, which can be decomposed into three main steps:

1. Metric Field Generation: An iterative technique is used to solve for metric tensors on

the nodes of a background mesh. The technique produces a boundary-aligned field, which is then adjusted to uniform anisotropy. With appropriate interpolation, this provides a piecewise-linear field over the volume. This was reviewed in Section 3.7.1.

2. Field-Guided Hex Tiling: The element shaping strategy attempts to form unit hexes that are aligned with the metric eigenvector streams and approximately adhere to the lengths encoded in the metric eigenvalues. This is achieved by creating new hex nodes at the intersections of approximately unit length streamsurface triplets. The meshing process begins by inserting a seed hex on the interior of the volume. During tiling, insertion operators determine how new hexes are extended from the current mesh. This continues until the boundaries are reached, at which point new elements are locally snapped to boundary surfaces, feature curves, and feature vertices to enable boundary conformity.

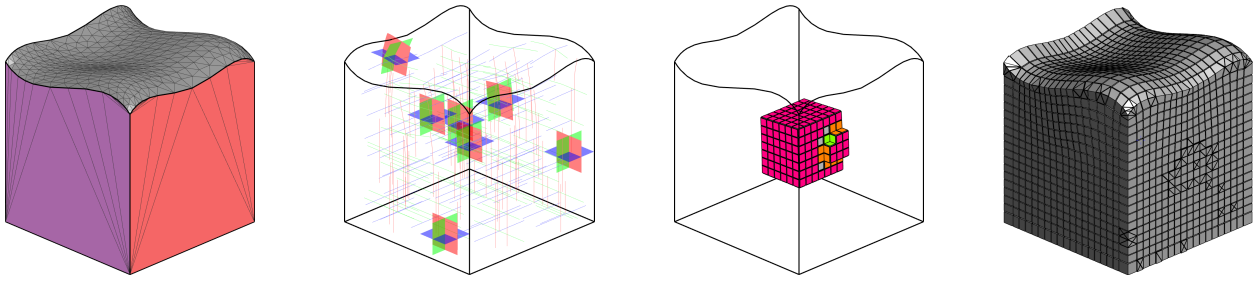
3. Hex-Dominant Mesh Generation: Packing of rectangular solid cells is augmented to incorporate the tiled hex fronts. The hex nodes are packed as fixed cells, and with some specific pre- and post-processing, the process is continued to obtain a hex-dominant mesh.

Figure 4.8 below depicts some of the main steps of the whole process, in the order described above.

4.3.2 Formation of Local Streamsurfaces

Streamlines are generated by specifying a starting point, an eigenvector index-sign pair (σ, e), and a desired length under the metric field. Fourth-order, fixed step-size Runge-Kutta [PTVF92] is used to generate successive points along a streamline while accumulating the metric-lengths of the new segments.

Due to the sign ambiguity of eigenvectors, the direction that most closely matches the



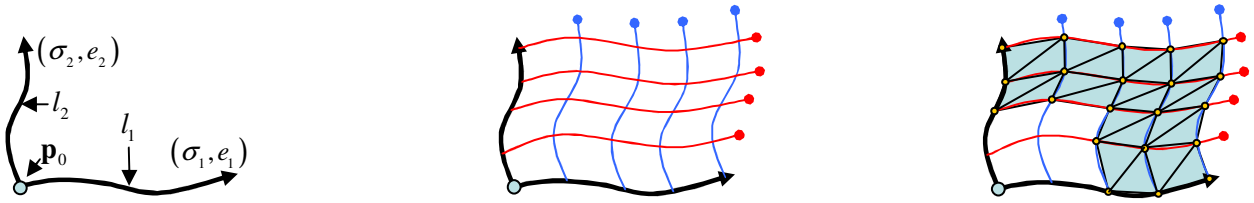
(a) Input geometry

(b) Streamlines and streamsurfaces of the generated tensor field

(c) Intermediate stage of hex front tiling

(d) Final hex-dominant result

Figure 4.8: Overview of the whole process



(a) Primary streamlines emanating from the initial point

(b) Secondary streamlines emanating from the primary streamlines

(c) Final triangulated quarter-band obtained from streamline net

Figure 4.9: Formation of streamsurfaces ($n = 4$)

established marching direction is chosen; similar considerations are made in [TDVC04]. The process is stopped once the measured length meets the desired length, or if the integrator is stalled. Depending on the use, either the entire polyline or just the endpoint is stored.

The next step is generating streamsurfaces along two eigenvector directions, (σ_1, e_1) and (σ_2, e_2), with nominal lengths l_1 and l_2, respectively (see Figure 4.9). Primary streamlines are traced according to these initial directions and lengths from the initial point, $\boldsymbol{p}_0$. The primary streamlines are then resampled to have points including the original first and last points.

From each resampled point on the (σ_1, e_1) streamline, a sufficiently long streamline is generated in the local direction corresponding to (σ_2, e_2), and vice-versa. These secondary

streamlines should be sufficiently long to account for rapidly converging/diverging eigenvector directions.

Due to the nature of these fields, it is not necessary for the secondary streamlines to intersect. Therefore, an optimization process is used to find the closest points on each pair of streamlines. Let $\boldsymbol{poly}_1$ and $\boldsymbol{poly}_2$ be arrays of m_1 and m_2 points, respectively, that contain the points from each streamline. Then the following parameterization provides a piecewise-continuous representation of each $(i = 1, 2)$:

$$\boldsymbol{p}_i(u) = \boldsymbol{poly}_i[u_0]\,(1 - u') + \boldsymbol{poly}_i[u_0 + 1]\,(u')\ , \tag{4.1}$$

where $u_0 = floor(u)$, $u' = u - u_0$, and $u \in [0, m_i - 1)$. One unit interval in the parameter space spans one line segment. Defining separate parameters for each curve, s and t, we can formulate an objective function from the squared-distance between the two curves:

$$f = \|\boldsymbol{p}_1(s) - \boldsymbol{p}_2(t)\|^2\ . \tag{4.2}$$

A steepest descent scheme with adaptive step-sizing is used to find the parameters $t^\star$ and $s^\star$ that minimize this function. The gradient of f can be written as:

$$\nabla f = 2\left([\boldsymbol{p}_1(s) - \boldsymbol{p}_2(t)] \cdot \dot{\boldsymbol{p}}_1(s), [\boldsymbol{p}_1(s) - \boldsymbol{p}_2(t)] \cdot \dot{\boldsymbol{p}}_2(t)\right)^{\mathrm{T}}\ . \tag{4.3}$$

Central differencing is used to evaluate the tangent vectors of the two curves, $\dot{\boldsymbol{p}}_1(s)$ and $\dot{\boldsymbol{p}}_2(t)$. The average of the locations evaluated from the optimal parameters is returned as the closest point.

The net of closest points forms a grid, which is split into triangles to build the streamsurface. In practice, the portion of the net within some length along the primary directions is not generated to lower cost. The streamsurfaces generated in this manner also occupy a

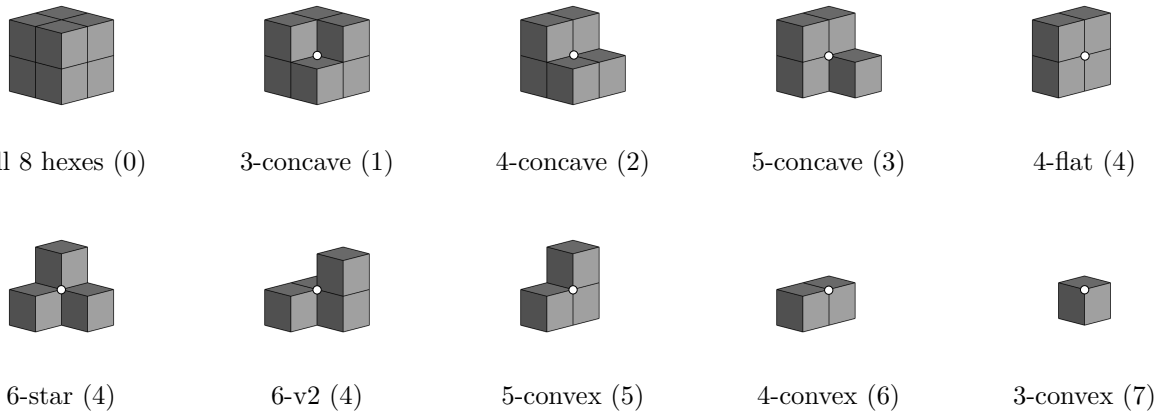


Figure 4.10: Skin vertex types

single "quadrant," so they are referred to as "quarter-bands."

4.3.3 Topological Insertion Operators and Face Groups

The rules that determine what types of elements to insert are fundamentally based on the local element-vertex connectivities of the skin vertices and faces. These vertex types can be enumerated by considering a central vertex in a structured hexahedral mesh. The vertex is initially surrounded by eight elements, and by removing different permutations of elements, a unique set of configurations of skin faces using the vertex can be obtained. Figure 4.10 below lists the types considered in this work. Some configurations have been excluded because they naturally do not arise due to the way elements are inserted; for instance, configurations where two hexes only share one edge or where there is a hexahedral through-hole or void.

The number of elements removed from the original eight is indicated in parentheses next to each vertex type name. In order, these vertex types will be referred to as: **3c**, **4c**, **5c**, **4f**, **6***, **6v2**, **5v**, **4v**, and **3v** from here on.

We will now present the topological insertion operators. Each operator places an element on the faces of a corresponding IFG. For each operator, a set of valid vertex types has been

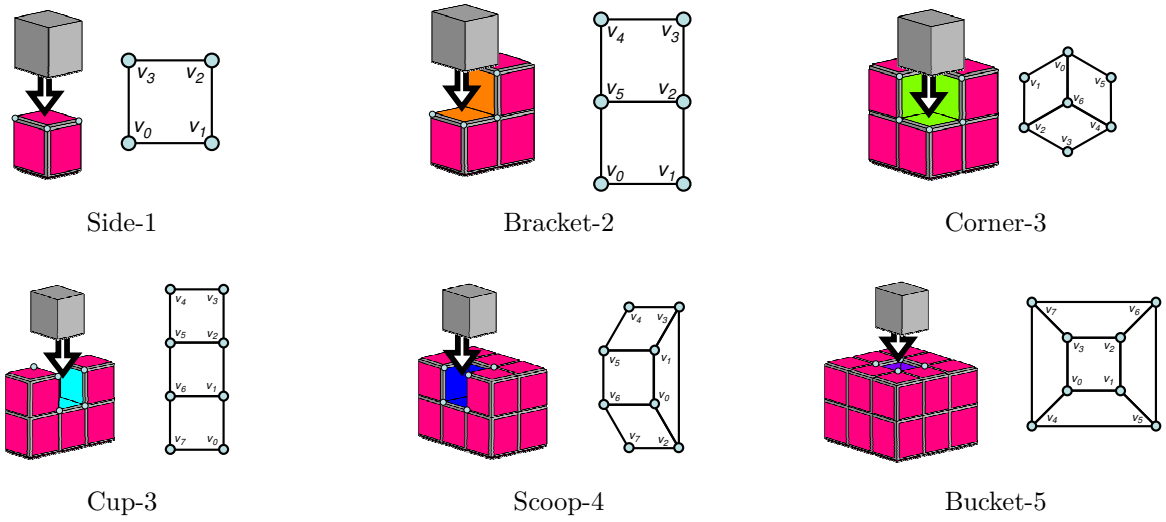


Figure 4.11: Topological insertion operators

determined for each vertex of the original skin faces that the element uses. The set of insertion operators and their IFGs is given in the figure below. Each possible insertion (left) is shown with its corresponding IFG (right) in Figure 4.11, and the vertex requirements for IFGs are enumerated in Table 4.1.

4.3.4 Planning and Scheduling Insertions

In this section, we discuss the precedence among the operators at a particular spot and how potential insertions over the entire skin are processed.

Due to the nature of the vertex type requirements, more complex insertions automatically precede simpler ones over an applicable skin region. This prevents some cases in which the mesh propagates into itself or forms sharp cracks. If vertices are instead classified by geometric criteria such as dihedral angles between the faces containing a vertex, then insertions should be searched for starting from the most complex one (one that is not contained in any other). One possible ordering is: *Bucket-5*, *Scoop-4*, *Cup-3*, *Corner-3*, *Bracket-2*, and finally, *Side-1*. This has been adopted as the current insertion precedence.

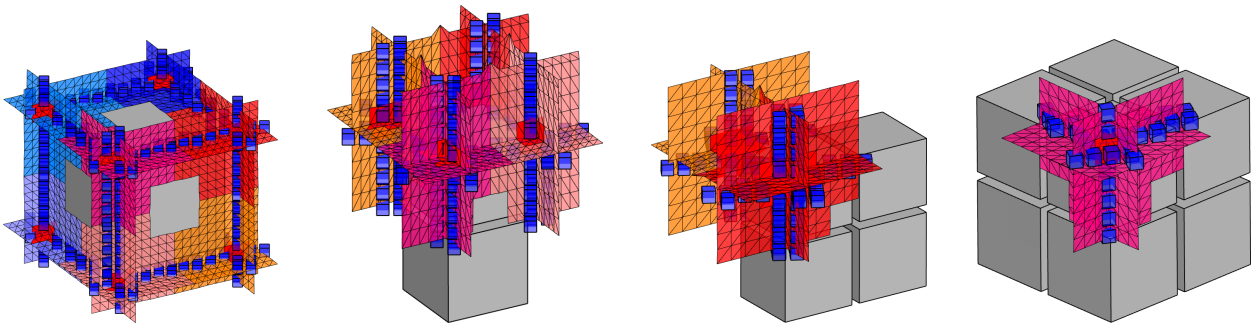
Table 4.1: Vertex type requirements for IFGs

IFG type	Vertex type requirements
Side-1	$\text{Type}(\{v_0, v_1, v_2, v_3\}) \in \{\mathbf{4f}, \mathbf{5v}, \mathbf{4c}, \mathbf{3v}\}$
Bracket-2	$\text{Type}(\{v_2, v_5\}) \in \left\{ \begin{array}{l} \mathbf{4c}, \mathbf{5c}, \mathbf{6}*, \\ \mathbf{6v2}, \mathbf{5v} \end{array} \right\}$, $\text{Type}(\{v_0, v_1, v_3, v_4\}) \in \left\{ \begin{array}{l} \mathbf{4f}, \mathbf{5v}, \\ \mathbf{4v}, \mathbf{3v} \end{array} \right\}$
Corner-3	$\text{Type}(v_6) = \mathbf{3c}$, $\text{Type}(\{v_0, v_2, v_4\}) \in \{\mathbf{4c}, \mathbf{5c}, \mathbf{6}*, \mathbf{6v2}, \mathbf{5v}\}$, $\text{Type}(\{v_1, v_3, v_5\}) \in \{\mathbf{3v}, \mathbf{4f}, \mathbf{4v}, \mathbf{4c}, \mathbf{5v}\}$
Cup-3	$\text{Type}(\{v_1, v_2, v_5, v_6\}) \in \{\mathbf{4c}, \mathbf{5c}, \mathbf{6}*, \mathbf{6v2}, \mathbf{5v}\}$, $\text{Type}(\{v_0, v_3, v_4, v_7\}) \in \{\mathbf{5v}, \mathbf{4v}, \mathbf{3c}\}$
Scoop-4	$\text{Type}(\{v_0, v_1\}) = \mathbf{3c}$, $\text{Type}(\{v_2, v_3, v_4, v_5\}) \in \{\mathbf{4c}, \mathbf{5v}, \mathbf{5c}, \mathbf{6v2}, \mathbf{6}*\}$, $\text{Type}(\{v_6, v_7\}) \in \{\mathbf{3v}, \mathbf{4f}, \mathbf{4v}, \mathbf{5v}\}$
Bucket-5	$\text{Type}(\{v_0, v_1, v_2, v_3\}) = \mathbf{3c}$, $\text{Type}(\{v_4, v_5, v_0, v_1\}) \in \{\mathbf{4c}, \mathbf{5c}, \mathbf{6}*, \mathbf{6v2}, \mathbf{5v}\}$

*Bracket-2**: There is an ambiguous case when both and are of type *5-convex*. To resolve this, an additional requirement is enforced: the edge between them should be used by three hexes

In order of precedence, IFGs are placed on unclaimed portions of the skin. These are also pushed onto an insertion queue. Once again, in order of precedence, the top IFG in the queue is popped and then used to place an element. If this fails, the IFG is maintained but marked as *failed* and pushed back onto the queue. Additionally, if an IFG has vertices that fall outside the volume, it is marked as *do_not_insert*. When an insertion is successfully made from an IFG, the skin and vertex types are locally updated and the IFG is deleted. The affected IFGs (*e.g.*, adjacent ones) are deleted and removed from the queue, then local re-planning is performed. One benefit is that a failed insertion (*e.g.*, due to field behavior) may eventually succeed if neighboring insertions succeed and trigger a reevaluation. Also, this scheme naturally prefers uniform front growth. As long as umbilics and highly distorted field regions are avoided, both features lead to avoidance of self-intersection.

Tiling proceeds until no IFGs remaining in the queue can be processed. The goals of avoiding umbilics, achieving boundary conformity, and staying within a ROI use the *do_not_insert* flag to appropriately terminate tiling in particular directions.



Seed — Six hex faces are formed using 24 quarter-bands

Side-1 — Five hex faces are formed using twelve quarter-bands

Bracket-2 — Four hex faces are formed using six quarter-bands

Corner-3 — Three faces are formed using three quarter-bands

Figure 4.12: Element shape definition using quarter-bands

4.3.5 Element Shaping

Consider a single hexahedron: If its faces are aligned with the metric eigenvectors, as should approximately be the case, then the faces are discrete analogues to similarly placed, continuous streamsurfaces. Mesh edges are then analogous to curves of intersections between any two adjacent streamsurfaces, and mesh nodes are analogous to the intersection of any three adjacent streamsurfaces. This is the guiding principle behind the formation of elements in the proposed method. When an element is to be placed, it may use existing skin faces (IFGs) and add new faces. Streamsurfaces are generated to be parallel to the new faces. The intersections of triplets of these streamsurfaces are used to locate newly introduced nodes needed to form a new hex. Figure 4.12 illustrates the collections of streamsurfaces used to define the insertion types: *seed*, *bracket-2*, and *corner-3*. The remaining types shown in Figure 4.11 only use existing nodes and do not require this procedure.

To generate a seed element, half unit-length streamlines are integrated along all six eigenvector sign-direction pairs. From the endpoint of each streamline along direction (σ_i, e_i), four quarter-band surfaces are generated from the eigen-directions normal to the local (σ_i, e_i) direction. Triplets of these quarter-bands intersect at a potential node location of the new seed hex. The intersection process is accelerated using a uniform lattice and Möller's presentation

of the separating axis theorem [AM05] to cull the set of triangle-triangle intersection tests that are performed.

For *Side-1*, *Bracket-2*, and *Corner-3* insertions, the eigen-directions for the quarter-bands are determined by looking at the eigen-directions of the metric tensors at the local skin nodes. The eigen-directions which align with skin edges or normals are selected to form the surfaces.

By design, this element formation method primarily aligns newly inserted skin faces with the metric field. Meeting the size requirement is secondary, and thus is only approximately satisfied. Otherwise the mesh would progressively deviate from local field directions, and local eigen-direction selection could not be used.

4.3.6 Boundary Conformity

Because the proposed method is designed to produce a boundary-aligned field and hence elements that approach boundary-alignment as they reach the surface, a relatively simple boundary capture method suffices for many models.

When elements are inserted in proximity to the boundary, a series of local and greedy snapping moves are attempted to enable capture of feature vertices, feature curves, and surfaces. For a newly inserted hex, nodes are considered for snapping to candidate feature vertices, then edges to candidate feature curves, and finally, faces to boundary surfaces. Non-dimensional fitness scores have been developed for each class to facilitate prioritizing and qualification of moves. To exclude moves that will result in poor quality elements, the shape metric is evaluated for the hexes incident on a moved node. If the minimum metric is below the threshold, then the move is reverted; 0.3 is used in this work. Additionally, to prevent the mesh from spilling out of the volume, the IFGs containing projected faces are appropriately marked.

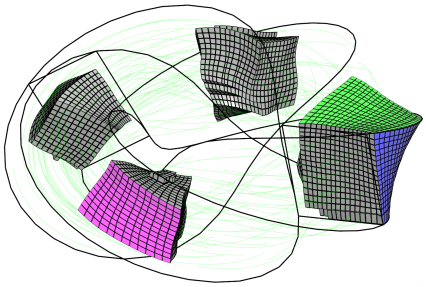
4.3.7 Hex-Dominant Mesh Finalization

The packing approach to hex-dominant mesh conversion from an initial constrained Delaunay tet mesh is only slightly modified for the purpose of this work. The following overview includes the necessary alterations, in bold, for incorporating the hex fronts into the process:

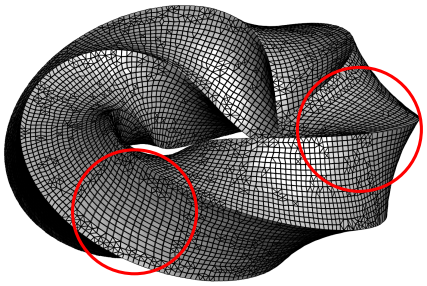
1. **Hex front nodes are packed as fixed bubbles.**
2. The rest of the curves, surfaces, and interior are then packed.
3. **Trimming is performed on regular bubbles that are within one metric unit of the front node bubbles.**
4. **After the surface is remeshed using the surface bubbles, edge swaps are performed to recover hex edges that lie on the boundary.**
5. A constrained Delaunay tet mesh is obtained from the bubble centers.
6. Topological and geometric quality improvement is performed on the tet elements.
7. Tet to hex conversion is attempted and followed by further quality improvement.

4.3.8 Preliminary Results

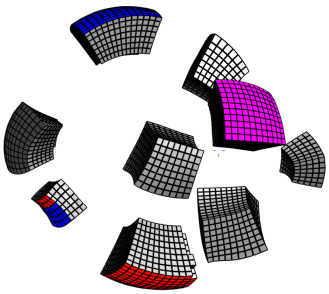
Several example results from this method are shown in Figure 4.13. Each example shows the hexahedral fronts followed by the resulting hex-dominant mesh with the exterior targeted ROIs circled. In all examples, fronts are seeded at interior positions away from umbilics. Colored faces indicate that a similarly colored surface has been captured by the mesh. The data for these models are summarized in Table 4.2. The application of this method to an analysis problem is presented in Section 6.2.1.



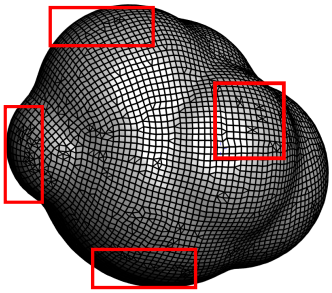
(a) Fronts generated for a $-5/4$ time square torus [Ede]



(b) Torus hex-dominant mesh with ROIs circled



(c) Fronts generated for an amorphous blob



(d) Blob hex-dominant mesh with ROIs circled

Figure 4.13: Preliminary results

Table 4.2: Statistics for additional examples

Model	Square Torus	Blob
Oriented bounding box:	$290.11 \times 294.90 \times 100.00$	$25.80 \times 20.51 \times 20.36$
Specified anisotropy:	4.0 pseudo-isotropic	0.5 pseudo-isotropic
# of tiled hexes (avg. edge len.):	9,811 (4.11)	5,704 (.46)
# of HDOM elements (avg. edge len.):	105,052 (4.47)	127,157 (.52)
% hex by volume (by #):	76.90% (36%)	75.32% (33%)
% tet by volume (by #):	23.10% (63%)	24.68% (66%)
(Min/Avg/Max) Hex Scaled Jacobian:	0.40 / 0.93 / 1.00	0.40 / 0.92 / 1.00
(Min/Avg/Max) Tet Radius ratio:	3.00 / 3.99 / 87.91	3.00 / 3.82 / 129.26

4.4 Global Metric Field Guided Methods in 2D

The local methods have demonstrated the ability to follow desired orientation, but have several limitations:

- Element generation may fail around umbilics.
- Element generation may fail when the field curvature is high with respect to the local sizes.
- Element sizing is influenced by insertion order and field orientation more so than field anisotropy.
- Boundary conformity is difficult (implementation).

To address these issues, we resort to a method that creates an arrangement, or set, of metrics streamlines that span the entire domain. Arrangements of lines, planes, and hyperplanes have been studied extensively in the field of computational geometry [O'R98]. For an arrangement that consists of lines in the plane, the goal is to construct the cells, or polygons, that are bounded by the lines and their intersections. Our approach utilizes this basic concept, but uses curvilinear arrangements instead, as shown in Figure 4.14. This method is similar in

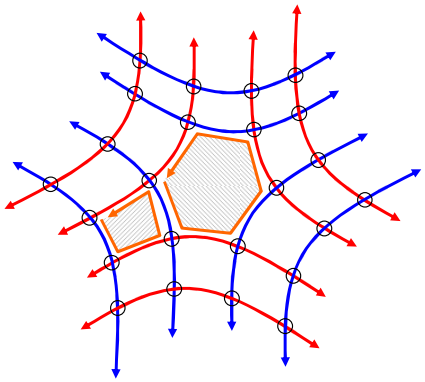


Figure 4.14: Streamline arrangement and element formation

spirit to approaches pioneered by Alliez et al. [ACSD+03] for computer graphics, but is

designed to handle planar domains with boundary features and produce highly anisotropic meshes. Furthermore, our method can operate on any arrangement of curves.

Both anisotropy and orientation can be sufficiently captured by the mesh resulting from a well conditioned arrangement. Furthermore, the overall method can be more efficient than the local one because it avoids the unnecessary recreation of small streamline portions.

The following sections begin by describing the procedure to generate a streamline arrangement — this is called curve placement. It is likely to have defects in the form of degenerate intersections and deviation from desired anisotropy. Therefore, cleanup, insertion, removal, and adjustment of the curves are performed inline with curve placement. Finally, we briefly discuss the conversion of the arrangement to a set of polygonal elements that can be made quad-dominant. The flowchart in Figure 4.15 shows the flow of data through the processes described in this section and in Chapter 5. The results of this method are shown in Chapter 1 as well as Section 6.2.2.

4.4.1 Curve Placement

The curve arrangement is populated by inserting new curves into an initial set. The initial set itself may be populated from final arrangement of the previous adaptation iteration. Alternatively, we provide a procedure to generate a "skeleton" arrangement that sandwiches umbilics (mimics Figure 4.5). It also uses the centers of a coarse packing of bubbles as seed locations. Streamlines in both eigen-directions are integrated at several seed locations so that the entire domain is covered. Optionally, boundary features are captured by inserting boundary curves and offset curves (for interior boundary curves only) into the arrangement. Upon termination of Algorithm 3, the domain is sufficiently covered with an arrangement of curves that capture the boundaries, umbilics, and primary and secondary metric directions. However, the spacing between curves will most likely have to be corrected by insertion, removal, and trimming. To correct low curve densities, short edges are located in the inter-

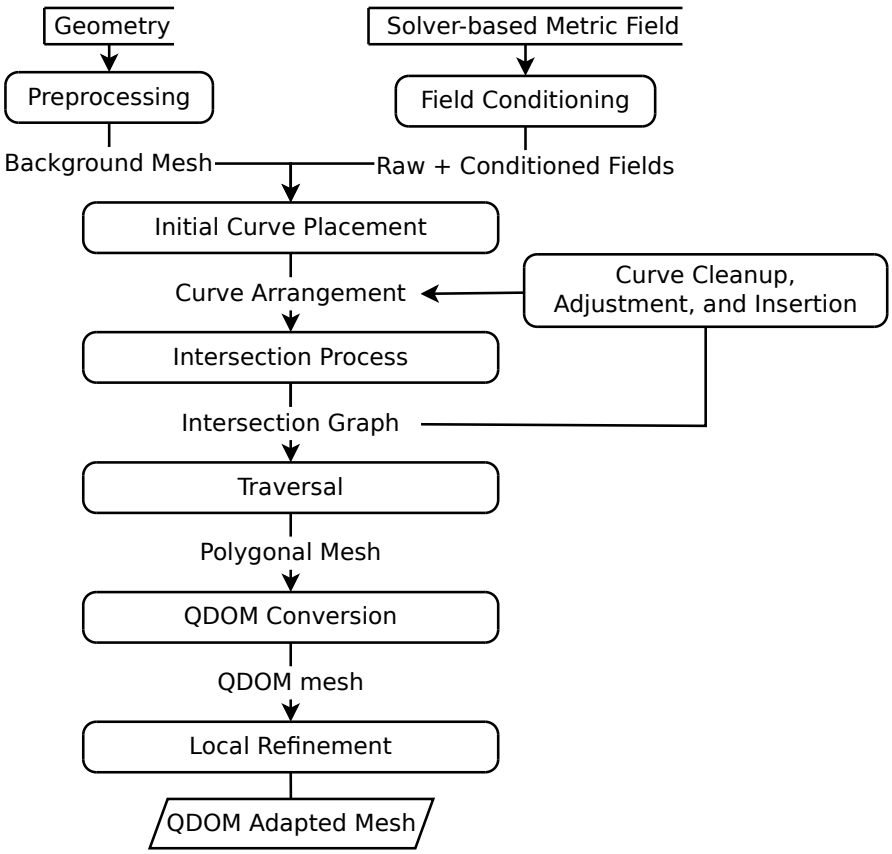


Figure 4.15: Process flow for global 2D method when applied to adaptation problems

section graph. Each short edge corresponds to a curve portion, and the midpoint of each curve portion can be used to integrate a new streamline curve in the orthogonal metric direction. This is similar to the visualization technique in [RPP+09], except that a search is performed to bisect portions by their metric-length. The other cases of high curve densities and degenerate configurations are addressed in the next section.

4.4.2 Curve Adjustment and Cleanup

The proposed method is designed to operate on any arrangement of curves which may be traced from the current metric field, representative of boundary features, edited, or imported from the previous remeshing iteration. Various degenerate cases may be introduced over the

Algorithm 3 Generating an initial curve arrangement

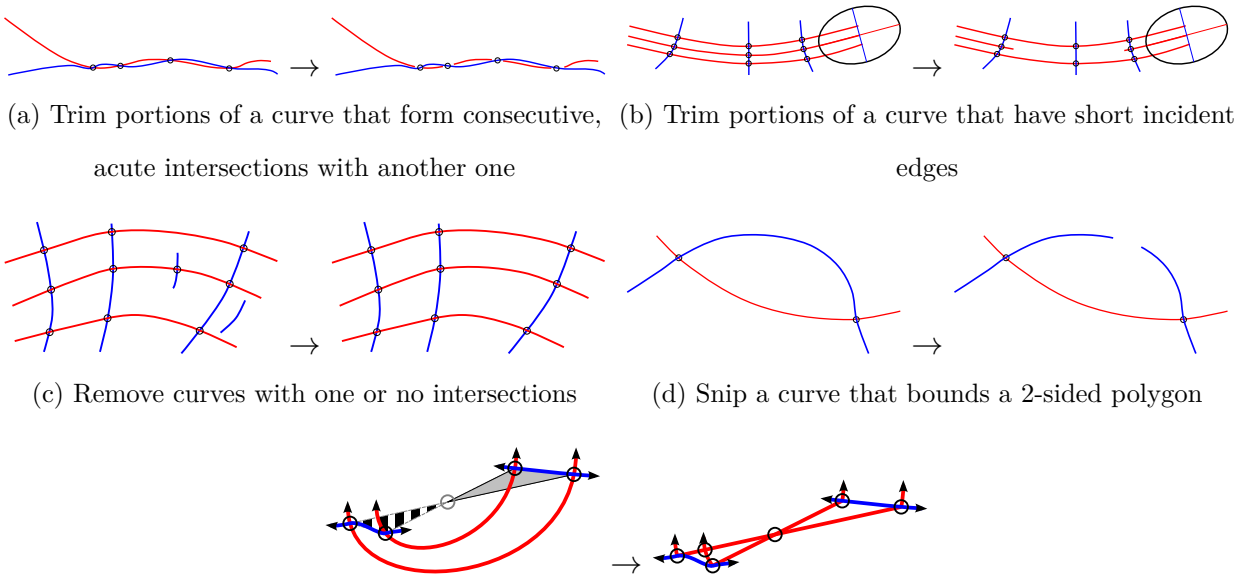
Require: Geometry, background mesh, and metric field that captures appropriate orientation
  Extract all umbilics of the current orientation metric field
  for Each umbilic point **do**
    Extract the local separatrix directions
    for Each separatrix **do**
      March a {positive, negative} distance ϵ along the current separatrix direction
      Integrate a curve with the standard termination criteria or of fixed length along the orthogonal eigenvector direction at that point
    end for
  end for
  Generate a non-uniform, isotropic metric field from the current metric field that represents the sizing
  Generate an isotropic, triangular packed mesh from the field
  Prioritize the bubbles by sorting in ascending order of size
  while All bubbles are not marked **do**
    if Current bubble is marked **then**
      Skip bubble
    end if
    Seed a curve from the current bubble
    if All bubbles that curve passes through are not marked **then**
      Accept curve and mark all traversed bubbles
    end if
  end while
  Add all domain boundary curves and boundary capture curves to the arrangement
  Mark boundary-generated curves as fixed to prevent modification

course of these actions, especially insertion of new curves as described in Section 4.4.1. For instance, a newly traced curve may "tangle" with an existing curve in the arrangement because of numerical integration error. Similarly, there are several other cases which may be necessary to catch to ensure valid results downstream in the pipeline and target metric conformity:

- Removal of intersections that form acute angles within some threshold: this usually occurs when non-traced curves are introduced or because of numerical integration. May cause mesh elements with acute angles.

- Curves that intersect only once with the arrangement: these curves usually don't contribute to closing a polygon and increase fragmentation as seen by a lower quadrilateral to triangle ratio.
- Snipping curves that form 2-sided polygons: this is a precondition for polygon formation.
- Trimming and removing curves that create columns of elements that are thinner than some factor of the locally requested sizes.
- Insertion of curves to refine edges that are longer than some factor of the locally requested size.
- Decimation of curves that preserves intersection points to remove discrepancy between curved and linear representations: this is necessary to prevent certain types of degenerate element formation (*i.e.*, self-intersection and inversion) that arise from using linear rather than curved edges in the polygonization.

As indicated in the list above, the three actions taken to resolve these cases are curve removal, curve trimming (removal of a portion), and curve insertion. Figure 4.16 shows various degenerate curve cases, the heuristics used to detect them, and the resulting action.



(a) Trim portions of a curve that form consecutive, acute intersections with another one

(b) Trim portions of a curve that have short incident edges

(c) Remove curves with one or no intersections

(d) Snip a curve that bounds a 2-sided polygon

(e) Decimate curves that produce invalid elements, while preserving existing intersections

Figure 4.16: Resolution of degenerate curve cases

The above cases are detected and resolved with a certain prioritization while simultaneously updating the curve arrangement and intersection as indicated in the pipeline loop of Figure 4.15. The exception is curve insertion, which may be performed over a fixed number of global passes, rather than execution until the sizing criteria are exactly satisfied. Decimation is another example of an operation that may introduce new degeneracies. It is typically applied after the others and then the entire process is repeated until all cases have been resolved. Any remaining size discrepancies at the end of the operations are entrusted to local mesh refinement. That is discussed in Chapter 5.

4.4.3 Element Generation

The sample arrangement in Figure 4.14 contains quadrilateral and non-quadrilateral elements, which is representative of the general case. Another important property of the streamline arrangements is that curves intersect at approximately 90° angles because the

eigenvectors are orthogonal but the streamlines are numerically integrated. Then, element generation has three stages: intersection, traversal, and conversion to a quad-dominant mesh.

In the intersection stage, each curve in the arrangement is logically broken up into chunks contained in a background mesh triangle. Then standard intersection testing only has to be performed against chunks of different curves that are present in the same background mesh triangle. Identified intersections are filtered to avoid duplicates and then converted to mesh nodes.

Next is traversal, which begins at an intersection and chooses one of four "quadrants" bounded by its incident streamlines. This is followed by marching from intersection to intersection along streamlines, such that the initial quadrant is always to the left of the traversal path. This yields a positively oriented loop of intersections, the nodes of which constitute a generate element.

If the element is a quadrilateral or a triangle, then the method is done. Otherwise, special templates can be applied to convert general polygons to a quad-dominant patch. For instance, a hexagon can be ideally split based on quality concerns (*e.g.*, a non-convex hexagon may not admit certain splits) and the local orientation requirements. All other cases are passed through a quad-dominant, field-aware recovery process similar to that described in [YS03b].

Chapter 5

Mesh Adaptation

Solution adaptation is an iterative process in which each adaptation iteration consists of several solver-side and mesher-side steps that are shown in Figure 5.1. This chapter outlines the procedures that are applied at the adaptation iteration level and at the meshing level within a particular iteration to achieve the goals stated in the previous sections.

5.1 Mesh Adaptation Loop

The proposed method takes as input an initial geometry and optionally, an input mesh. If an initial mesh is not provided, a geometry- and user input-based metric field can be constructed and this method applied to generate it. Each subsequent remeshing iteration requires the geometry, current metric field from the solver, and data from the previous mesh iteration. A simplified flowchart for the solution adaptation process is shown in Figure 5.1.

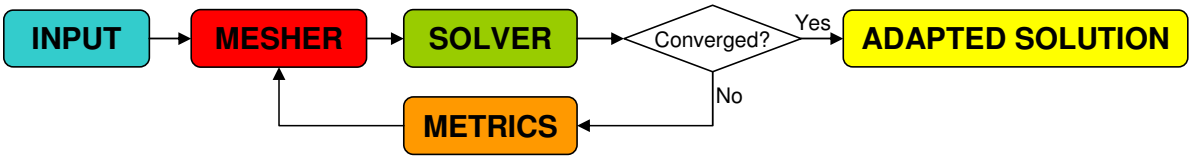


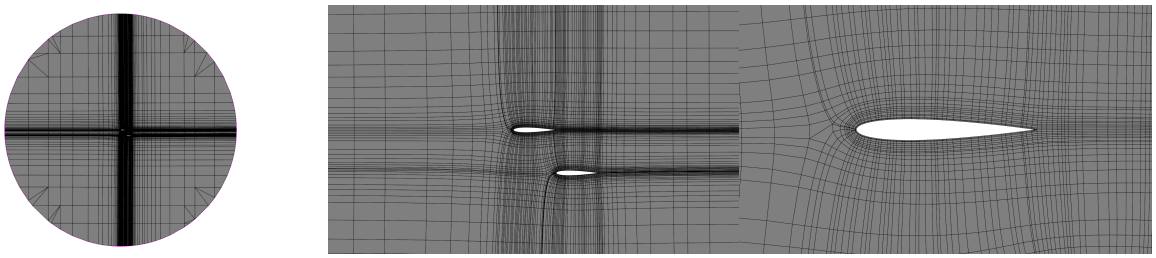
Figure 5.1: Overall solution and mesh adaptation loop

5.2 Per-Iteration Meshing Strategy

Within a particular adaptation iteration, the mesh adaptation process block in Figure 5.1 can be expanded into the following steps:

1. Metric-Field Conditioning: Techniques for regularization by noise removal and fairing of orientation are employed to produce variations of the raw metric field from the solver. Some of these fields are utilized to independently guide orientation and anisotropy requirements.
2. Region-of-Interest Extraction: Critical portions of the domain are extracted based on solution and solution-based metric information. The high aspect-ratio technique is then restricted to these "subdomains."
3. Curve-Arrangement Creation: Metric-streamlines are integrated using the metric field and placed (spaced) according to the field in orthogonal directions to form a net. Curve-pair intersections result in a net, the curve intersection graph, which can be traversed to form a polygonal mesh. After a suitable conversion operation, this yields a quad-dominant mesh.
4. Curve-Arrangement Conditioning: Curves are trimmed, removed, and inserted to remove degenerate intersection cases and better respect anisotropy requirements.
5. Complementary-Region Remeshing: The portion of the original domain not covered by the subdomain mesh is extracted via a Boolean operation. The original metric field is adjusted to blend with the subdomain mesh boundaries, and the complementary region is meshed using BubbleMesh.
6. Final Mesh Assembly: The subdomain and complementary region meshes are united and exported for the next iteration.

Metric fields were previously treated in Chapter 3 and Section 4.4 detailed the creation



(a) View of entire mesh (b) Zoomed-in view of both airfoil elements (c) Zoomed-in view of mesh around a single airfoil

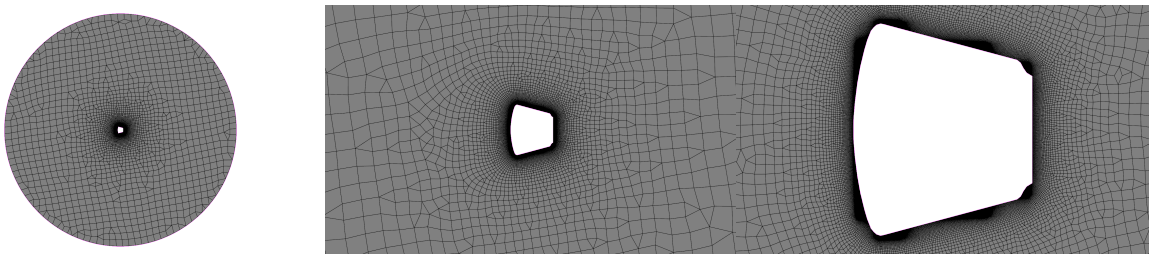
Figure 5.2: Initial mesh generation for double 0012 configuration using boundary-aligned metric field and global 2D method for mesh generation

and conditioning of curve arrangements. We first present some tools for initial mesh generation to kick-off an adaptation loop, and in the following sections, we discuss how to characterize regions of interest in the domain. These regions can be used to partition the domain in such a way that the proposed methods can mesh critical regions and are complemented by existing, high-quality methods in other regions. Furthermore, the global 2D method is supplemented with a set of local mesh refinement templates to satisfy high aspect-ratio requirements.

5.2.1 Initial Mesh Generation

A starting mesh is required to initialize an adaptation loop. In this section, initial meshes are generated using the metric field generation techniques previously described in Chapter 3 as well as mesh generation via curve arrangements and bubble-packing. Figure 5.2 shows an initial mesh for a double NACA 0012 configuration using the global 2D method for mesh generation. Supplementary curves, offset from the viscous boundary, are automatically inserted for some degree of boundary layer capture. Polygonal elements may occur, especially at farfield boundaries, so they are automatically decomposed into triangular elements.

In the next example (Figure 5.3), an initial metric field is generated by considering an angle-of-attack α to guide orientation in the farfield. A fixed anisotropy is prescribed along



(a) View of entire mesh (b) Zoomed-in view of capsule (c) Further zoomed-in view of capsule

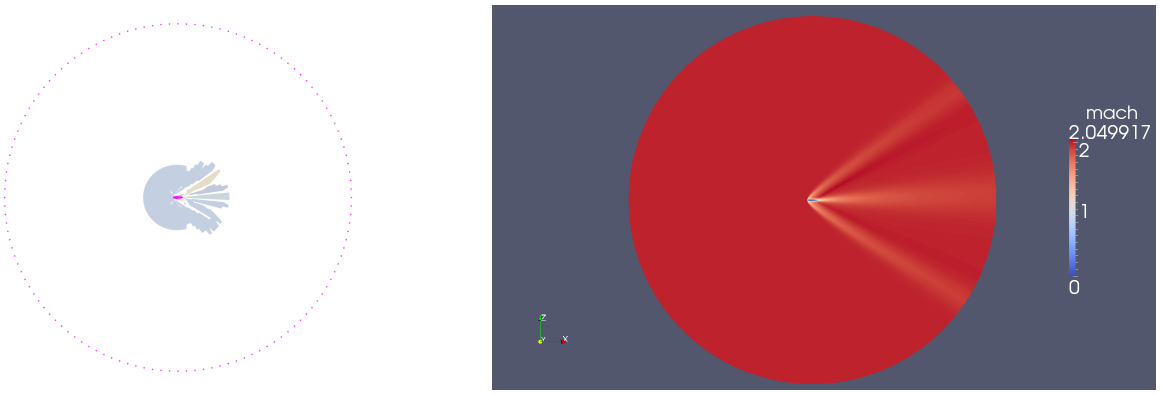
Figure 5.3: Initial mesh generation for 2D capsule using boundary-aligned metric field, anisotropy on viscous surfaces, and an angle-of-attack. Mesh generation is performed using BubbleMesh®.

the viscous boundary of the 2D capsule, which transitions to isotropy in the farfield. Finally, a quad-dominant mesh is produced through bubble-packing.

5.2.2 Region-of-Interest Extraction

Physically based relaxation methods such as BubbleMesh [YS03b] are capable of capturing spatially varying metric fields. However, it can be difficult to enforce a non-staggered, high aspect-ratio, nodal distribution which favors layered quadrilateral elements in critical regions such as around shocks, boundary layers, and wakes for CFD problems. In other regions, however, it possible to create a high quality quad-dominant mesh. This is addressed by placing curve arrangements in the critical regions, or "subdomains," and packing rectangular solid cells in the complementary ones. The challenge is determining the critical regions based on metric and/or original solution information.

Aspect-ratio (AR) thresholding is an example of a simple subdomain criterion. In Figure 5.4a, background mesh triangles that have at least one nodal metric with an aspect-ratio greater than 5 are shaded. Note that aspect-ratio is defined here as $h_{\max}/h_{\min}$ and is always greater than or equal to 1. The corresponding Mach number plot in Figure 5.4b suggests that while the criterion identifies parts of the shock, boundary layer, and wake near the

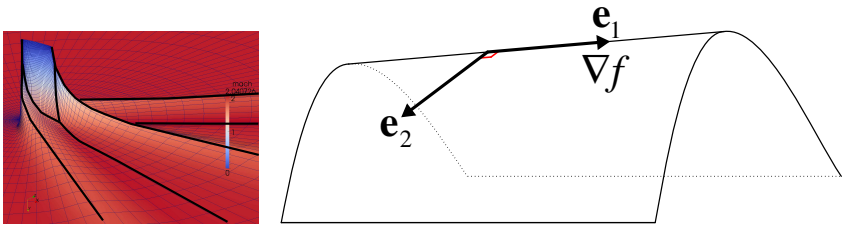


(a) Subdomains (b) Solution Mach number plot

Figure 5.4: Subdomains based on aspect-ratio threshold

airfoil, those features are not captured further out in the farfield.

Recall that the metric orientation and aspect-ratio are determined from the Hessian of a scalar. Mach has been selected in this case. Previous research has shown that the aspect-ratio is large when the magnitude of one principal curvature of the Mach surface is significantly greater than the other. This can be verified in Figure 5.5a, where the bow shock and wake begin as sharp features with a high ratio of principal curvature magnitudes and begin to flatten out as one moves into the farfield. With the exception of the boundary layer,



(a) Mach carpet plot (b) Creases of a Monge patch

Figure 5.5: The significance of creases to subdomains

such features form creases [EGM+94]. Extracting creases — that is, ridges and valleys of a scalar function — would be more appropriate to capture complete structures in the solution field. Note that the exact topology of the creases is not desired, but rather an inflated region around them in which the techniques of Section 4.4.1 can be applied.

For a function defined over a 2D domain, ridges and valleys consist of the locus of maxima and minima, respectively, of certain 1D slices of the function. These slices are taken along Hessian eigenvectors, or principal curvature directions, that correspond to the largest-magnitude directional derivative. The cases of ridges and valleys are illustrated in Figure 5.5b. For creases according to the height definition, which are described in [EGM+94], the conditions are as follows for a scalar function $f_{\mathcal{H}}$:

$$\lambda_1 < 0 \quad \text{and} \quad \mathbf{e}_1 \cdot \nabla_{\mathbf{x}} f_{\mathcal{H}} = 0\,, \text{ or} \qquad \text{(Ridges)} \tag{5.1}$$

$$\lambda_2 > 0 \quad \text{and} \quad \mathbf{e}_2 \cdot \nabla_{\mathbf{x}} f_{\mathcal{H}} = 0. \qquad \text{(Valleys)} \tag{5.2}$$

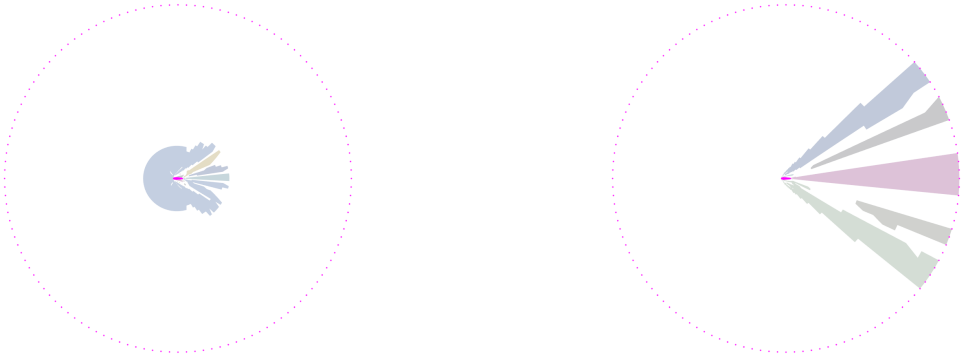
In the above expressions, $\lambda_{1,2}$ and $\mathbf{e}_{1,2}$ are the eigenvalues and eigenvectors of the Hessian of a scalar function $f_{\mathcal{H}}$, respectively. Only rough polygonal areas containing creases are desired, so the condition is relaxed to:

$$\left|\mathbf{e}_i \cdot \hat{\nabla}_{\boldsymbol{x}} f_{\mathcal{H}}\right| \leq \cos\left(\theta_{\min}\right). \tag{5.3}$$

The original orthogonality condition is replaced with a minimum unsigned angle $\theta_{\min}$ that must be maintained between the normalized gradient vector, $\hat{\nabla}_{\boldsymbol{x}} f_{\mathcal{H}}$, and the appropriate eigenvector $\mathbf{e}_i$. Local gradients and Hessian matrices are approximated using local quadratic fits on the vertex solution data so that background mesh vertices can be marked if they satisfy Equation (5.3). Subdomains are formed from polygons that are incident on marked vertices.

Boundary-layer coverage is enforced by explicitly adding the set of triangles within a specified distance from the internal boundaries to the subdomain set.

Subdomains are then subjected to the following operations that mimic their image processing counterparts:



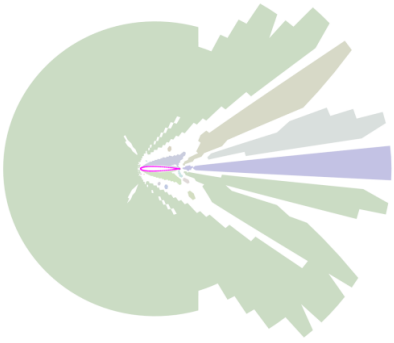
(a) AR subdomains (b) Ridge-based subdomains

Figure 5.6: Comparing aspect-ratio subdomains with ridge-based subdomains

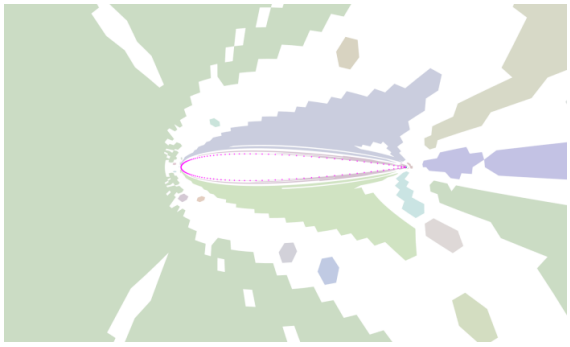
1. Dilation by distance or number of layers
2. Erosion by distance or number of layers
3. Hole-filling

Image closing [HSZ87] consists of a dilation step followed by an erosion step with specified distances. These distances are provided as percentages that are applied to a characteristic length derived from the model. The result is consolidation of subdomain islands that arise from the various approximations and richness of features in the solution field. Subdomains that share at least one edge are automatically merged during the various operations. Similarly, the formation of new subdomains due to erosion is also tracked. Finally, hole-filling closes up any interior voids in the subdomains.

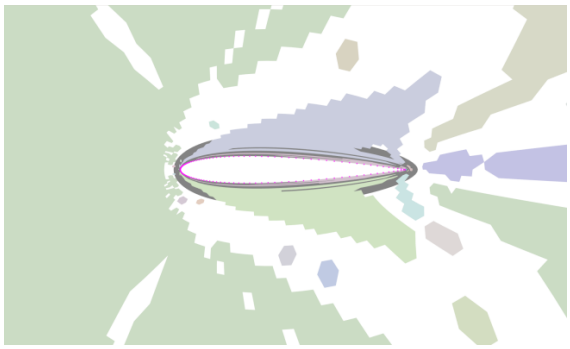
Many of these subdomain operations are illustrated in Figure 5.7 along with streamline and mesh generation (followed by refinement) that is restricted to the subdomains — the means by which potentially high aspect-ratio elements can be generated in regions of interest.



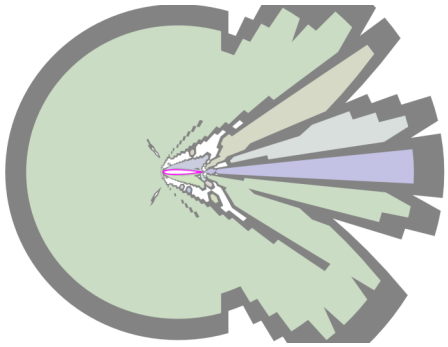
(a) Aspect-ratio thresholded subdomains (AR >= 5.0). Disjoint subdomains have unique colors.



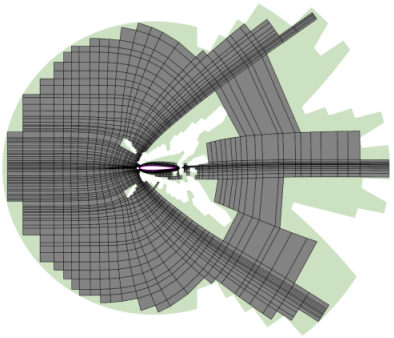
(b) AR subdomains (zoomed in)



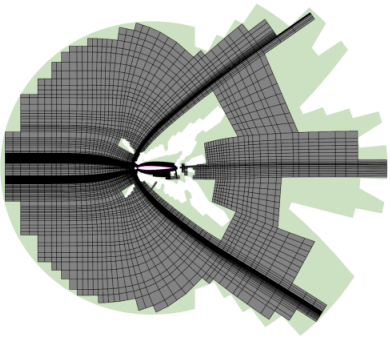
(c) Union of AR subdomains with boundary-layer subdomain. New coverage regions are gray.



(d) Dilated subdomains. Dilated regions are gray.

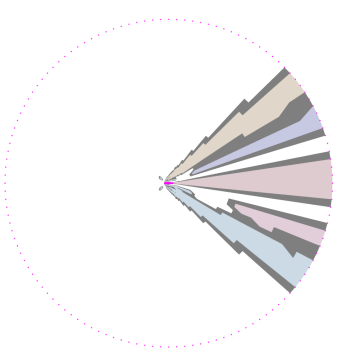


(g) Mesh elements formed from restricted streamlines

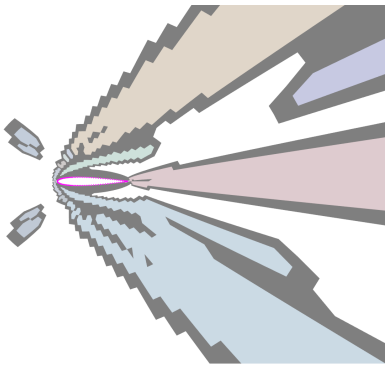


(h) Mesh after non-local (propagating) refinement

Figure 5.7: Subdomain operations and subdomain-restricted mesh generation



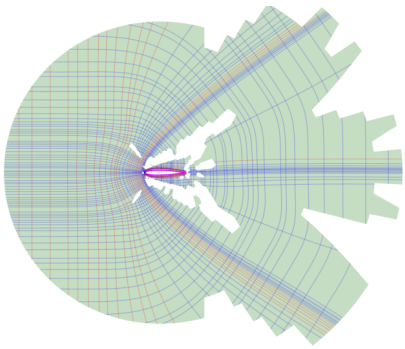
(a) View of entire domain



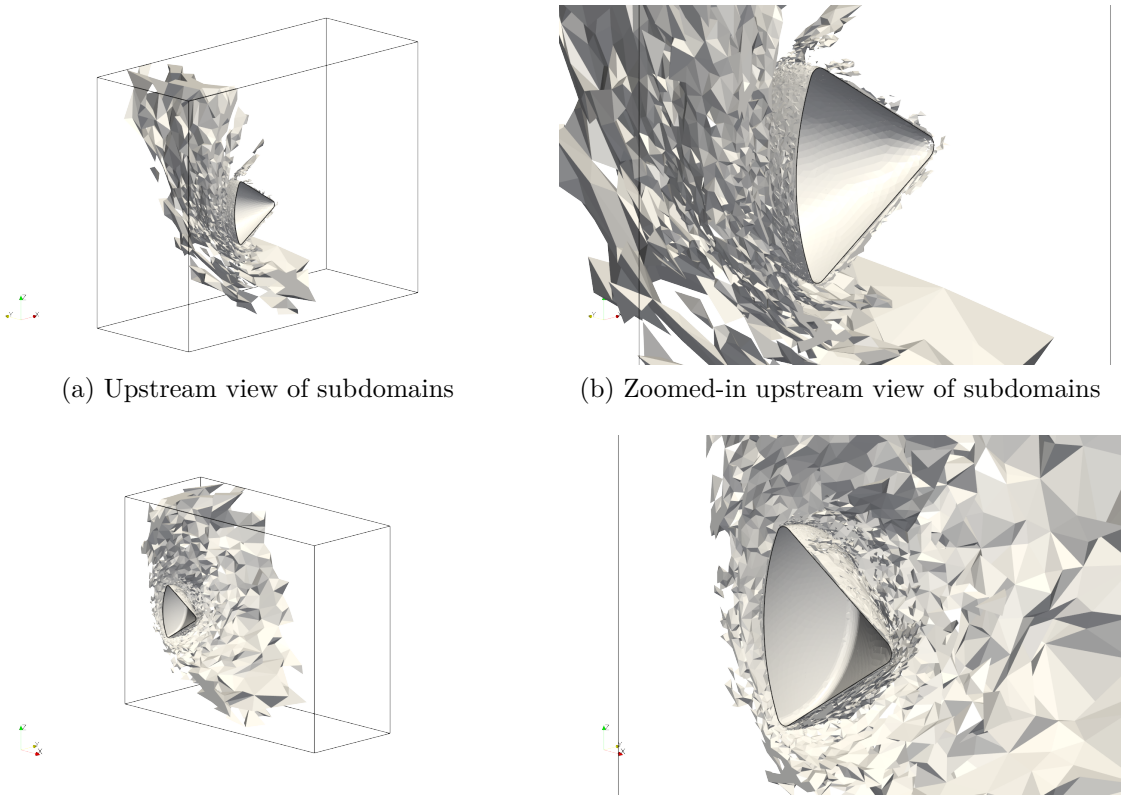
(b) Zoomed-in view of subdomains

Figure 5.8: Subdomain extraction and dilation based on Mach creases

(e) Subdomains after closing (dilation followed by erosion)



(f) Streamlines restricted to subdomains



(a) Upstream view of subdomains

(b) Zoomed-in upstream view of subdomains

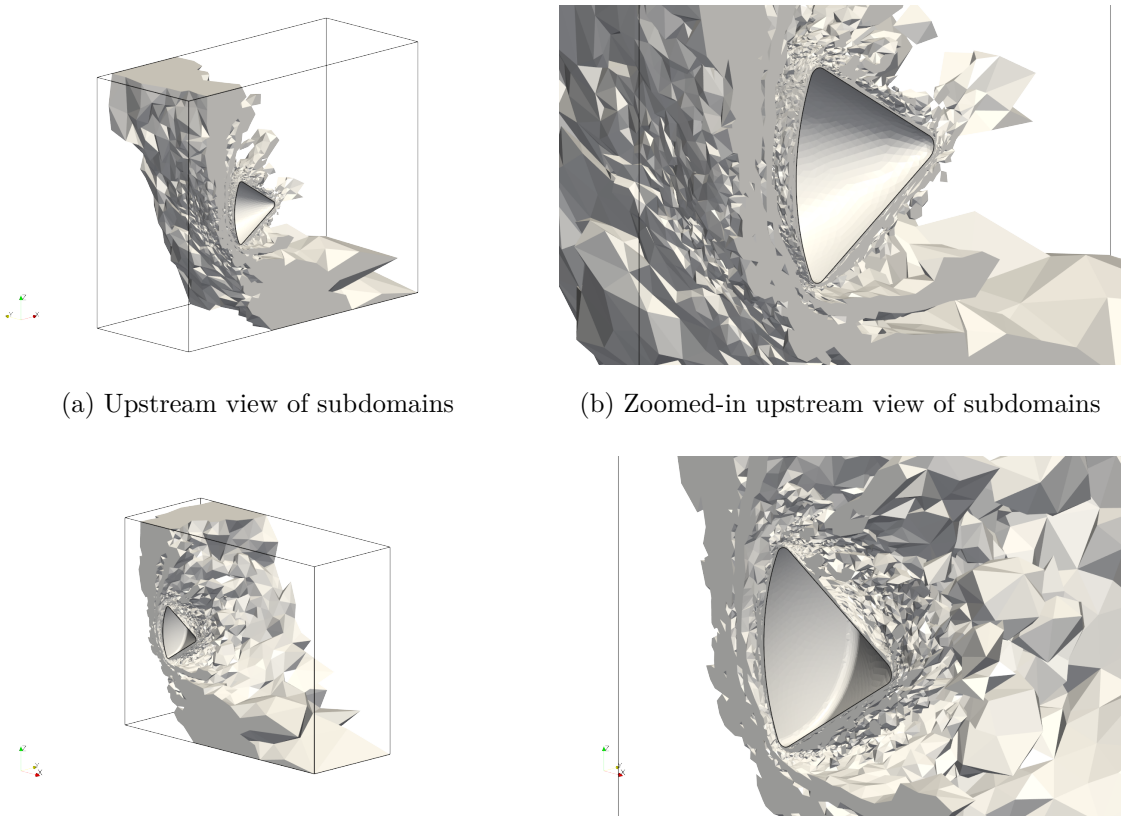
(c) Downstream view of subdomains

(d) Zoomed-in downstream view of subdomains

Figure 5.9: Defining subdomains for the 3D capsule problem by thresholding FA

The net result is a set of marked background mesh triangles that allow efficient determination of critical regions. The curve placement algorithms described so far are then restricted to subdomain triangles.

Similar measures can be taken in 3D. In Figure 5.9, we consider using a threshold criterion on fractional anisotropy (FA). Figure 5.10 shows the effect of applying dilation in 3D.



(a) Upstream view of subdomains

(b) Zoomed-in upstream view of subdomains

(c) Downstream view of subdomains

(d) Zoomed-in downstream view of subdomains

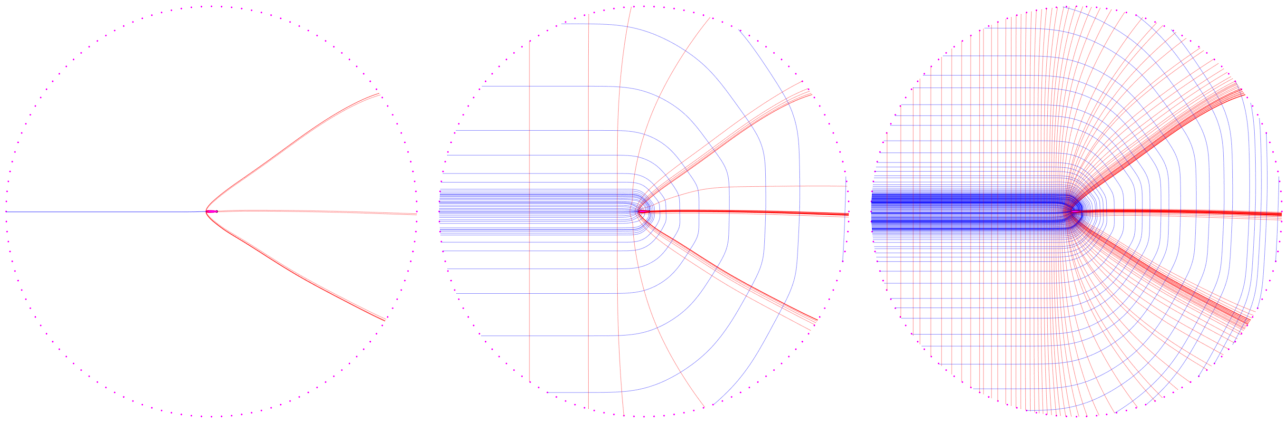
Figure 5.10: Dilation of FA subdomains for 3D capsule problem

5.2.3 Streamline Cleanup and Processing for Anisotropic Gradation

The mechanics of streamline placement and processing were sufficiently covered in Section 4.4.1 and Section 4.4.2; we demonstrate their results for the 0012 problem in Figure 5.11 and Figure 5.12, respectively.

5.2.4 Local Mesh Refinement

Any general polygons present in the mesh are triangulated and quads are recovered using a process similar to that in [VSI00]. The last measure taken to rectify the sizing and anisotropy



(a) Initial global curve arrangement (b) Intermediate global curve arrangement (c) Final global curve arrangement

Figure 5.11: Incremental streamline placement for 0012 airfoil problem

of the traced mesh is a set of local refinement passes on the mesh. Edges that are shorter than $1/\sqrt{2}$ according to the local metrics are subdivided and the 2-refinement-style templates in Figure 5.13 are applied based on the configuration of subdivided edges. Once again, quads are recovered from general polygons introduced in this step. Quads with three edges marked for subdivision are upgraded to uniform refinement. The result of local mesh refinement is compared to that of global refinement for the 0012 case in Figure 5.14.

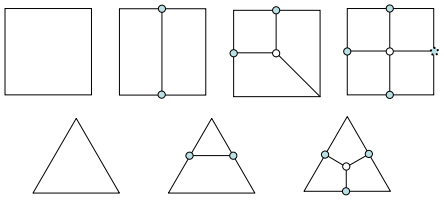


Figure 5.13: Local refinement patterns that operate on quad-dominant meshes

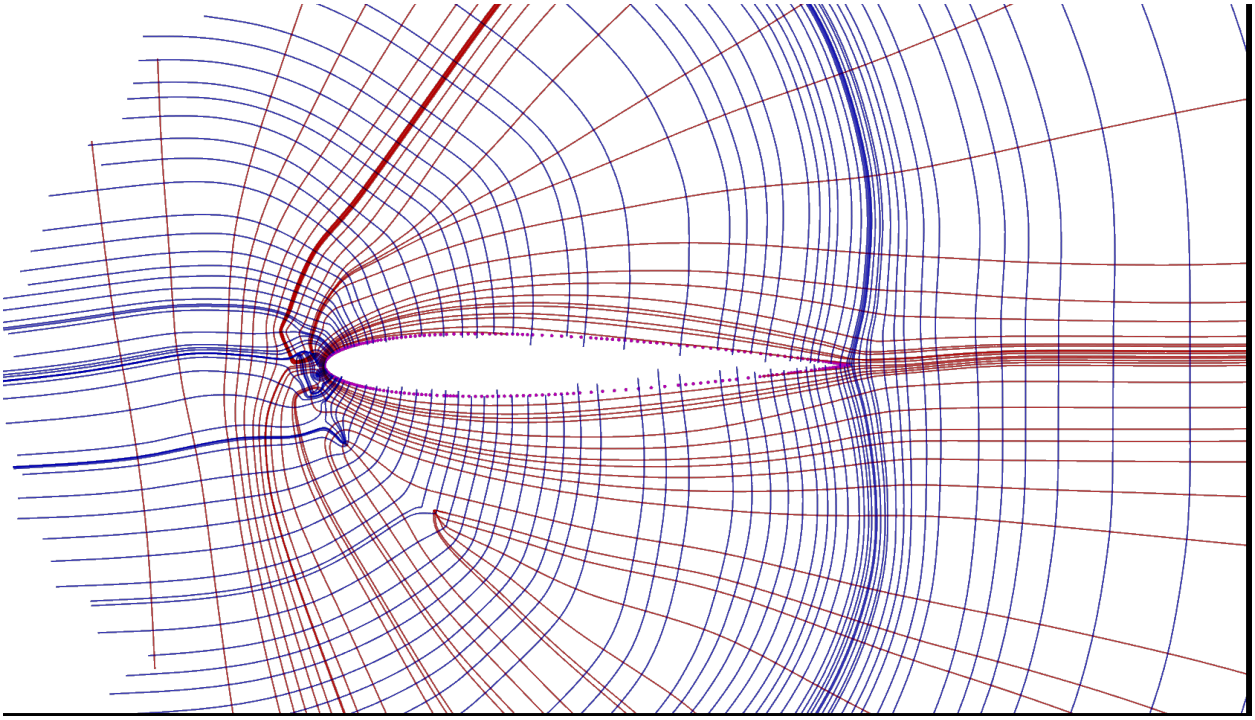
5.2.5 Complementary-Region Meshing

The traced mesh is subtracted from the original planar geometry to produce the complementary region. Furthermore, meshing constraints and the conditioned metric field are passed on for remeshing via bubble-packing (Figure 1.5e). However, it is possible that the traced

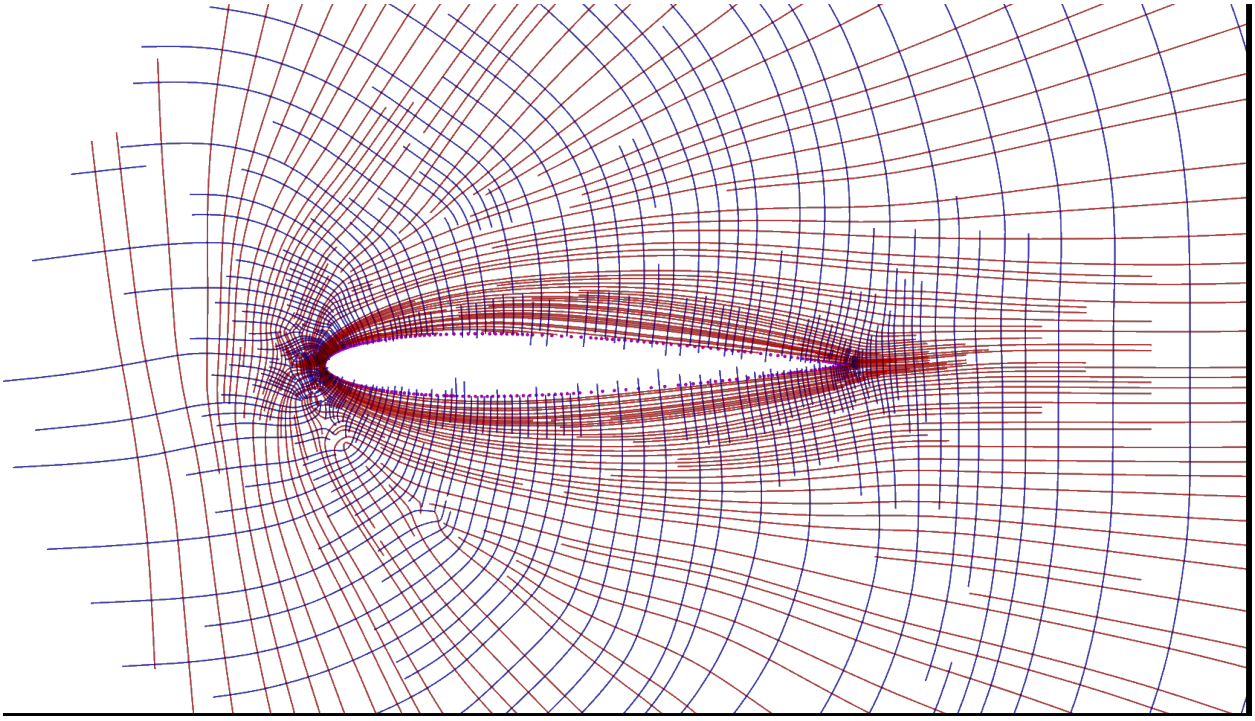
mesh does not entirely conform to the local metrics at the interface between it and the complementary region. Thus, the conditioned metric field is adjusted so that the packed mesh transitions to the rigid, traced mesh. This is achieved by blending only the metrics that are within a specified distance from the traced-packed interface, while holding all other metrics fixed. Blending simply uses a familiar Laplacian smoothing scheme on the components of the metrics in linear-size form.

5.2.6 Adaptation Control

Many factors can cause the adaptation process to stall or diverge, especially when complete remeshing is involved rather than incremental local modification. One factor is the metric field, which may make excessive requests during an intermediate adaptation iteration. The present method considers filtering of metric data over iterations to facilitate stable, adaptive remeshing. This is done prior to use by any of the previously defined operations. A simple scheme that is demonstrated in Section 6.2.2 resamples the previous field onto the current background mesh and blends it with the new metric field. Previous metrics are not available during the 0^{th} iteration, so the implied metrics of the initial mesh are used instead. Then, a normalized weight that is usually $\leq .5$ is assigned to the new field and the blending is performed by log-Euclidean interpolation at each node. Figure 5.15 shows the effect of blending using metric ellipses for the 0012 problem.

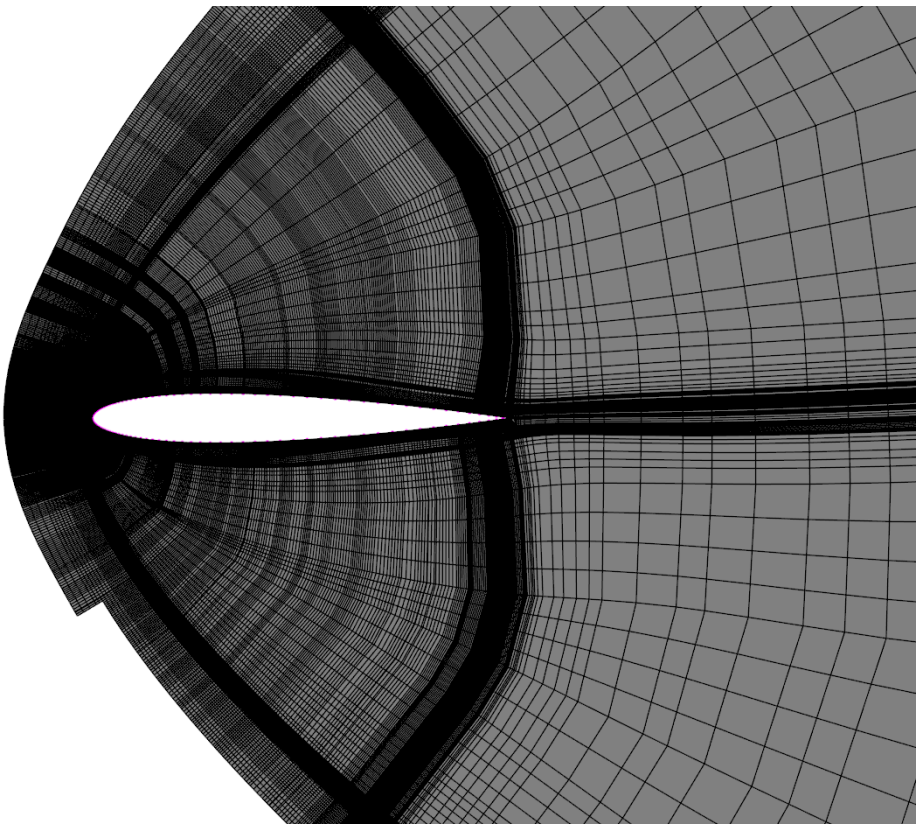


(a) Global streamline arrangement

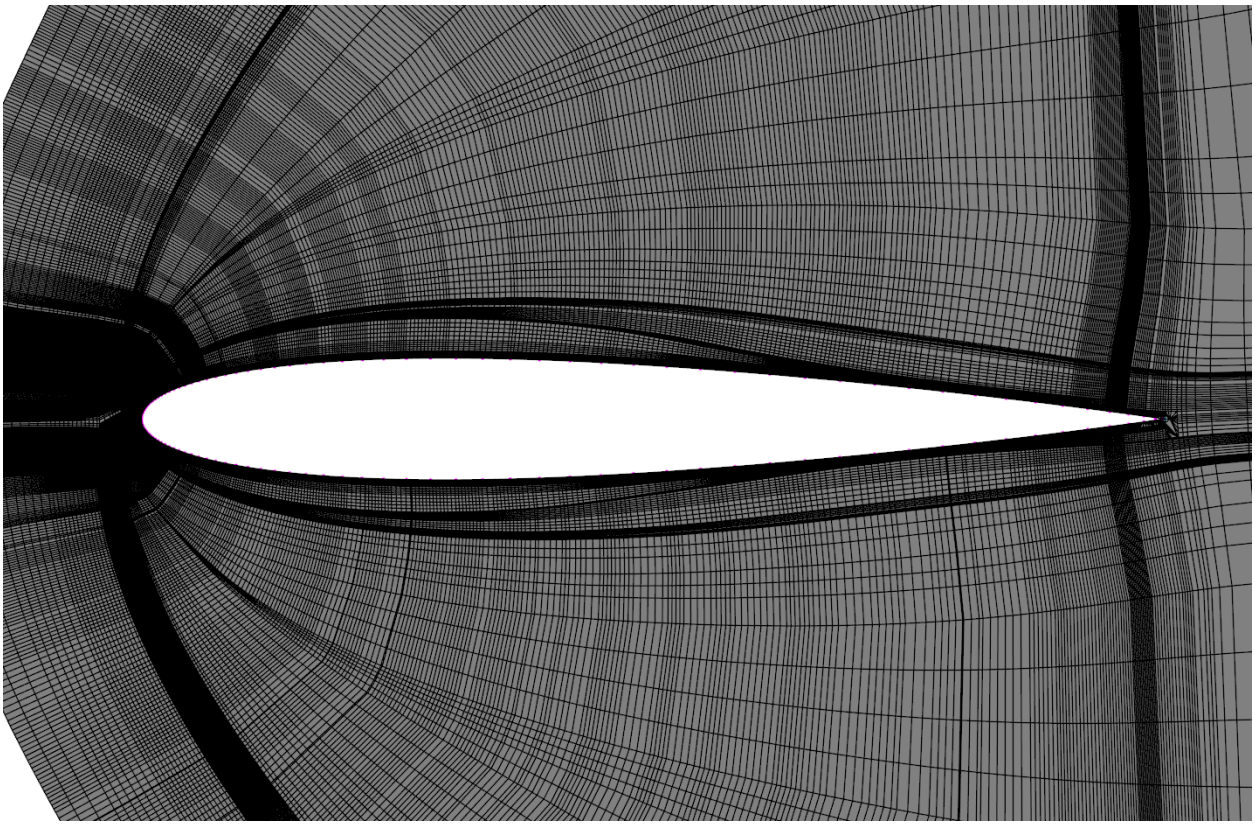


(b) Arrangement after cleanup and trimming operations

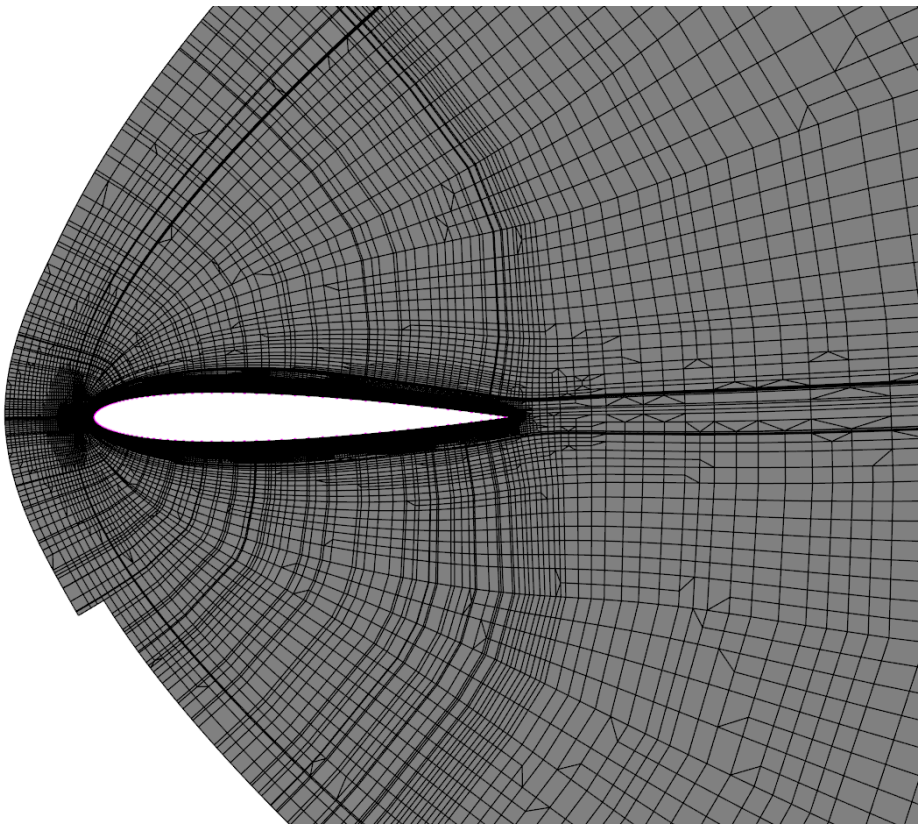
Figure 5.12: Streamline trimming to achieve better metric conformity



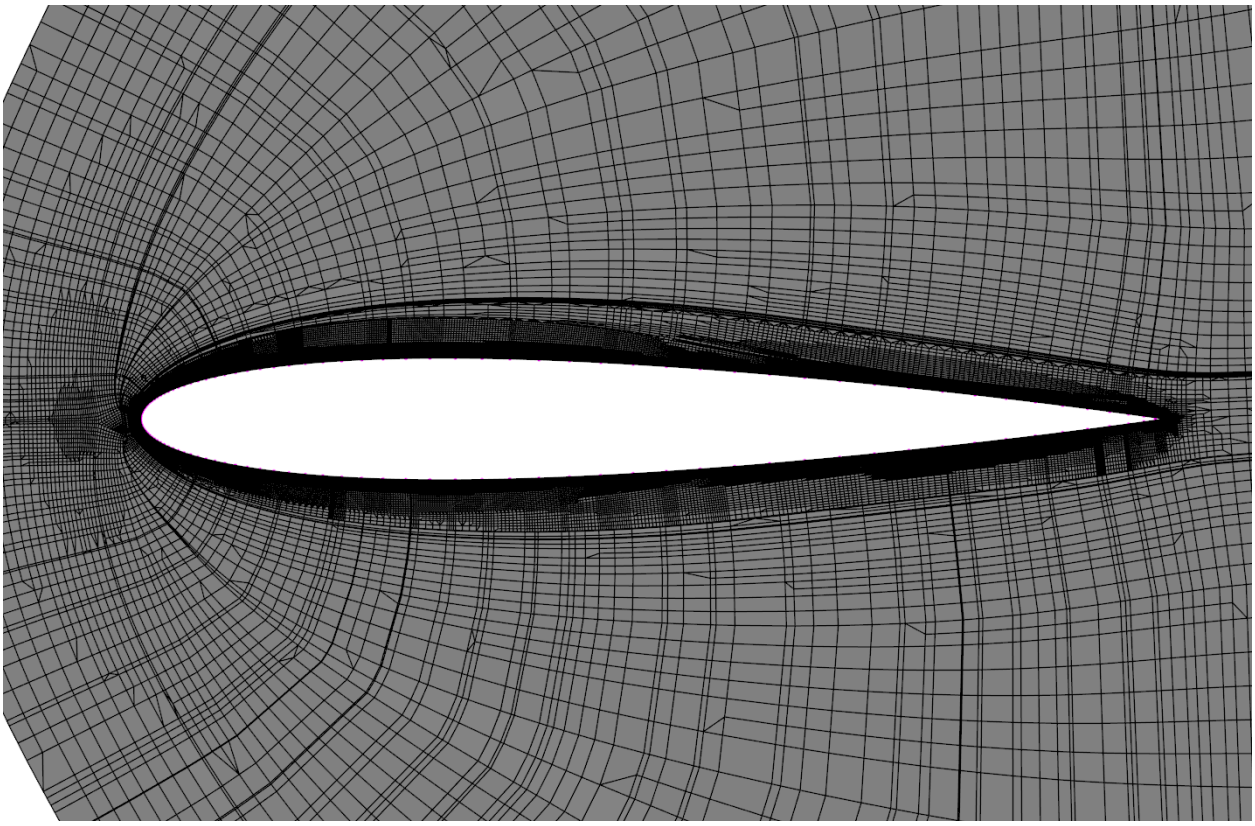
(a) Globally-refined mesh



(b) Zoomed-in view of globally-refined mesh

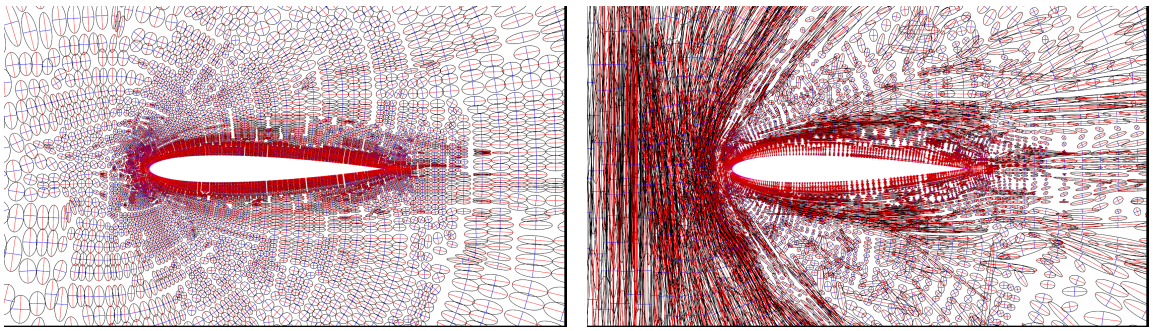


(c) Local refinement using templates



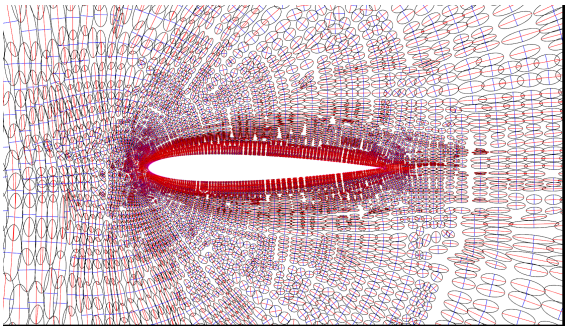
(d) Zoomed-in view of local refinement using templates

Figure 5.14: Comparison of global refinement with application of local 2-refinement templates



(a) Metrics from previous iteration

(b) Raw metrics from current iteration



(c) Blended metrics to use while preparing mesh for next iteration

Figure 5.15: Blending metrics from the previous iteration with those of the current one

Chapter 6

Design, Analysis, and Manufacturing Applications

6.1 Graphic and Product Design

Graphic design sometimes requires repeating patterns. In these applications, we exploit pseudo-isotropic and anisotropic field generation to place unstructured patterns that can exhibit varying orientation, nominal size, and/or anisotropy. Geometry and user input are factored into the process as boundary conditions and in other ways. BubbleMesh® is used to place field-guided primitives that can be fed into a vector graphics pipeline.

In the first application, we look at generating metric fields based on sketch input. First, a 2D background mesh with suitable padding is generated to cover a square region, as seen in Figure 6.1. A user can sketch a path, which is subsequently filtered (e.g., Savitzky-Golay filtering) and then inserted into the background mesh (as seen in Figure 6.2). Boundary conditions are set along sketch curves using a base anisotropy and by approximating curve tangents for orientation. After a field is generated (Figure 6.3), the nodal metrics are optionally post-processed by adjusting their anisotropy as a function of distance from the sketched

curve. This allows for a transition to isotropy away from the sketch inputs and/or boundaries. Finally, bubble-packing and mesh generation provide a visualization of the end-results: see Figures 6.4, 6.5, and 6.6.

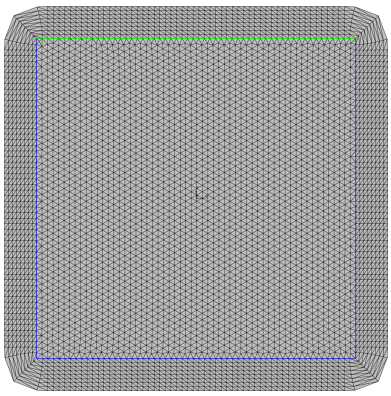
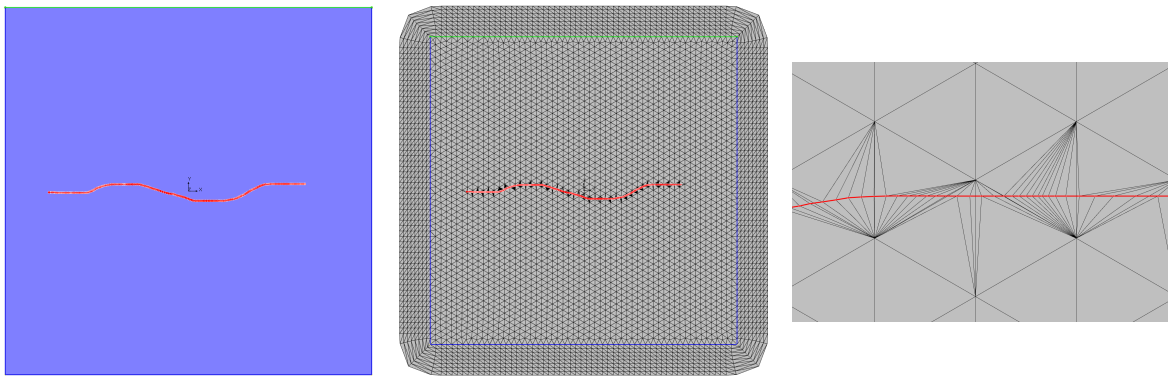


Figure 6.1: A background mesh used as a canvas for sketch input and field generation



(a) Filtered sketch input

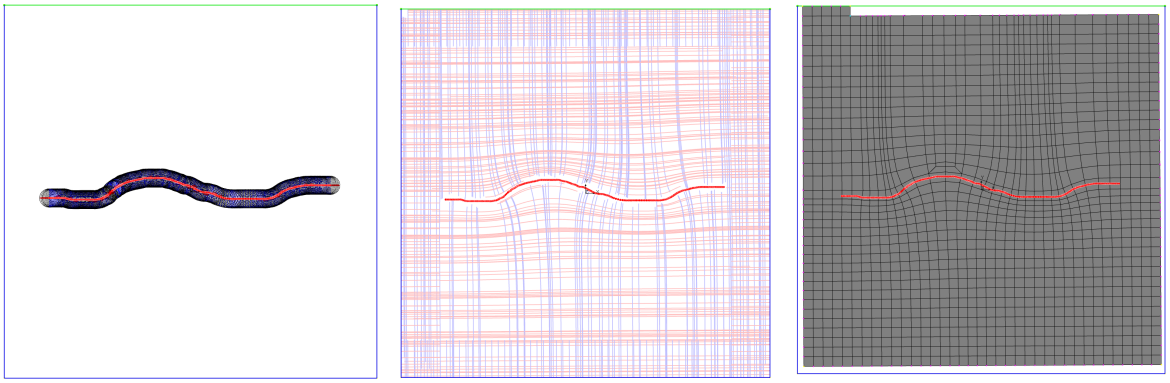
(b) Sketched path inserted into background mesh

(c) Zoomed-in view of updated background mesh

Figure 6.2: User sketch input after filtering the path (left) and inserting path points into the background mesh (center). On the right is a zoomed-in view showing the updated background mesh.

So far, the end-results have been meshes. The next example, shown in Figure 6.7, results in a mosaic-style logo. Edges and polygons in the mesh are exported in a vector graphics format for subsequent graphic editing.

Finally, we illustrate placement and shaping of anisotropic patterns on a CAD model [AVS15].



(a) Tensor boundary conditions set along sketched curve

(b) Resulting metric tensor field, visualized using streamlines

(c) A mesh representation of the field, obtained using the local 2D method

Figure 6.3: Example using sketch input

A metric field is generated from anisotropic boundary conditions set at various points in the domain (converted from user input). In practice, boundary conditions at such points are copied (transported, using the same type of tensor transport seen in Equation (3.18)) to nearby nodes (controllable through parameters such as number of layers and radius) to increase influence on the overall field. This also prevents the anisotropy at a boundary condition from quickly decreasing as one moves away from the boundary condition. The field is first used to place features via bubble-packing. Next, features are oriented and shaped (stretched, usually) using the same field and a scale factor (< 1.0) and placed at bubble centers. Finally, features can be generated in an automated fashion using a CAD API. These steps are shown in Figure 6.8.

6.2 Analysis Problems

6.2.1 Mesh Generation for Large Deformation Structural Statics

To demonstrate the efficacy of the proposed local method in 3D, a large deformation, elasto-plastic analysis of a lower control arm has been performed using ABAQUS [Sim08]. Figure 6.9

shows the control arm model, computed volume tensor field along with umbilic structures, hexahedral mesh fronts, and the resulting hex-dominant mesh. The oriented bounding box of the model has dimensions of $16.77 \times 4.61 \times 17.60$ units and pseudo-isotropy is imposed on the field with a nominal mesh size of 0.2 units. The current implementation is not yet fully optimized for performance, but the overall time for field generation, tiling, and generation of the final hex-dominant mesh is under 15 minutes.

A total of 8,046 hexes with an average length of 0.20 were generated during tiling. The final hex-dominant mesh consists of 62,300 elements, with an average edge length of 0.23. By quantity, 20,181 (32%) of the elements are hexes and 42,119 (67%) are tets. The hex elements constitute 73.52% of the total volume, whereas only 26.48% is occupied by tets. The minimum and maximum scaled Jacobian metrics of the hexes are 0.4 and 1.0, and the average is 0.93. The radius-ratio metric for the tet elements ranges from 3.0 (ideal) to 697.69, with an average value of 4.12. The distributions of hex and tet element quality according to these metrics are presented in Figure 6.10.

For the analysis, an elasto-plastic material model based on isotropic hardening was specified with generic parameters for steel: E=210 GPa, ν=0.3, σ_y=480 MPa, ultimate strength σ_u=550 MPa, and tangent modulus $E_{p,t}$=349.42 MPa. The interior cylindrical surfaces on the left portion of the model were displacement constrained, and lateral and vertical loads of 1,080 kN and 540 kN, respectively, were transmitted to the cylindrical surface at the apex. The deformed configuration is shown in Figure 6.11, in which the gray regions denote fully plastic zones.

Inspection of the resulting hex-dominant meshes reveals that a significant portion of the hex fronts are recovered at the end of the meshing process, and with reasonable quality overall. However, the results also warrant further work on improvement of the worst quality elements (primarily tets) and perhaps a more aggressive strategy for hex recovery.

6.2.2 Highly Anisotropic Adaptive Remeshing for CFD

We apply the global 2D method to a steady supersonic flow over a 0012 airfoil. Using the FUN3D solver [FUN11], an adaptation loop is carried out following the high-level data flow diagram in Figure 6.12 (with orange process blocks further described in Figures 6.13, 6.14, and 6.15) — a detailed version of Figure 5.1 that employs the techniques presented in this thesis along with the FUN3D solver. To trace the execution, begin with the input meta-block consisting of an initial mesh, boundary conditions (BCs), and solver parameters. These are passed to the solver to produce a solution and solution-derived metrics on the mesh. The metrics then feed into the adaptive remeshing meta-process, which is further explained below. This meta-process produces a new mesh, which is then passed to the solver with the original BCs and solver parameters. This process continues until termination (either planned or due to exhaustion of resources, etc.) and yields a final adapted mesh with solution.

The airfoil has a reference length of 1, and is embedded in a circular domain with a radius of 20 chord-lengths. The parameters specified for the laminar, viscous analysis are as follows: $Mach = 2$, $Re = 1 \times 10^6$, and the angle of attack $\alpha = 2^\circ$. The solutions are performed according a schedule that starts off with first-order accurate iterations, followed by second-order accurate iterations during which a flux limiter is enabled. The inviscid flux construction scheme is `LDFSS`, and the `hVanAlbada` flux limiter is used. The initial mesh for these adaptation iterations was prepared by running eight adaptation iterations with the proposed method on an initial un-adapted quad mesh with $Re = 1 \times 10^3$. Figures 6.17 and 6.18 shows the meshes and Mach plots from the resulting solutions for adaptation iterations 0–11, starting with the input mesh at Iteration 0. The scale for the solution plots ranges from 0 to approximately Mach 2. Over the course of the adaptation run, the number of elements grows monotonically from $37,433$ to $82,078$ and aspect ratios on the order of 100 are achieved (maximum specified in metric construction). During this process a quad-to-tri element count ratio of $\approx 82\%$ is maintained. A sample log-convergence history for the mass,

momentum, and energy residuals is provided in Figure 6.16. Overall, the increase in element count over iterations is accompanied by improved resolution of the bow shock and wake in the solution.

6.3 Manufacturing

6.3.1 Model Synthesis for Additive Manufacturing

Results are underway for preparing designs that are suitable for additive manufacturing. The basic recipe is similar to the graphic and product design examples:

1. Generate a metric field that respects boundary alignment and possibly user input.
2. Adjust this field to produce some anisotropic gradation, such as transitioning from anisotropy at the boundary to isotropy in the interior.
3. Produce a tetrahedral or hexahedral-dominant mesh.
4. Produce a realizable structure from the edges of the mesh.
5. Alternatively, generate streamlines and/or streamsurfaces of the metric field to form a realizable structure.

Following are some preliminary results using the "cimplex" model (`cimplex_1.hh.stp`) from "cast-then-machined" models in the National Design Repository [RG97]. Figure 6.19 shows a form-fitted metric field (with exaggerated anisotropy) and a structure generated by emanating streamlines from sample points in the volume. Starting from each sample point, streamlines are integrated in both directions along all three eigenvector-fields.

6.4 Observations

Several key observations can be drawn from the present work. In part, these observations are based on experience gained while formulating and implementing the proposed methods in an abstract setting. It is also important to consider the unique challenges presented by problems in different application domains (as seen in this chapter); they serve both as motivations and measuring sticks to evaluate (and define the meaning of) the performance of the proposed methods. In the following, we summarize some of these observations.

Metric Field Generation Using the metric diffusion and form-fitting approach on surfaces and volumes, it is possible to generate orientation metric fields that capture boundary alignment as well as alignment with user input that may lie on the interior of the domain. As stated previously, these fields are pseudo-isotropic, meaning that a slight perturbation in eigenvalues is used to maintain an orientation with three distinct principal directions. This is demonstrated through applications such as the mosaic logo and hex-dominant mesh with targeted all-hex regions (for structural analysis), and structure generation for additive manufacturing. Results show that boundary-aligned (and input-aligned) orientation can be attained and is generally suitable when used with bubble-packing methods. As for limitations, tensors may undergo 90° rotations on the boundary from node to node — this creates umbilics in the field. A consequence is rapid (as one moves in space proximal to an umbilic) bends in streamlines and streamsurfaces of the eigenvector-fields, even though eigenvector frames are closely aligned (ignoring correspondence by ordering). Another limitation is the lack of explicit control over the location and type of umbilics in resulting metric fields. This can be seen in the "lobes" of the heart logo (Figure 6.7), which exhibit different (asymmetric) patterns due to different locations and types of umbilics present.

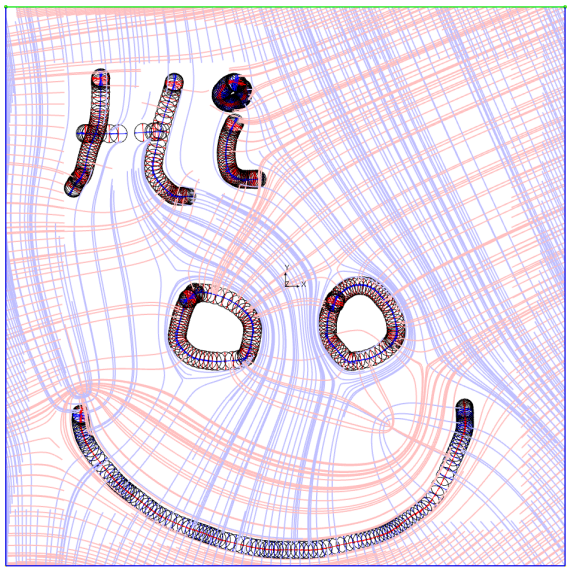
Alternatively, one can prescribe anisotropic tensors as boundary conditions in order to obtain a fully anisotropic metric field. This is primarily demonstrated in design examples

(*e.g.*, Figure 6.6 and Figure 6.8) that convert user input to anisotropic tensor boundary conditions. These examples show that diffusion can be an effective tool to produce variation in orientation and anisotropy over the domain, given sufficient boundary conditions. However, this method doesn't provide control over the gradation of orientation and anisotropy, nor over umbilics (see above). To some extent, the former is mitigated by enforcing additional boundary conditions: copying (transporting) a single node's boundary condition to its neighbors (to increase overall influence as well), or setting boundary conditions along entire curves. Another technique is to post-process the diffused field to create an anisotropic variation away from boundary conditions. However, that is a very application-specific technique.

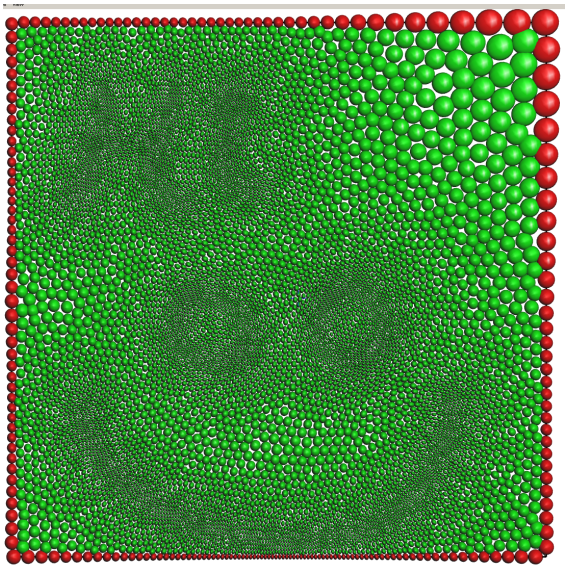
Performance of Proposed Methods Regarding the analysis applications in the present work, we have demonstrated that the proposed method can aid capture of solution features and promote solver convergence. For instance, anisotropic adaptive remeshing results show increased mesh anisotropy and improved capture of the Mach profile (when it is used to compute solution-based metrics) over adaptation iterations. Within iterations, solver convergence may be assisted by the meshes produced here. The question then arises of how to objectively measure the performance of the proposed methods for analysis problems. Some preliminary thoughts include studying convergence of solutions outputs (*e.g.*, lift-to-drag), metric conformity of produced meshes, and solver residual convergence. The methods presented here should be systematically benchmarked against current state-of-the-art methods, perhaps as done by [IBK$^+$17]. Also, it may be beneficial to incorporate some of these measures (*e.g.*, metric non-conformity) into the methods (say anisotropic mesh generation in regions of interest) in order to better satisfy field orientation and anisotropy requirements.

In another sense, "performance" can refer to computational complexity and implementation performance when operating on metric fields and generating meshes. There is certainly room for improvement on both fronts. Considering that metric fields are represented by

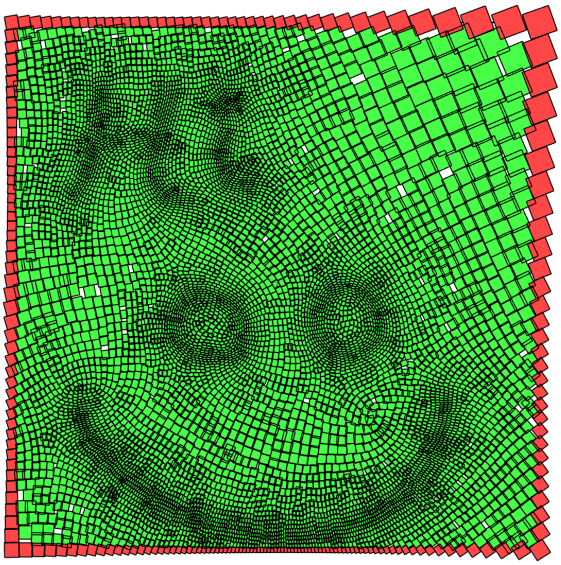
metric components on a mesh, it should be possible to take advantage of the same opportunities that solvers have: distributed parallelism among compute nodes using mesh partitioning, process- and thread-level parallelism within a compute node, and instruction-level parallelism within processing elements when performing local calculations. While computational complexity concerns should be handled during algorithm design, implementation performance is something that can be improved once the methods are proven further.



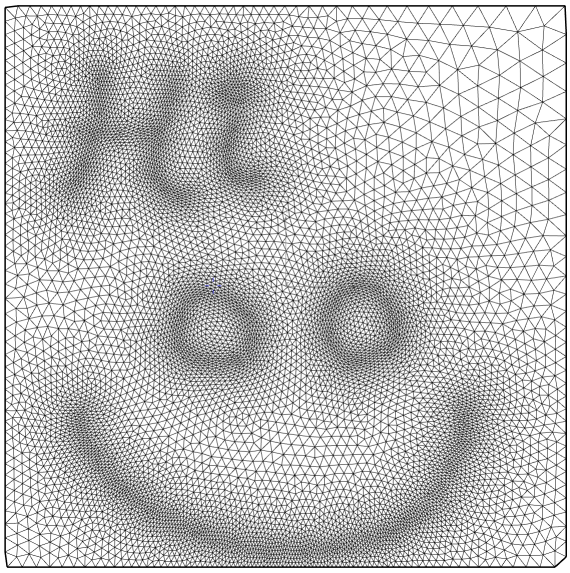
(a) Sketched input curves and resulting metric field streamlines



(b) Anisotropic bubble packing of metric field

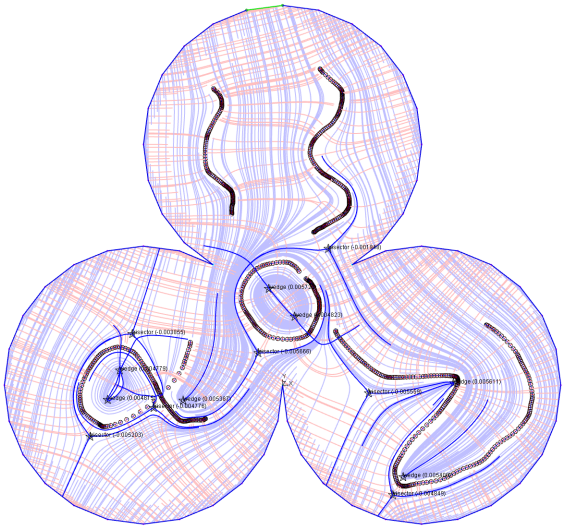


(c) Anisotropic rectangular solid cell packing of metric field

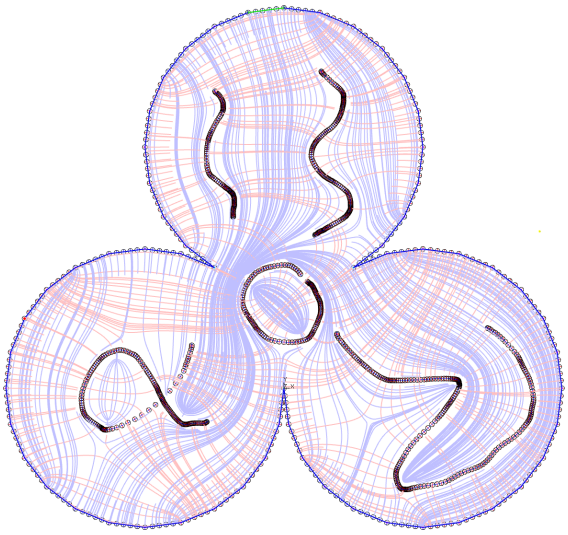


(d) Anisotropic triangular mesh of domain using field

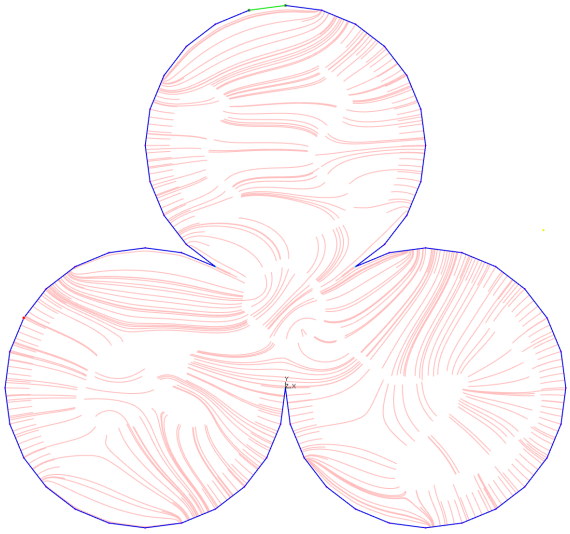
Figure 6.4: Sketch-based anisotropic metric field and mesh generation



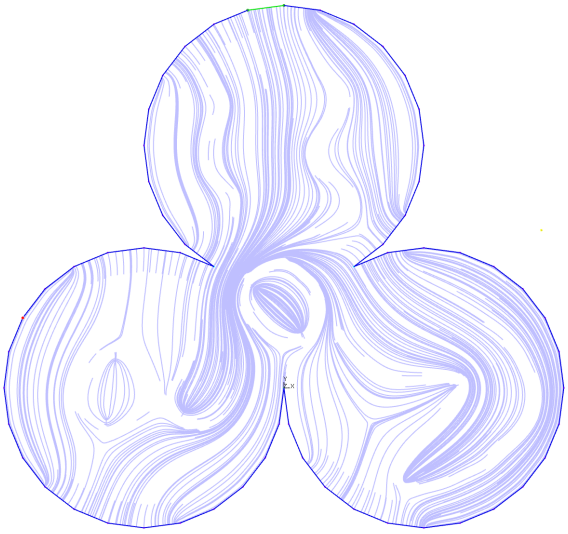
(a) Raw metric field with umbilics based on sketch input



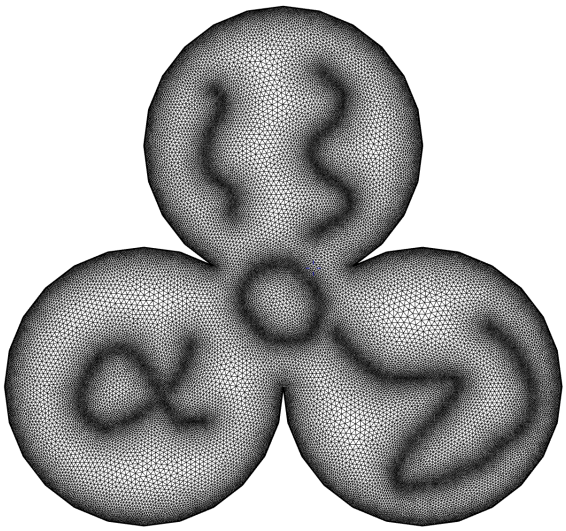
(b) Metric field adjusted to align with boundary



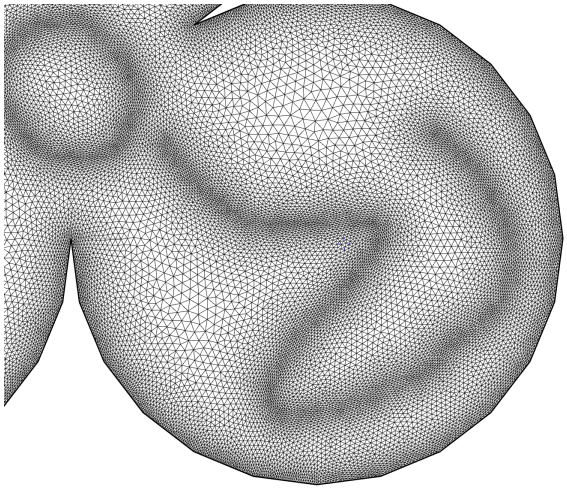
(c) Primary streamlines of metric field



(d) Secondary streamlines of metric field



(e) Anisotropic triangular mesh produced from field



(f) Zoomed-in view of anisotropic triangular mesh produced from field

Figure 6.5: Sketch-based field and mesh generation incorporating anisotropy and boundary alignment

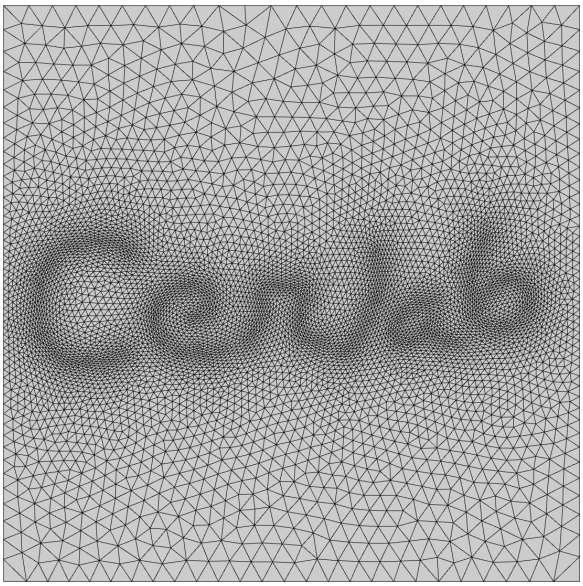
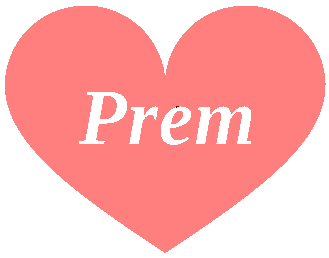
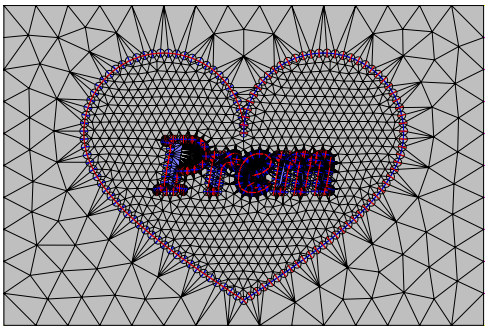


Figure 6.6: Computational Engineering and Robotics Lab

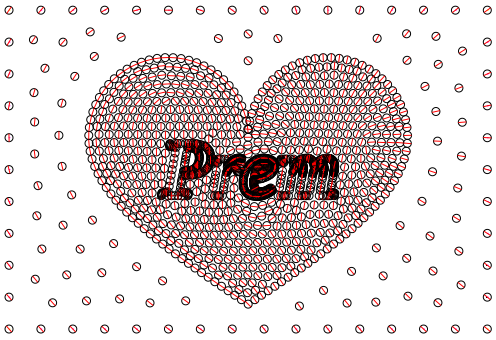


(a) Initial geometry



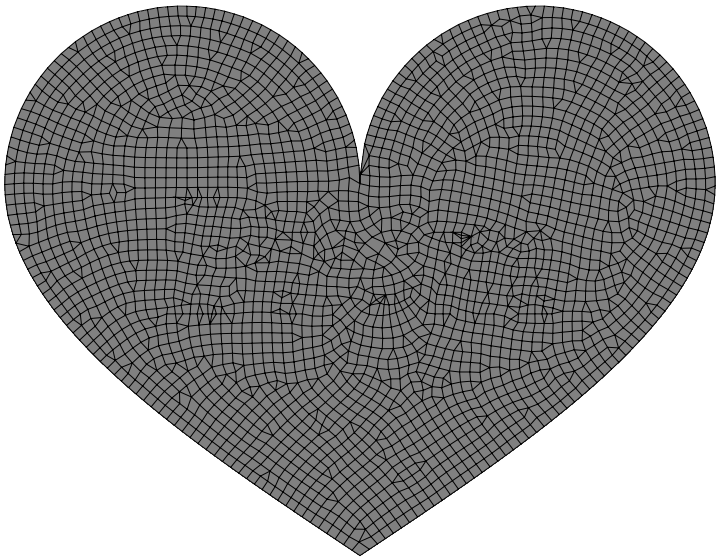
(b) Background mesh

(c) Pseudo-isotropic metric field aligned with boundary and inset text



(d) Primary metric directions shown using glyphs

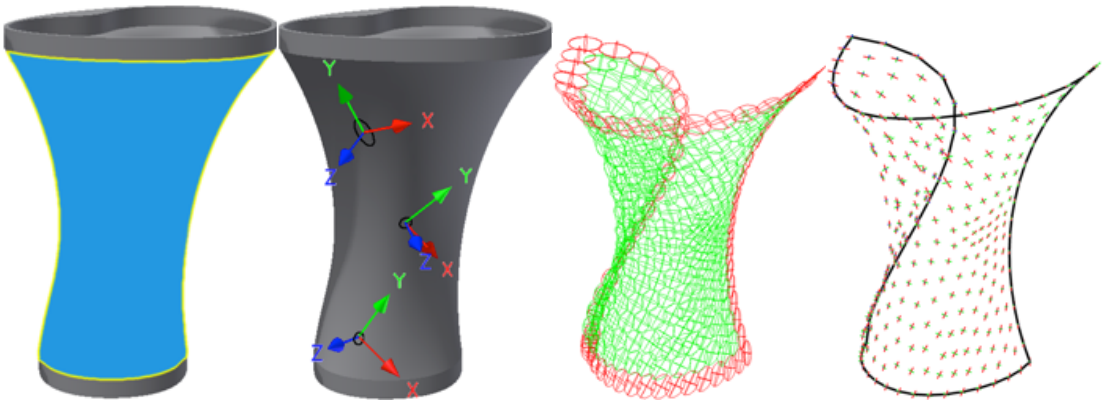
(e) Secondary metric directions shown using glyphs



(f) Resulting quad-dominant mesh

(g) Result after post-processing mesh in a vector graphics editor

Figure 6.7: Logo development using isotropic feature-aligned field generation, bubble-packing, and vector graphics editing

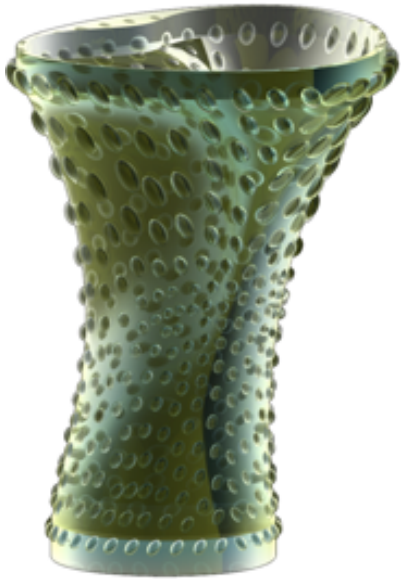


(a) Original CAD geometry

(b) Anisotropic boundary conditions converted from user input

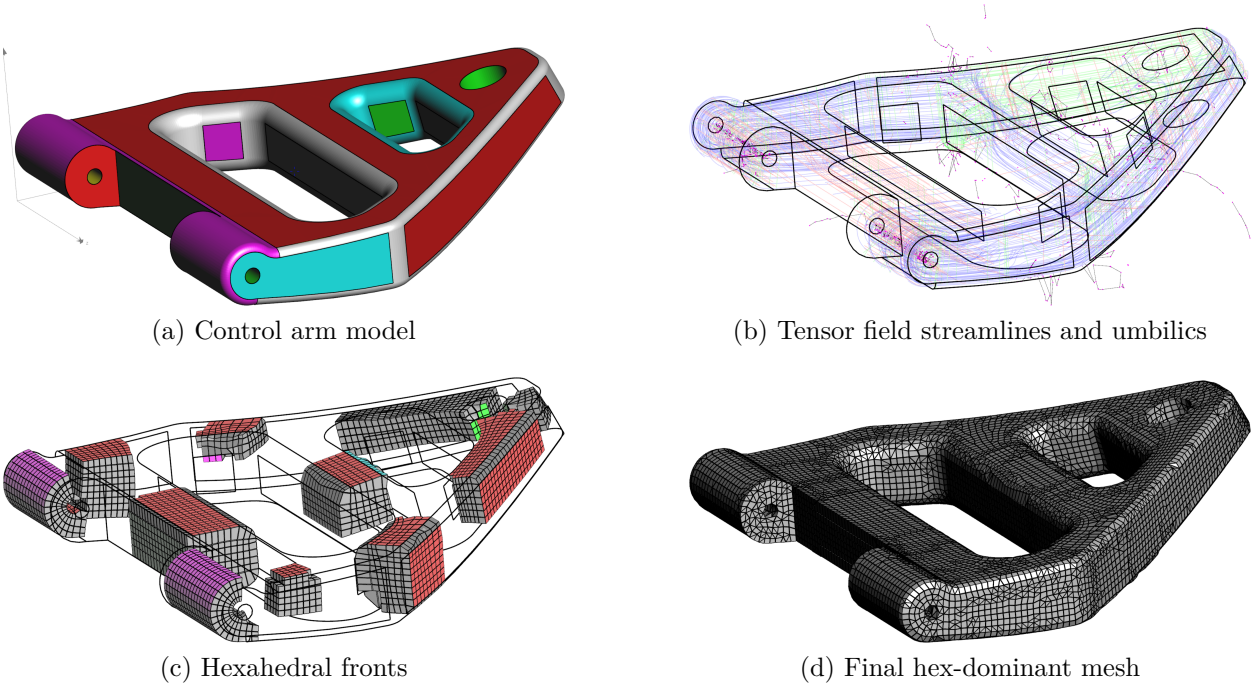
(c) Bubble-packing of generated field for feature placement

(d) Glyph representation of orientation and anisotropy for generated features



(e) Final rendering of CAD geometry with generated pattern

Figure 6.8: Placement and shaping of anisotropic features on a CAD model (reproduced from [AVS15])



(a) Control arm model

(b) Tensor field streamlines and umbilics

(c) Hexahedral fronts

(d) Final hex-dominant mesh

Figure 6.9: Control arm results

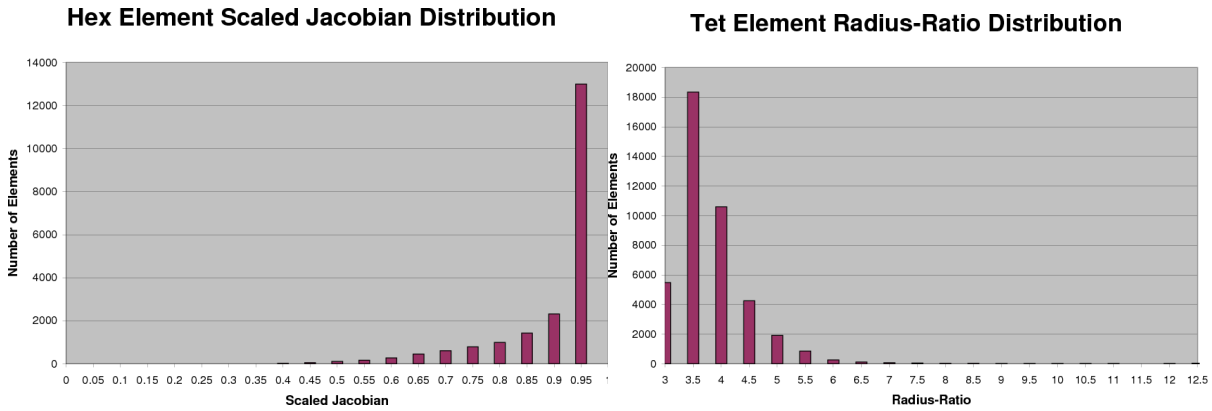
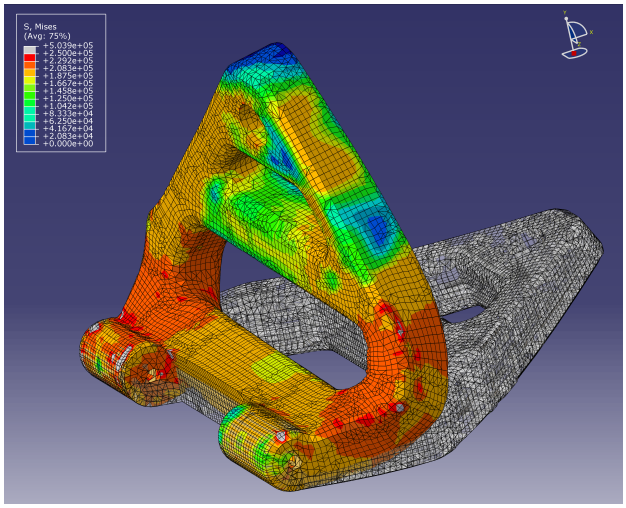
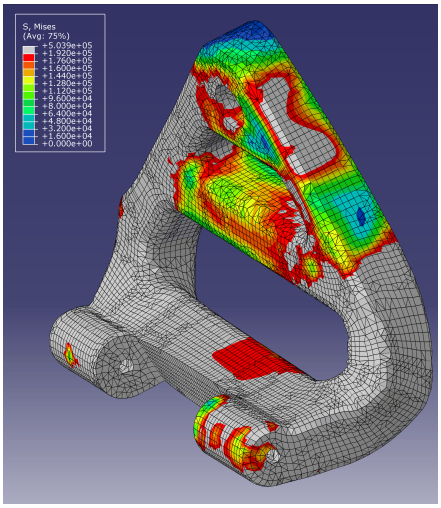


Figure 6.10: Hex (Scaled Jacobian) and tet (Radius-Ratio) quality distributions



(a) Von-Mises plot on deformed configuration along with undeformed configuration



(b) Deformed configuration with plastic zones shown in gray

Figure 6.11: Von-Mises stress distribution with fully plastic zones in deformed configuration

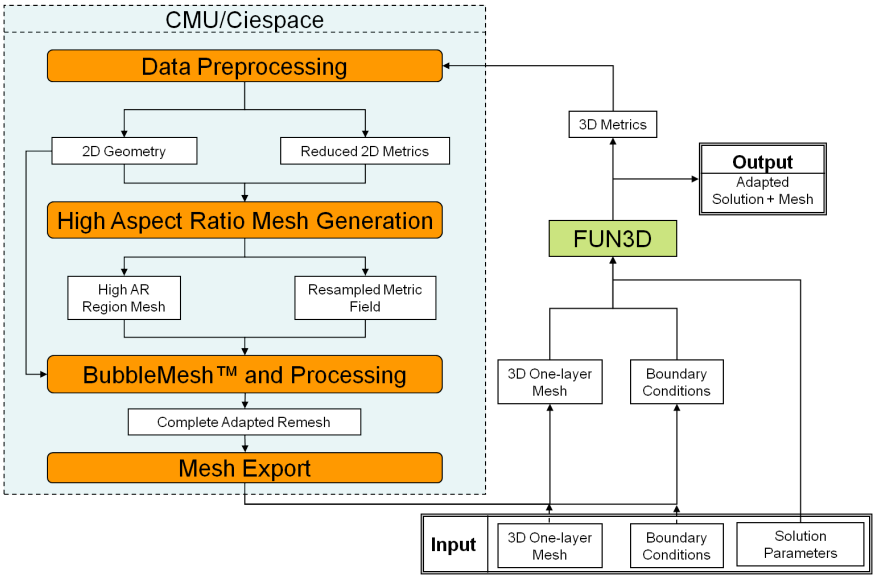


Figure 6.12: High-level adaptation data flow diagram: white rectangular blocks represent input/intermediate/output data, while rounded rectangular blocks and the FUN3D block represent processes

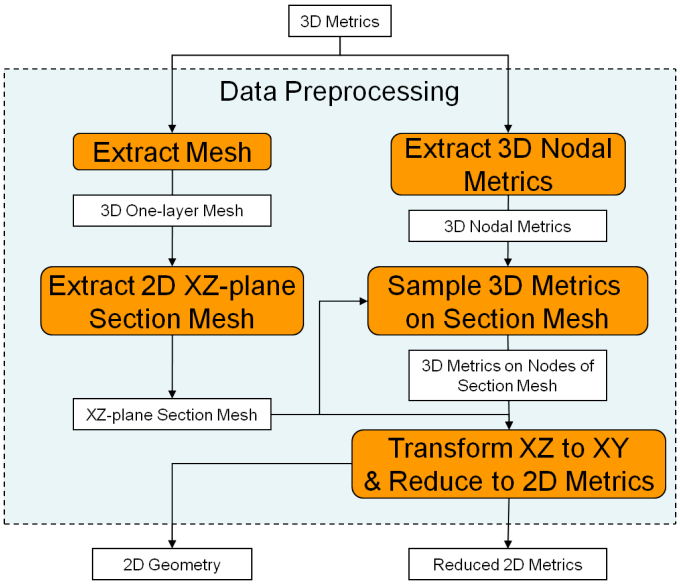


Figure 6.13: Exploded data flow diagram for data pre-processing

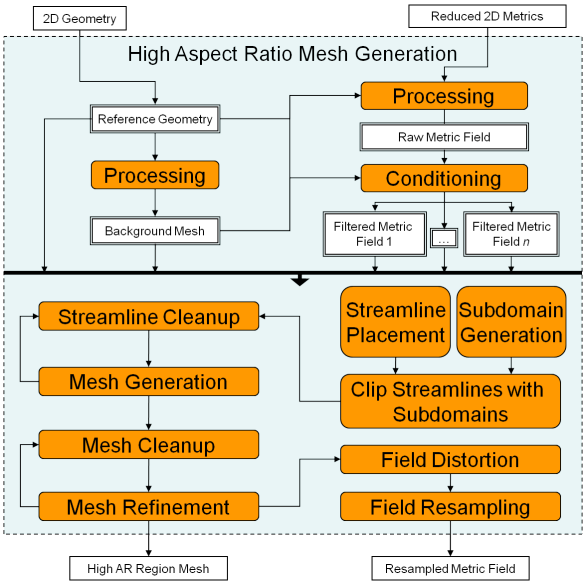


Figure 6.14: Exploded data flow diagram for high aspect-ratio mesh generation

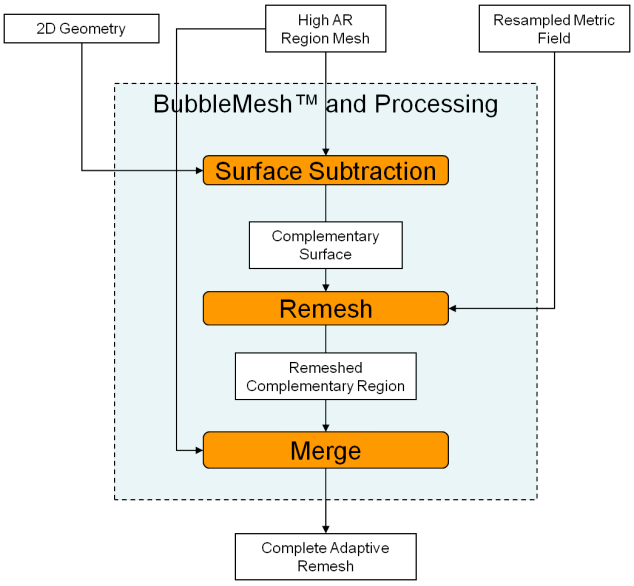


Figure 6.15: Exploded data flow diagram for assembly of complete adapted mesh using BubbleMesh®

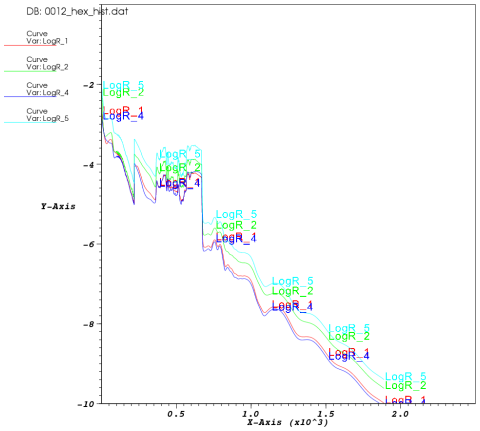
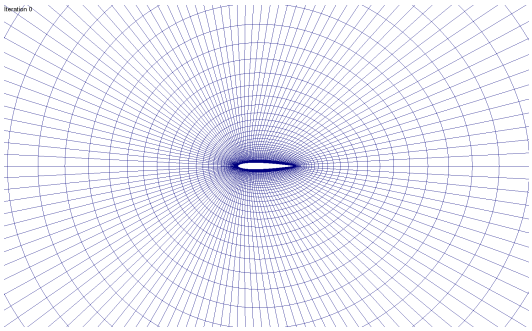
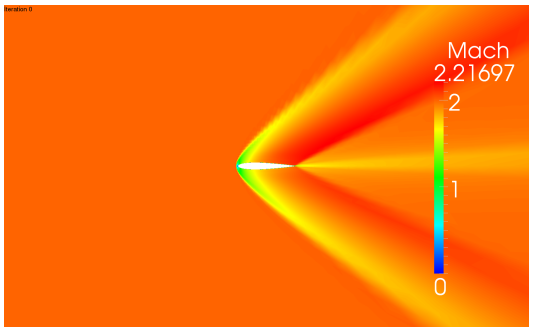


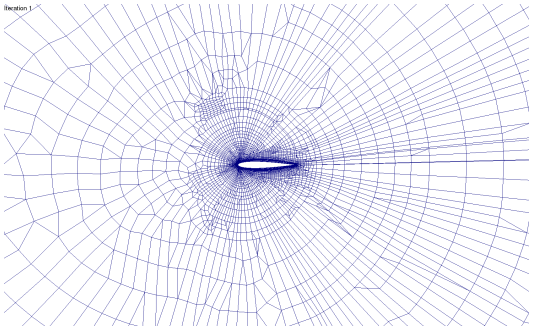
Figure 6.16: Convergence history for an adaptation iteration



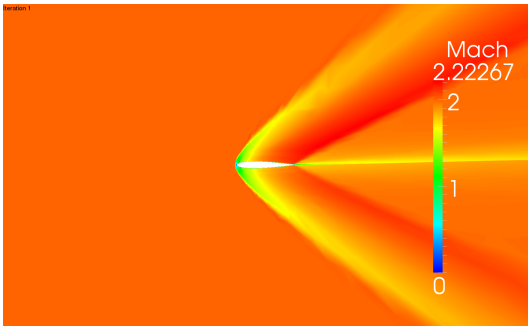
(a) Iteration 0 mesh



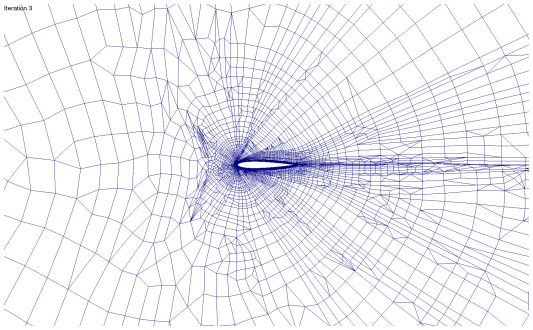
(b) Iteration 0 solution



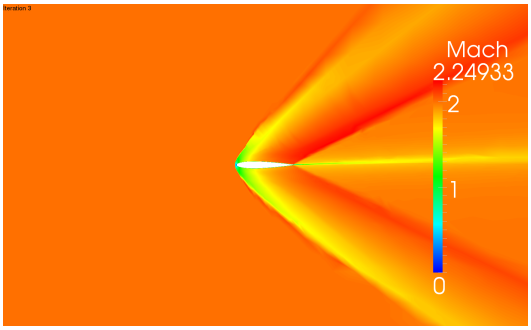
(c) Iteration 1 mesh



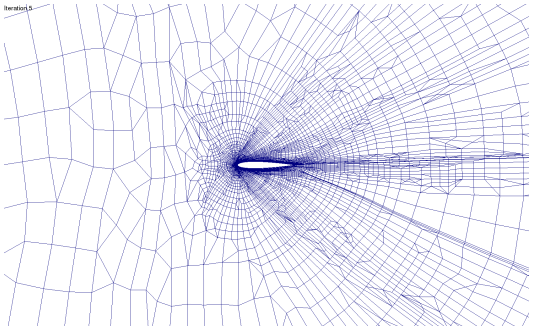
(d) Iteration 1 solution



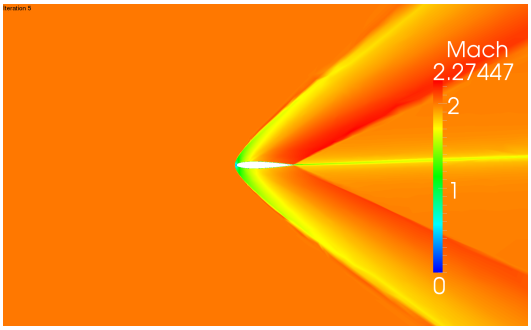
(e) Iteration 3 mesh



(f) Iteration 3 solution

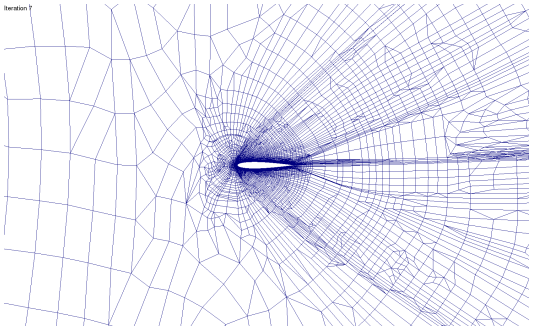


(g) Iteration 5 mesh

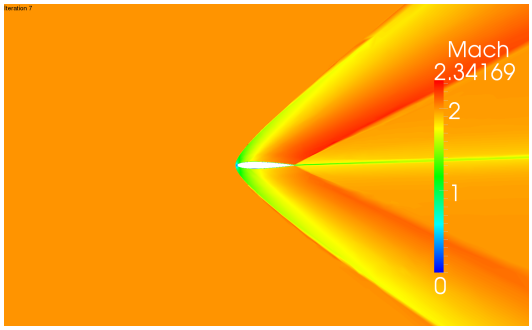


(h) Iteration 5 solution

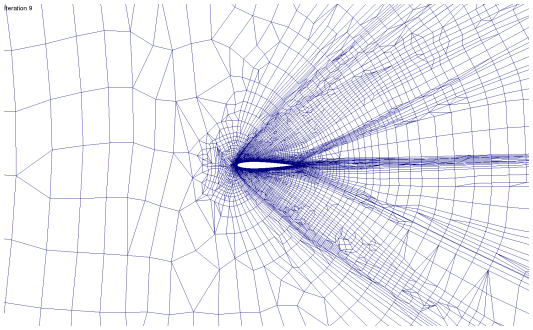
Figure 6.17: Adapted meshes and Mach plots for the 0012 airfoil (Iterations 0–5)



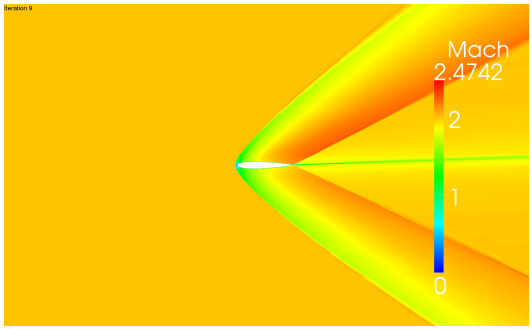
(a) Iteration 7 mesh



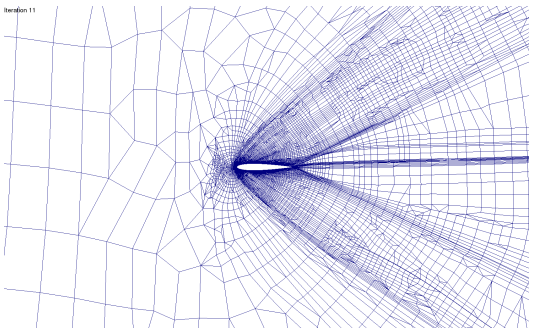
(b) Iteration 7 solution



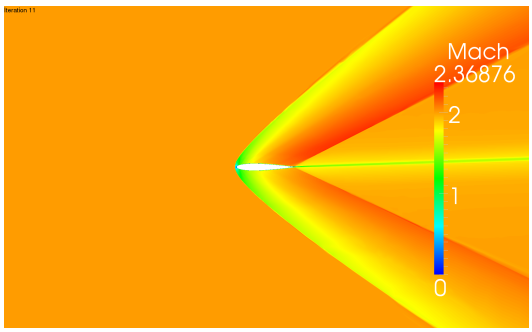
(c) Iteration 9 mesh



(d) Iteration 9 solution

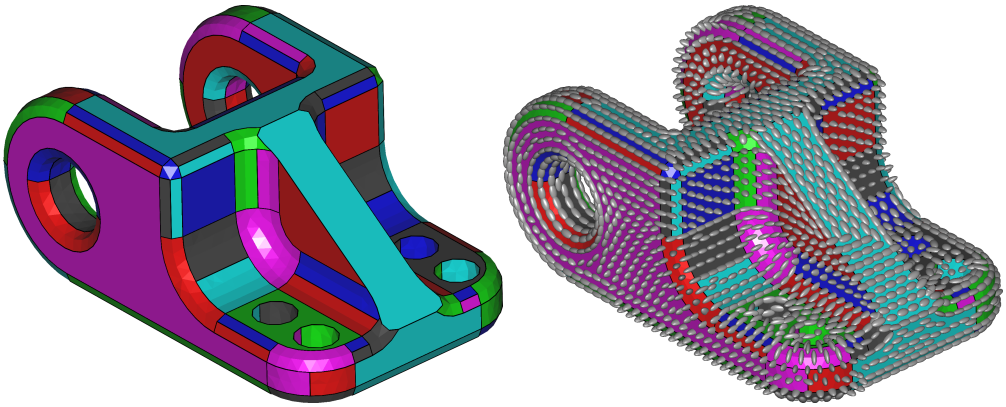


(e) Iteration 11 mesh

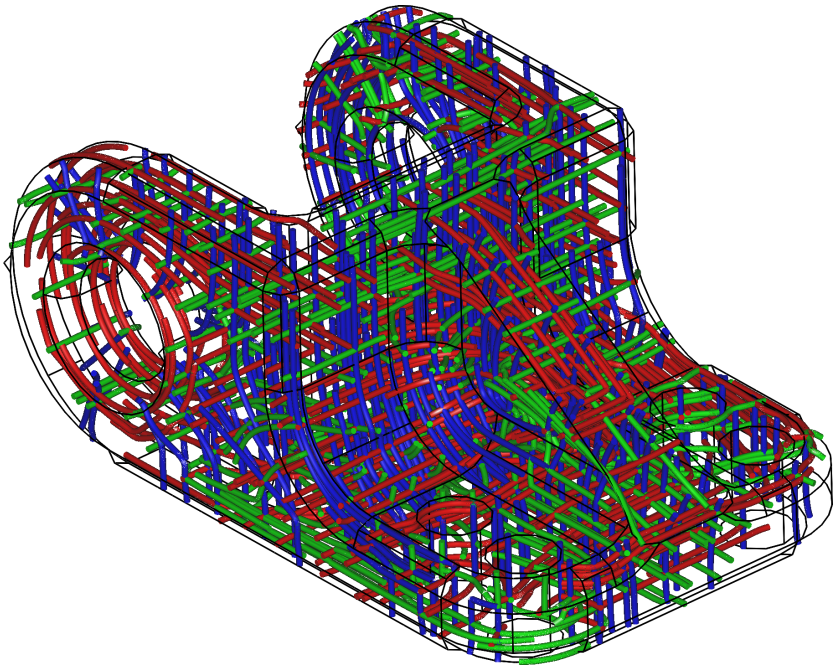


(f) Iteration 11 solution

Figure 6.18: Adapted meshes and Mach plots for the 0012 airfoil (Iterations 7–11)



(a) Tessellation of cimplex CAD geometry (b) Ellipsoid glyphs of boundary-aligned metric field



(c) Raw visualization of streamline-based structure

Figure 6.19: Structure generated for cimplex model for additive manufacturing

Chapter 7

Conclusion

This thesis has presented a framework using Riemannian metric tensor fields for controlling orientation and anisotropy. Several tools have been developed within this framework to address the problems of representing, generating, and conditioning orientation and anisotropy information for various applications. Additional tools have been proposed to exploit metric fields for the generation of anisotropic meshes and to engage in adaptive remeshing for analysis problems.

First, we discussed the representation and manipulation of metric tensor fields. This includes constructing a continuous metric field from discrete data (interpolation), generating metric fields based on geometry, studying their topology, and visualizing metric data. Techniques to de-noise and fair solution-based metrics were discussed and specialized for our particular applications.

Then, we developed a "local" meshing approach that starts meshing in an advancing front mesh from anywhere in the input geometry, usually guided by a metric field computed from the geometry. After developing a 2D proof of concept that utilizes metric streamlines to shape elements, this was extended to a local 3D method utilizing metric streamsurfaces. Acknowledging the limitations of this approach, a "global" meshing approach was developed

in 2D to more flexibly handle arbitrary orientation and anisotropy requirements.

Next, we discussed an adaptation controller that coordinates these components for adaptive remeshing problems. In conjunction with the 2D global method, it was used to support adaptive remeshing for problems requiring high aspect-ratio elements with favorable orientation.

Finally, recipes using all of these techniques were employed in different ways to address applications in various application domains:

Graphic & Product Design For the orientation and placement of mosaic tiles, pseudo-isotropic tensor fields were generated based on boundary alignment as well as user input alignment (*e.g.*, sketch-based curves). Meshes produced using these fields are consumed by methods that produce vector graphics designs. In a similar fashion, anisotropic fields can be generated from user constraints (at individual nodes, along curves, and in local neighborhoods of both) and then optionally post-processed to produce controlled anisotropic gradation. This has been applied to pattern generation for CAD models with patterns oriented, sized, and spaced by metric fields.

Structural & Fluid Dynamics In the first application, large deformation structural statics, boundary-aligned fields were generated using the form-fitting approach. From these pseudo-isotropic fields, it is possible to grow hexahedral-element regions using the local 3D approach and incorporate them into a quality hex-dominant mesh. The second application is anisotropic mesh generation for adaptive CFD problems. Here, a more complex recipe employed metric field conditioning, identification of critical regions, generation of a highly-anisotropic field-guided mesh in these critical regions, and a method to produce a complete mesh of the domain for the next iteration.

Additive Manufacturing This is ongoing work in which structures are generated from boundary-aligned fields on an input geometry. As with the design applications, there are different options that can be explored when it comes to generating & post-processing fields and generating structures. The presented example utilized metric-streamlines to form the structure. It is also possible to generate a mesh and use its connectivity to form elements of a structure.

In conclusion, the methods proposed in this thesis show promise when it comes to applications in various domains. It is our hope that they can evolve to tackle industry-grade problems through future study and refinements.

The contributions of the thesis were briefly stated in Section 1.3. To elaborate:

- Metric field generation: In this work, we present various methods to generate pseudo-isotropic and anisotropic fields on surfaces and volumes using the Laplace operator (tensor diffusion). Tensor diffusion is an existing concept; for instance, it has been used to "smooth" existing fields. The major contributions of this work in this regard are:

    1. Innovative specification of boundary conditions to generate anisotropic fields: conditions are set at interior points, along curves, and along surfaces, as well as on their local neighborhoods. This allows us to "fill in" the field over the entire domain in a way that produces variation in orientation and anisotropy.

    2. Boundary-aligned orientation field (pseudo-isotropic) generation for surfaces and volumes: a new approach has been developed in which fields are generated and updated over a domain while iteratively adding boundary-aligned boundary conditions.

- Employing conditioning (noise removal, fairing, blending, and smoothing) to address issues with existing and generated metric fields: here, we adapt and extend existing

techniques in order to produce multiple fields that can be independently queried for anisotropy (de-noised fields) and orientation (faired fields). Tensor interpolation and diffusion are also used in interesting ways to resample fields from one mesh to another, to transition between two fields, and to transition between existing fields and new meshes.

- Novel mesh generation algorithms that exploit metric field structure: we developed local 2D and 3D meshers that can grow valid meshes from the inside-out given an appropriate field (*i.e.*, boundary-aligned field). The meshers utilize streamlines of metric eigenvector-fields to shape elements and to guide marching directions. We also developed a global 2D mesher. It uses the existing concept of generating a mesh from streamlines of a metric or curvature tensor field. However, the method is specialized in several ways. First, a new integration step-size scheme has been developed to obtain more suitable streamlines. Second, we have proposed methods for determining critical regions in a domain — *i.e.*, where more anisotropic elements should be generated — and we confine the highly anisotropic meshing scheme to these regions. Third, we have developed streamline cleanup and processing operations to obtain valid meshes which may better reflect gradation in anisotropy. Next, we apply refinement templates to the resulting meshes to improve capture of anisotropy. Finally, we have a scheme to combine this specialized mesh with a mesh outside the critical regions.
- Successful application of these techniques to design, analysis, and manufacturing problems: recipes using all of the aforementioned tools in various orders and combinations have been developed to target different applications.

7.1 Ongoing and Future Work

The observations drawn from this work naturally lead to ideas on continued and future work.

The first is a "unified" framework for metric tensor field processing, motivated by some amendments to the initial problem statement in this work. Suppose that metric tensors are retained for representing orientation and anisotropy — at least as a *lingua franca* between components inside and outside the framework (metric field generators, mesh generators, pattern generators, solvers, adaptation codes, etc.). Then the question is, can we generate metric fields:

- While *simultaneously* factoring in different sources of information: user input, geometry, pre-existing meshes, and solution-based information?
- With the ability to satisfy various constraints on anisotropy (*e.g.*, maximum aspect-ratio)?
- With control over field properties such as spatial gradation (of orientation and anisotropy)?
- With control over metric field topology (locations and types of umbilics)?

Furthermore, can conditioning of metric fields be folded into field generation?

A proposed general solution to this is the "unified" framework mentioned above. Its main components are to:

1. Distill input information (user input, geometry, existing mesh elements, and solution-based information) into constraints on orientation, anisotropy, and/or gradation in either. Constraints may be partial; *e.g.*, fewer than three principal directions and/or sizes may be specified. Constraints may also apply to derived quantities like aspect-ratio. Constraints may also be placed on umbilic locations and types.
2. Use a method to generate a field over the whole domain while satisfying all the constraints as best as possible.

It remains to find a suitable field generation method to accomplish this. Regarding the input factors, there are many geometric factors that can be distilled into orientation and anisotropy constraints: lengths & areas, curvatures (of curve and surfaces) and torsion, and

thicknesses (of surfaces and volumes). Such a framework could be a truly anisotropic extension of work by Quadros et al. (see [QVB+10] and related works) on isotropic sizing function generation. Note that there are anisotropic "source-entity" methods such as the work by Aubry et al. [ADKM15]. The potential applications and benefits of a unified framework are myriad. Once again, consider mesh adaptation for CFD problems. Simultaneous consideration of different inputs (*i.e.*, solution-based metrics *and* boundary-layer meshing parameters from a user) could protect mesh anisotropy in a boundary-layer region at the metric level. This is often treated at the meshing level by enforcing layered elements in the boundary-layer region [PLK+16]. That might still be needed, though this would provide support at the metric level.

In addition to this, other ongoing work includes generating anisotropic structures for additive manufacturing. One approach is to make a structure from the edges of an anisotropic field-guided mesh. It may be interesting to further express anisotropy by modulating local thicknesses and other geometry parameters based on local requirements.

As a future goal, it would be exciting to revisit the idea of anisotropic 3D mesh generation and adaptation with high aspect-ratio hexahedral elements in critical regions. The unified framework and a suitable method for creating global streamsurfaces (or similar) are likely to be key ingredients.

Finally, there are some interesting open questions that have arisen from the present work:

- Anisotropy-orientation compatibility: if a mesh is a discrete sampling of a continuous metric field, it is not always possible for the mesh to satisfy the orientation *and* anisotropy requirements from a metric field. Under what conditions on a field can a mesh satisfy both (to a prescribed degree)? On a related note: the local mesh generation methods in this thesis prioritized orientation capture at the expense of correct anisotropy.
- Path independence and streamline integration: this is perhaps related to the previous

question. Suppose we integrate a unit-length (under the local metric) streamline along one eigenvector field, and from its endpoint integrate another unit-length streamline along another eigenvector field. Then consider swapping the order of the two-part integration. In general, one will not reach the same overall endpoint. Under what conditions on a metric field will that occur? This has been observed in the development of the local 2D mesh generation scheme, where we extend streamlines beyond unit-length in the hopes of finding an intersection. The 3D method similarly extends streamsurface patches, but also note that each patch consists of nets of streamlines that do not generally intersect — which is why closest points are sought.

- Deeper connections to differential geometry: Can any insight be gained by explicitly realizing a mapping for which a metric field is the implied metric? What conditions (*e.g.*, the Gauss-Codazzi equations, etc.) must a metric field satisfy to realize such a mapping? Note: there is some very recent (at the time this thesis is being submitted) relevant work that remains to be studied by the author, see [ZWL+18].

Perhaps answers to these questions can provide guidance in the generation and conditioning of metric fields.

Nomenclature

$:$	Double contraction
$\cdot$	Inner (dot) product; single contraction
$\nabla_{\boldsymbol{\mathcal{M}}}$	Gradient with respect to metric tensor $\boldsymbol{\mathcal{M}}$, $\frac{\partial}{\partial \mathcal{M}_{ij}} \mathbf{e}_i \otimes \mathbf{e}_j$
$\boldsymbol{\mathcal{M}}$	A second-order metric tensor
$\mathcal{M}_{ij}$	Components of a second-order metric tensor
$\nabla, \nabla_{\boldsymbol{x}}$	Spatial gradient operator, $\frac{\partial}{\partial \boldsymbol{x}} \mathbf{e}_i$
$\otimes$	Outer (tensor) product
$\underline{\underline{\mathbf{C}}}$	A fourth-order tensor
$\mathbf{A}$	A second-order tensor
$\times$	Vector cross product
$\boldsymbol{v}$	A vector
$\|\boldsymbol{v}\|, \|\mathbf{A}\|_2$	L^2 (Euclidean, Frobenius) norm
$\|\mathbf{A}\|_2^2$	Square of L^2 norm
$\|\boldsymbol{v}\|_{\boldsymbol{\mathcal{M}}}$	L^2 norm under metric $\boldsymbol{\mathcal{M}}$

Bibliography

[ACSD+03] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22(3):485–493, 2003. 3.5, 3.5, 4.4

[ADKM15] R. Aubry, S. Dey, K. Karamete, and E. Mestreau. Smooth anisotropic sources with application to three-dimensional surface mesh generation. *Engineering with Computers*, September 2015. 00000. 7.1

[AFPA06] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Fast and Simple Computations on Tensors with Log-Euclidean Metrics. Rapport de recherche RR-5584, INRIA, 2006. 3.4.2

[AM05] T. Akenine-Möller. Fast 3d triangle-box testing. SIGGRAPH '05: ACM SIGGRAPH 2005 Courses, Article 8, 2005. 4.3.5

[AVS15] Diego Andrade, Ved Vyas, and Kenji Shimada. Automatic Generation of Anisotropic Patterns of Geometric Features for Industrial Design. *Journal of Mechanical Design*, 138(2):021403, December 2015. (document), 6.1, 6.8

[BGPJ06] K.L. Bibb, P.A. Gnoffo, M.A. Park, and W.T. Jones. Parallel, gradient-based anisotropic mesh adaptation for re-entry vehicle configurations. In *Proceedings of the 9th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*. American Institute for Aeronautics and Astronautics, 2006. 2.3, 3.7.2

[BH96] F. J. Bossen and P. S. Heckbert. A pliant method for anisotropic mesh generation. In *Proceedings of the 5th International Meshing Roundtable*, pages 63–76, 1996. 3.2

[Bla00] T. Blacker. Meeting the challenge for automated conformal hexahedral meshing. In *Proceedings of the 9th International Meshing Roundtable*, pages 11–19, 2000. 4.1

[BM93] T. D. Blacker and R. J. Meyers. Seams and wedges in plastering: A 3d hexahedral mesh generation algorithm. *Engineering with Computers*, 2(9):83–93, 1993. 2.2.1

[BPM+95] Steven E. Benzley, Ernest Perry, Karl Merkley, Brett Clark, and Greg Sjaardema. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings of the 4th International Meshing Roundtable*, pages 179–191, 1995. 2.2.1

[DS05] Arbtip Dheeravongkit and Kenji Shimada. Inverse pre-deformation of finite element mesh for large deformation analysis. *Journal of Computing and Information Science in Engineering*, 5(4):338–347, 2005. 1

[DS07] Arbtip Dheeravongkit and Kenji Shimada. Inverse adaptation of a hex-dominant mesh for large deformation finite element analysis. *Computer-Aided Design*, 39(5):427 – 438, 2007. Geometric Modeling and Processing 2006. (document), 1, 1.1

[Ede] Herbert Edelsbrunner. Square tori: $-5/4$ time, 180 wrapped tubes. `http://pub.ist.ac.at/~edels/Tubes/tori/quad/n5.stl`. 4.13a

[EGM+94] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for image analysis. *Journal of Mathematical Imaging and Vision*, 4(4):353–373, 1994. 5.2.2

[FG08] Pascal Jean Frey and Paul-Louis George. *Mesh Generation: Application to*

Finite Elements. ISTE Ltd and John Wiley & Sons, Inc., London, 2nd edition, 2008. 2.1, 2.2.1, 2.2.2

[FUN11] FUN3D – Fully Unstructured Navier-Stokes. `http://fun3d.larc.nasa.gov`, 2011. 6.2.2

[Gno10] P.A. Gnoffo. Updates to multi-dimensional flux reconstruction for hypersonic simulations on tetrahedral grids. In *Proceedings of the 48th AIAA Aerospace Sciences Meeting*. American Institute for Aeronautics and Astronautics, 2010. 2.2.1, 2.3

[HLL97] L. Hesselink, Y. Levy, and Y. Lavin. The topology of symmetric, second-order 3d tensor fields. *Visualization and Computer Graphics, IEEE Transactions on*, 3(1):1–11, 1997. 3.8

[HSZ87] R.M. Haralick, S.R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 9(4):532–550, 1987. 5.2.2

[IBK+17] Daniel Ibanez, Nicolas Barral, Joshua Krakos, Adrien Loseille, Todd Michal, and Mike Park. First benchmark of the unstructured grid adaptation working group. *Procedia Engineering*, 203:154 – 166, 2017. 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain. 6.4

[Kin04] Gordon Kindlmann. Superquadric tensor glyphs. In *Proceedings of IEEE TVCGEG Symposium on Visualization*, pages 147–154, 2004. 3.5, 3.4f

[KTW07] Gordon Kindlmann, Xavier Tricoche, and Carl-Fredrik Westin. Delineating white matter structure in diffusion tensor MRI with anisotropy creases. *Medical Image Analysis*, 11(5):492–502, October 2007. 3.6

[LDV+04] P. Labbé, J. Dompierre, M.-G. Vallet, F. Guibault, and J.-Y. Trépanier. A universal measure of the conformity of a mesh with respect to an anisotropic metric

field. *International Journal for Numerical Methods in Engineering*, 61(15):2675–2695, 2004. 3.6, 3.6

[LGT99] Y. Lu, R. Gadh, and T. J. Tautges. Volume decomposition and feature recognition for hexahedral mesh generation. In *Proceedings of the 8th International Meshing Roundtable*, pages 269–280, 1999. 2.2.1

[LTU99] X.Y. Li, S.H. Teng, and A. Üngör. Biting ellipses to generate anisotropic mesh. In *Proceedings of the 8th International Meshing Roundtable*, pages 97–108, 1999. 3.2

[MTT98] R. J. Meyers, T. J. Tautges, and P. M. Tuchinsky. The "Hex-Tet" hex-dominant meshing algorithm as implemented in cubit. In *Proceedings of the 7^{th} International Meshing Roundtable*, pages 151–158, 1998. 4.3

[Ogd97] R.W. Ogden. *Non-Linear Elastic Deformations*. Dover, 1997. 2.1

[O'R98] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, Cambridge, UK, Second edition, 1998. 4.4

[Par03] M.A. Park. Three-dimensional turbulent RANS adjoint-based error correction. In *AIAA Paper*, pages 2003–3849, 2003. 2.3, 3.7.2

[PFA06] X. Pennec, P. Fillard, and N. Ayache. A riemannian framework for tensor computing. *Int. J. Comput. Vision*, 66(1):41–66, 2006. 3.4.2

[PLK+16] Michael A. Park, Adrien Loseille, Joshua Krakos, Todd R. Michal, and Juan J. Alonso. Unstructured Grid Adaptation: Status, Potential Impacts, and Recommended Investments Towards CFD 2030. In *46th AIAA Fluid Dynamics Conference*, Washington, D.C., June 2016. American Institute of Aeronautics and Astronautics. 7.1

[PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical*

Recipes in C: The Art of Scientific Computing. Cambridge Univ. Press, UK, Second edition, 1992. 4.3.2

[PZ07] J. Palacios and E. Zhang. Rotational symmetry field design on surfaces. In *Proceedings of ACM SIGGRAPH*, page 55, 2007. 3.7.1

[QVB+10] William Roshan Quadros, Ved Vyas, Mike Brewer, Steven James Owen, and Kenji Shimada. A computational framework for automating generation of sizing function in assembly meshing via disconnected skeletons. *Eng. with Comput.*, 26(3):231–247, June 2010. 7.1

[RG97] William C Regli and Daniel M Gaines. A repository for design, process planning and assembly. *Computer-Aided Design*, 29(12):895 – 905, 1997. Models available at `http://edge.cs.drexel.edu/repository/` and `http://edge.cs.drexel.edu/repository/repository/`. 6.3.1

[ROCV16] J. Robbins, S.J. Owen, B.W. Clark, and T.E. Voth. An efficient and scalable approach for generating topologically optimized cellular structures for additive manufacturing. *Additive Manufacturing*, 12:296 – 304, 2016. Special Issue on Modeling & Simulation for Additive Manufacturing. 1

[RPP+09] O. Rosanwo, C. Petz, S. Prohaska, I. Hotz, and H.-C. Hege. Dual streamline seeding. In *Proc. Pacific Vis '09*, pages 9–16, 2009. 4.4.1

[Sim08] Simulia Corp. *ABAQUS Analysis User's Manual, Version 6.8*, 2008. 6.2.1

[SKOB06] M. L. Staten, R. A. Kerr, S. J. Owen, and T. D. Blacker. Unconstrained paving and plastering: Progress update. In *Proceedings of the 15th International Meshing Roundtable*, pages 469–496, 2006. 2.2.1

[SLI98] K. Shimada, J.-H. Liao, and T. Itoh. Quadrilateral meshing with directionality control through the packing of square cells. In *Proceedings of the 7th International Meshing Roundtable*, pages 61–76, 1998. 3.2, 4.1

[SS03] R. Sundareswara and P. Schrater. Extensible point location algorithm. In *Proceedings of the 2003 International Conference on Geometric Modeling and Graphics*, pages 84–89, 2003. 3.4.1

[SSW96] R. Schneiders, R. Schindler, and F. Weiler. Octree-based generation of hexahedral element meshes. In *Proceedings of the 5th International Meshing Roundtable*, pages 205–216, 1996. 2.2.1, 4.2

[Str88] G. Strang. *Linear Algebra and Its Applications*. Brooks-Cole, Third edition, 1988. 3.2

[SW13] Gerald Jay Sussman and Jack Wisdom. *Functional differential geometry*. MIT Press, 2013. 3.2

[TAP97] John C. Tannehill, Dale A. Anderson, and Richard H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Taylor & Francis, Second edition, 1997. 1, 3.7.1

[TDVC04] K. F. Tchon, J. Dompierre, MG Vallet, and R. Camarero. Visualizing mesh adaptation metric tensors. In *Proceedings of the 13th International Meshing Roundtable*, pages 353–363, 2004. 4.3.2

[TG14] Douglas S Thomas and Stanley W Gilbert. Costs and cost effectiveness of additive manufacturing: A literature review and discussion. *NIST Special Publication*, 1176, 2014. 1

[TGDC05] K.-F. Tchon, F. Guibault, J. Dompierre, and R. Camarero. Adaptive hybrid meshing using metric tensor line networks. In *Proceedings of the 17th AIAA Computational Fluid Dynamics Conference*. American Institute for Aeronautics and Astronautics, 2005. 3.5

[TWM85] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation: Foundations and Applications*. North-Holland, 1985. 1, 2.2.1

[VCD+16] Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. Directional field synthesis, design, and processing. *Computer Graphics Forum*, 35(2):545–572, 2016. 2.1

[VS09] Ved Vyas and Kenji Shimada. Tensor-guided hex-dominant mesh generation with targeted all-hex regions. In *proceedings of the 18th International Meshing Roundtable*, pages 377–396. Springer, 2009. 4.3

[VSI00] N. Viswanath, K. Shimada, and T. Itoh. Quadrilateral meshing with anisotropy and directionality control via close packing of rectangular cells. In *Proceedings of the 9th International Meshing Roundtable*, pages 227–238, 2000. 1, 1.2, 5.2.4

[VSmI00] N. Viswanath, K. Shimada, and T. Itoh. Quadrilateral meshing with anisotropy and directionality control via close packing of rectangular cells. In *Proceedings of the 9th International Meshing Roundtable*, pages 217–225, 2000. 4.1

[WH06] Joachim Weickert and Hans Hagen, editors. *Visualization and Processing of Tensor Fields*. Springer, Berlin, 2006. 2.1

[WK97] D. White and P. Kinney. Redesign of the paving algorithm: Robustness enhancements through element by element meshing. In *Proceedings of the 6th International Meshing Roundtable*, pages 323–335, 1997. 1.2

[WWB+07] M. Welk, J. Weickert, F. Becker, C. Schnorr, C. Feddern, and B. Burgeth. Median and related local filters for tensor-valued images. *Signal processing*, 87(2):291–308, 2007. 3.9

[YS00] S. Yamakawa and K. Shimada. High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. In *Proceedings of the 9th International Meshing Roundtable*, pages 263–273, 2000. 2.2.2, 3.2, 4.1

[YS03a] S. Yamakawa and K. Shimada. Fully-automated hex-dominant mesh generation

with directionality control via packing rectangular solid cells. *International Journal for Numerical Methods in Engineering*, 57(15):2099–2129, 2003. 2.2.3, 4.3

[YS03b] S. Yamakawa and K. Shimada. Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells. *International Journal for Numerical Methods in Engineering*, 57(15):2099–2129, 2003. 4.4.3, 5.2.2

[YS08] S. Yamakawa and K. Shimada. Subdivision templates for converting a non-conformal hex-dominant mesh to a conformal hex-dominant mesh without pyramid elements. In *Proceedings of the 17th International Meshing Roundtable*, pages 497–512, 2008. 2.2.3

[ZHB07] Y. Zhang, T. J.R. Hughes, and C. L. Bajaj. Automatic 3d mesh generation for a domain with multiple materials. In *Proceedings of the 16th International Meshing Roundtable*, pages 367–386, 2007. 2.2.1

[ZPP05] X. Zheng, B.N. Parlett, and A. Pang. Topological lines in 3d tensor fields and discriminant Hessian factorization. *IEEE Trans. Vis. and Comput. Graphics*, 11(4), 2005. 3.8

[ZWL+18] Zichun Zhong, Wenping Wang, Bruno Lévy, Jing Hua, and Xiaohu Guo. Computing a high-dimensional euclidean embedding from an arbitrary smooth riemannian metric. *ACM Trans. Graph.*, 37(4):62:1–62:16, July 2018. 7.1