2009

The quasi-uniformity condition and three-dimensional geometry representation as it applies to the reproducing kernel element method

Nathaniel O. Collier
University of South Florida

The Quasi-Uniformity Condition and Three-Dimensional Geometry Representation as it Applies to the Reproducing Kernel Element Method

by

Nathaniel O. Collier

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Civil and Environmental Engineering
College of Engineering
University of South Florida

Major Professor: Daniel C. Simkins, Jr., Ph.D.
Andrés Tejada-Martinez, Ph.D.
Stanley Kranc, Ph.D., P.E.
Sudeep Sarkar, Ph.D.
David Rabson, Ph.D.

Date of Approval:
March 25, 2009

Keywords: meshing, regularity, RKEM, interpolation, surface representation

DEDICATION

To God, who blesses us with knowledge of His universe. To my parents, who have instilled in me a lifelong love of learning. To María, who has been patient with me and supported our family while I went off on this adventure. To Dan Simkins, who opened the door for me to learn much more than I knew to be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

The Quasi-Uniformity Condition and Three-Dimensional Geometry Representation as it Applies to the Reproducing Kernel Element Method

Nathaniel O. Collier

ABSTRACT

The Reproducing Kernel Element Method (RKEM) is a hybrid between finite elements and meshfree methods that provides shape functions of arbitrary order and continuity yet retains the Kronecker-δ property. To achieve these properties, the underlying mesh must meet certain regularity constraints, unique to RKEM. The aim of this dissertation is to develop a precise definition of these constraints, and a general algorithm for assessing a mesh is developed. This check is a critical step in the use of RKEM in any application.

The general checking algorithm is made more specific to apply to two-dimensional triangular meshes with circular supports and to three-dimensional tetrahedral meshes with spherical supports. The checking algorithm features the output of the uncovered regions that are used to develop a mesh-mending technique for fixing offending meshes. The specific check is used in conjunction with standard quality meshing techniques to produce meshes suitable for use with RKEM.

The RKEM quasi-uniformity definitions enable the use of RKEM in solving Galerkin weak forms as well as in general interpolation applications, such as the representation of geometries. A procedure for determining a RKEM representation of discrete

point sets is presented with results for surfaces in three-dimensions. This capability is important to the analysis of geometries such as patient-specific organs or other biological objects.

CHAPTER 1

INTRODUCTION

The goal of research in Computational Mechanics is to develop methods to aid in the engineering design process. The cost and duration of this process can be greatly reduced if a design can first be analyzed numerically before being physically constructed and tested. The Finite Element Method (FEM) has long been the dominant computational method used and with good reason. However, FEM has its limitations and therefore other methods have evolved to address its weaknesses.

One weakness of FEM is that it is difficult to construct higher order interpolants, which limits the class of problems FEM can solve. Usually FEM is limited to solving differential equations of first or second order. While FEM can be used to solve problems of this class, another drawback is when a quantity is desired which comes from derivatives of the FEM solution. Since FEM interpolations are C^0 continuous, the derivatives are discontinuous. To obtain a smoothed quantity based on the derivative of the FEM solution, other techniques must be used to soften these discontinuities.

Many other methods have emerged (meshfree methods, spectral methods, boundary element methods) which are suited for higher order problems, yet this frequently comes at the cost of other desirable function properties that are common in FEM. Specifically, the Kronecker-δ property is of interest for the enforcement of essential boundary conditions. The Reproducing Kernel Element Method (RKEM) was developed to achieve a higher order global smoothness property while maintaining the Kronecker-δ property, details of which will be presented in a later section.

The Reproducing Kernel Element Method is similar to the Finite Element Method in that they both use elements, polygonal or polyhedral subdivisions of the problem domain, to construct their interpolation basis. The collection of these elements is referred to as a mesh. In both cases, the mesh must have certain regularity properties to assure a proper result. These regularity constraints are known as the quasi-uniformity condition.

This dissertation is devoted to the study of the quasi-uniformity mesh condition as it applies to RKEM. While this condition was previously identified, it was not completely understood. In Chapter 2, this work will present precise definitions which form the basis for a checking algorithm. This algorithm is used to check meshes in both two- and three-dimensions.

Since the main application of RKEM is in the solution of partial differential equations, the solution procedure is presented here as is commonly seen in texts. This procedure, known as the Galerkin procedure, and the methods used to approximate solutions, are intimately tied to the regularity constraints on meshes. Following the presentation on methods is a discussion about mesh regularity and its motivation.

1.1 Galerkin Procedure for Solving Partial Differential Equations

In physical problems, the equations are typically developed in a strong form. This means that the equation holds true at every point in the problem's domain. The finite element method and its descendants all attempt to approximate solutions to the weak, or variational, form of partial differential equations. The form is 'weak' in the sense that the solution is globally valid, but may not satisfy the strong form at every point. The weak form can be derived from the strong form, as shown in the

following boundary value problem.

$$\nabla^2 u(x) = \rho(x) \quad \forall\ x \in \Omega \tag{1}$$

$$u(x) = g(x) \quad \forall\ x \in \Gamma_g \text{ (Essential BC)} \tag{2}$$

where Ω is the problem domain, $\rho(x)$ is some forcing function, and $g(x)$ is essential boundary condition which is applied on Γ_g, the portion of the boundary on which essential boundary conditions are enforced. This strong form can be converted to the weak form by multiplying by a test function, $\delta u(x)$, and then integrating over the problem domain, Ω.

$$\int_\Omega \nabla^2 u \delta u d\Omega = \int_\Omega \rho(x) \delta u d\Omega \tag{3}$$

In the case of this problem, integration by parts (Eq. 2.2.12 in [17]) can be used directly to simplify this to

$$-\int_\Omega (\nabla u) \cdot (\nabla \delta u) d\Omega = \int_\Omega \rho(x) \delta u d\Omega. \tag{4}$$

As shown in [17], under certain restrictions the strong and weak forms are equivalent. Now the solution variable and the test function are expanded into an approximation consisting of basis functions and weights.

$$u(x) = \sum_i N_i(x) u_i \qquad \delta u(x) = \sum_i N_i(x) \delta u_i \tag{5}$$

where $N_i(x)$ is the i^{th} basis function and u_i and δu_i are the corresponding constant weights. It is often more convenient to recast this summation as a dot product of vectors.

$$u(x) = \mathbf{N}\mathbf{u} \qquad \delta u(x) = \mathbf{N}\delta\mathbf{u} \tag{6}$$

The gradients of these functions can be approximated in the same way. It is customary to define

$$\mathbf{B} = \nabla \mathbf{N} \tag{7}$$

which leads to

$$\nabla u(x) = \mathbf{B}\mathbf{u} \quad \nabla \delta u(x) = \mathbf{B}\delta\mathbf{u}. \tag{8}$$

These functions are then inserted into the weak form. This is known as the Galerkin procedure. Here this solution function and test function are expanded using the same set of basis functions. This is called Bubnov-Galerkin, while using a different basis for each is known as Petrov-Galerkin. Finite element solutions are a finite dimensional projection of an infinitely dimensioned function, and therefore the FEM solution is not capable of representing the exact solution. The motivation for using a different basis to represent the test function is in problems such as advection/diffusion where an enriched basis for the test function space can account for the missing information and stabilize the solution.

$$-\int_\Omega (\mathbf{B}\mathbf{u}) \cdot (\mathbf{B}\delta\mathbf{u}) d\Omega = \int_\Omega \rho(x)\mathbf{N}\delta\mathbf{u} d\Omega \tag{9}$$

However, $\delta\mathbf{u}$ and $\mathbf{u}$ are constants and can be factored out of the integrals.

$$\delta\mathbf{u}\left(-\int_\Omega \mathbf{B}^T\mathbf{B} d\Omega \mathbf{u} + \int_\Omega \rho\mathbf{N}^T d\Omega\right) = 0 \tag{10}$$

Because Eq. 10 must be true for any $\delta\mathbf{u}$,

$$-\int_\Omega \mathbf{B}^T\mathbf{B} d\Omega \mathbf{u} + \int_\Omega \rho\mathbf{N}^T d\Omega = 0 \tag{11}$$

or

$$\mathbf{K}\mathbf{u} = \mathbf{P} \tag{12}$$

where

$$\mathbf{K} = \int_{\Omega} \mathbf{B}^T \mathbf{B} d\Omega \tag{13}$$

$$\mathbf{P} = \int_{\Omega} \rho \mathbf{N}^T d\Omega. \tag{14}$$

The matrix $\mathbf{K}$ is called the stiffness matrix and the vector $\mathbf{P}$ is called the load vector. Although this material is frequently first studied in the context of finite elements, there is nothing in this procedure particular to finite elements. The procedure is independent of how the basis functions, $N_i(x)$, are constructed. It is the different methods of constructing these functions that lead to different methods, all of which have particular strengths and weaknesses.

1.2 Methods of Constructing Interpolants

The following section provides a brief introduction to three different methods of constructing interpolants. The treatment is only meant to introduce the concept, giving some intuition to the basis function formulation. For full details, consult the given references.

1.2.1 Finite Elements

Finite elements [17, 38] is one way to construct a set of basis functions for use in the solution of the Galerkin weak form. The idea is to discretize the problem domain by tessellating the space into polygons, usually of uniform type. In finite elements, a collection of these polygons is referred to as a mesh and the polygon vertices are referred to as nodes.

The basis functions are then written explicitly for each individual element. When the basis functions are written for an element, they are referred to as shape functions. The shape functions are typically written using Lagrange interpolation, for which a

sample one-dimensional element and its shape functions can be seen in Fig. 1(a). These functions

1. form a partition of unity, essential for solution convergence in the Galerkin weak form

2. possess the Kronecker-δ property at element nodes, useful in enforcing essential boundary conditions

3. are polynomials, particularly useful when selecting an integration rule for the computation of the stiffness matrix and load vectors.

This method of formulation is also useful because one-dimensional functions can be directly used to form higher dimensional basis functions. The use of elements also has a computational advantage when forming the stiffness matrix. The stiffness matrix computation (Eq. 13) involves an integration of the product of basis function derivatives over the domain. Since each function is defined on elements, the domain of the product of these basis function derivatives is simple to obtain. This means that the numeric computation of this integration is faster because the exact integration domain does not need to be determined via other, more computationally expensive, methods.

1.2.2 Meshfree Methods

While elements are a convenient form of constructing basis functions, they are not the only method. A class of methods known as meshfree methods have emerged which do not use a mesh to define basis functions. There are many flavors of meshfree methods [1, 4, 5, 12, 21, 24], but many descend from Smoothed Particle Hydrodynamics [15] (SPH). For a review of these methods see [20]. In SPH, the approximation of the solution variable is represented as a convolution of the unknown variable and

a weighting function.

$$\bar{u}(x) = \int u(y)w(x-y)dy \tag{15}$$

Note that this expression is true if $w(x)$ is the Dirac-δ function. The weighting function is chosen to mimic the Dirac-δ function and also has the property

$$\int w(x)dx = 1. \tag{16}$$

The goal of this method is the same as in finite elements: to discretize the unknown solution variable which will allow the conversion of the weak form into a system of linear equations. Equation 15 does not yet do that. This convolution is then approximated by nodal integration,

$$\bar{u}(x) = \sum_{i=1}^{n} u(y_i)w(x-y_i)\Delta V_i \tag{17}$$

where n represents a fixed number of discrete points and ΔV_i is a scalar representation of the portion of the domain represented by that point. The variable y_i represents the location of these points in the domain. In meshfree methods, these points are referred to as particles. The nodal integration leads to the loss of the partition of unity property, specifically,

$$\sum_{i=1}^{n} w(x-y_i)\Delta V_i \neq 1. \tag{18}$$

Many corrections have been proposed, which has led to a large number of methods. One method of correcting this interpolation is found in the Reproducing Kernel Particle Method [21, 23] (RKPM) which is included here for its relationship to RKEM. The idea is to modify the weight function in such a way as to maintain the partition of unity property. This is done by multiplying the weight function by a polynomial of arbitrary order. For illustrative purposes, a one-dimensional polynomial of second

order is shown here.

$$P(x) = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \mathbf{b} \tag{19}$$

where $\mathbf{b}$ is a vector of unknown polynomial coefficients. This polynomial becomes part of a function termed the kernel, expressed here:

$$\phi_i(x) = w(x - y_i)P(x - y_i)\Delta V_i. \tag{20}$$

The coefficients in $\mathbf{b}$ are determined by solving a linear system of equations that ensure that the kernel can reproduce constants, restoring its partition of unity. This system also assures that polynomials up to the order of P can be reproduced. So the complete RKPM interpolation ends as

$$\sum_i \phi_i(x)u_i. \tag{21}$$

A consequence of this formulation is that RKPM basis functions cannot be easily written explicitly and require a matrix inversion at every point they are evaluated. While this is computationally expensive, the trade off is a set of basis functions that are smooth to the order of the weighting function continuity and can globally reproduce any polynomial up to the degree of the corrective polynomial. These functions do not, however, possess the Kronecker-δ property, so the enforcement of essential boundary conditions requires an alternate procedure.

While the formulation of RKPM basis functions is not as intuitive as FEM basis functions, it is helpful to conceptualize it in the following way. The RKPM basis functions are particle-centered polynomials that are blended from particle to particle by convolution with a weighting function. Graphically this can been seen in Fig. 1(b).

1.2.3 Reproducing Kernel Element Method

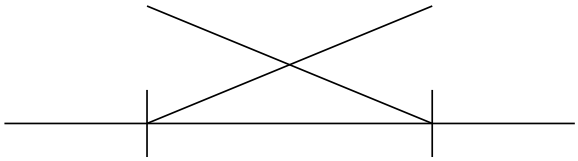
The Reproducing Kernel Element Method is a hybrid mesh/meshfree method whose main aim is to take advantage of the higher order smoothness of meshfree basis functions while preserving the Kronecker-δ property common to finite element basis functions. This method was originally discussed in a series of papers [22, 25, 27, 32]. In the first of these papers, the authors write the RKEM interpolant and then proceed to prove its properties. The RKEM interpolant is written

$$\bar{u}(\mathbf{x}) = \sum_{e=1}^{N_e} \left[\int_{\Omega_e} \mathcal{K}(\mathbf{x}-\mathbf{y}) d\mathbf{y} \left(\sum_{i=1}^{n_e} \psi_{e,i}(\mathbf{x}) u(\mathbf{x}) \right) \right] \tag{22}$$

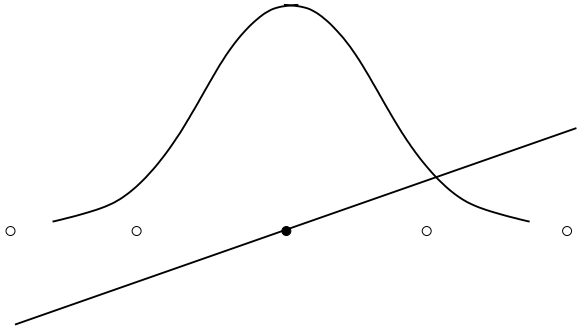
where N_e are the number of elements in the mesh, n_e are the number of nodes per element, $\mathcal{K}$ is the meshfree kernel, and ψ are the global partition polynomials, analogous to the corrective polynomial in the meshfree formulation.

The RKEM interpolant shares concepts from both FEM and meshfree methods. Finite element basis functions are written explicitly in such a way that C^0 continuity is maintained, referred to as compatibility. Even for higher order finite element functions, the continuity from element to element is still C^0. Meshfree methods, such as RKPM, enforce compatibility by smoothing polynomial fields from particle to particle by kernels, but at the cost of the Kronecker-δ property. Reproducing Kernel Element Method interpolants enforce compatibility of element defined polynomials with node centered kernels. The element's polynomial fields are no longer truncated as in finite elements, but allowed to extend the domain of the problem. This can be graphically seen in Fig. 1(c).

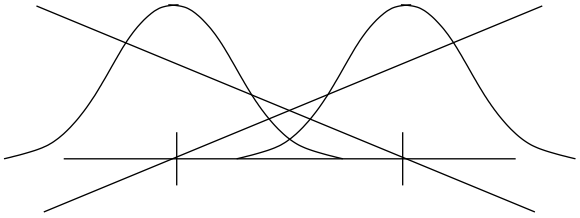
The RKEM interpolant looks complicated but it can be split into its different parts. The right side, $\sum_{i=1}^{n_e} \psi_{e,i}(\mathbf{x}) u(\mathbf{x})$, is the finite-element part, where the global partition polynomials can be thought of as extended FEM basis functions. The left



(a) FEM shape functions



(b) RKPM polynomial field and kernel



(c) RKEM polynomial fields and node-centered kernels

Figure 1. Conceptual differences in the forming of basis functions

side is the meshfree part, $\int_{\Omega_e} \mathcal{K}(\mathbf{x}-\mathbf{y})d\mathbf{y}$, which is the smoothing by node-centered kernel functions.

A benefit of RKEM is that higher-order interpolants may be constructed trivially by adding degrees of freedom to element nodes and accompanying polynomial fields. While higher-order finite elements may be constructed in the same manner, this is accomplished with the addition of mid-side nodes which complicates the meshing problem. Furthermore, while higher-order finite elements are possible, the continuity across element boundaries remains C^0. In RKEM, the compatibility of these higher-order polynomial fields is enforced by the node-centered kernels. This allows for the construction of globally smooth, higher-degree elements without the introduction of mid-side nodes.

In FEM, the impetus for element shapes other than triangle or tetrahedron is to enrich the interpolation field without the addition of mid-side or interior nodes. However, elements such as quadrilaterals and hexahedra increase the difficulty of meshing. RKEM has the advantage of increased interpolatory power using only vertex nodes on the triangle and tetrahedron. Thus, this work will primarily focus on triangular and tetrahedral elements. This choice is not requisite, as the formulation of RKEM quasi-uniformity applies to any element shape.

1.3 Mesh Regularity

The quasi-uniformity condition refers to a regularity constraint on the size and shape of the mesh elements. Mesh regularity is not unique to RKEM: finite elements also requires that the mesh meet regularity conditions. This section will detail what quasi-uniformity is in the context of FEM and compare and contrast it to that of RKEM.

1.3.1 FEM Quasi-Uniformity

In [17], finite-element convergence is shown to depend on the maximum element characteristic length, leading to the desire to restrict element aspect ratios. With suitable constraints on aspect ratios, mesh refinement will guarantee diminishing maximum characteristic element size and lead to convergence. This condition is termed *quasi-uniformity*. In the context of RKEM, the use of the term quasi-uniformity is somewhat different. To avoid ambiguity, the term *RKEM quasi-uniformity* is used.

Finite element quasi-uniformity is easily verified by direct inspection of the aspect ratios of the elements. This metric, or an equivalent, is included in meshing algorithms, like those outlined in [28], which are designed for a variety of element geometries.

1.3.2 RKEM Quasi-Uniformity

As a hybrid mesh/meshfree method, RKEM uses elements to define its local interpolation field and nodal kernels to enforce compatibility. Support sizes for the kernels must be chosen large enough for each node such that the entire mesh domain is covered by the support domain of at least one node. Failure to do so results in the loss of the partition of unity property. On the other hand, the support of one node cannot include any other node; otherwise the Kronecker-δ property will be lost. These properties lead to regularity conditions on RKEM meshes first described in [22]. This is the main topic of this dissertation and will be further discussed in chapter 2.

1.3.3 Main Difference

The main practical difference between FEM and RKEM quasi-uniformity is that in RKEM, failure to meet the condition in Def. 2.1.3 prevents basis functions from being defined in part of the domain, rendering an incomplete function space; on the domain. In contrast, the FEM quasi-uniformity condition does not impact the

ability to define the function space, it enters in the size of the interpolation error. Even though the term's usage is describing a different phenomenon, the RKEM mesh constraints do require the mesh to be somewhat uniform as in finite elements. Yet the concept cannot be condensed into merely assuring low aspect ratios. Therefore the term *quasi-uniform* aptly applies.

Note that RKEM meshes must also be quasi-uniform in the FEM sense when used to approximate the solution to a Galerkin weak form. As previously mentioned, RKEM quasi-uniformity ensures basis function properties, while FEM quasi-uniformity relates to the assurance of error bounds. Theorem 4.1 in [25] states that, similar to that of FEM, the RKEM error estimate is bound to the characteristic length. This means that in refinement, RKEM will have the same accuracy limitations that FEM does when elements of high aspect ratios are present. For this reason, RKEM meshes must be quasi-uniform in *both* senses.

While the effect of this interpolation error on analysis is an important topic, this work is concerned with detailing the condition necessary for RKEM interpolant construction. This is critical for the use of RKEM in applications where it is used for other interpolation tasks, such as geometry representation [34], the primary topic of Chapter 3.

1.3.4 Quality Mesh Generation

The way that FEM obtains meshes that are quasi-uniform is to design an algorithm that guarantees their generation. This is known in the literature as quality mesh generation. Three conceptual steps exist in mesh generation. First is the construction of elements given a point set without regard to quality. Second is the addition of points (Steiner points) to the input point set to allow the creation of better-quality elements. The last step is movement of nodes and topological changes to improve a given mesh; this is generally known as smoothing.

In triangular meshes, the Delaunay criterion [36] is used to obtain a unique triangularization from any non-trivial data set. The Delaunay criterion does not itself guarantee regularity but is a way to construct the elements without regard to quality. Typically a set of boundary nodes is first triangulated using this technique, and then points are inserted to achieve some measure of element regularity.

It is the methods of inserting points that differentiates one triangular mesh generation algorithm from another. One such technique inserts points at the circumcircle centers as shown by [6]. This technique further constrains the order of insertion and guarantees elements with a bound on the minimum angle, which can be directly related to the element aspect ratio. Another popular technique is to use an advancing front. A row of elements is first created at the boundary by inserting points needed to generate only boundary elements. This procedure is repeated progressively inward, either connecting existing nodes or generating a new location that will produce a quality element. Tetrahedral meshing follows these concepts with similar philosophy.

The previously mentioned quality mesh generation techniques all involve the insertion of nodes and the subsequent changes in element topology. Smoothing techniques focus more on the nodal locations and topology of the mesh. Laplacian smoothing [13] is a popular technique, which relocates a node to the centroid of its neighboring nodes. Recent work [35] presents a method for smoothing the mesh by linearly transforming the element from its initial state to an equilateral triangle. This affects multiple nodes simultaneously and is iteratively applied while aspect ratios and/or minimum angle criteria are maintained. None of these methods addresses key issues in assessing RKEM quasi-uniform meshes.

1.3.5 Meshfree Support Issues

Since the RKEM quasi-uniform condition is not solely a property of the elements and mesh but also of the supports of the kernels associated with each node, RKEM

inherits some of these issues from meshfree methods, which will now be reviewed. In [11] the technique for determining appropriate support sizes involves including a sufficient number of particles so that the moment matrix used in computing basis functions is non-singular. While this is true, there are more complex cases in which a sufficient number of particles can be included, yet the resulting functions are biased due to particle distribution. These situations are briefly discussed in [19] and shown in Fig. 2, adapted from their work. Note that this, too, differs from RKEM. In

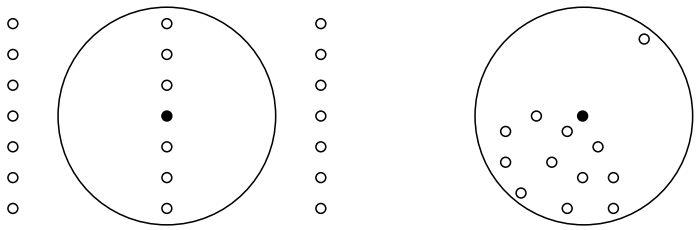


Figure 2. Biased yet admissible meshfree supports (adapted from [19])

meshfree methods such as the Reproducing Kernel Particle Method (RKPM), both a support window and a polynomial field are centered at the particle. Supports must contain other particles to form a partition of unity. In RKEM, the polynomial fields are defined on elements and the support window is centered on the nodes. The support window serves to smooth the polynomial functions from element to element. Therefore in RKEM support size is critical, but not in the same sense as in meshfree. This is what requires every element domain to be covered by at least one support. While FEM, RKEM, and meshfree methods all share difficulty in determining element and/or support sizes, none of the techniques outlined here addresses the issues that are unique to RKEM.

1.4 Geometry Representation

While the original application of RKEM was in the solution of partial differential equations, the RKEM basis functions can be used for general interpolation, such as

in the representation of geometries. Chapter 3 will discuss a procedure for the use of RKEM in representing three-dimensional surfaces.

CHAPTER 2

THE RKEM QUASI-UNIFORMITY CONDITION

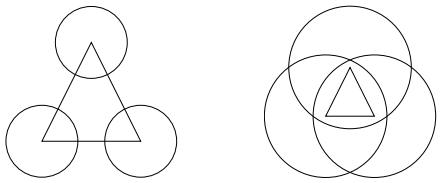
There are issues in generating RKEM quasi-uniform meshes that current quality mesh generation algorithms are not suited to solve. Again it is important to emphasize that for use in analysis, the mesh must be quasi-uniform in the FEM sense, due to the convergence of RKEM being, as in FEM, tied to the characteristic element length. This requirement can be satisfied using the existing quality meshing algorithms and will not be discussed further in this work.

There are two additional concepts important to RKEM quasi-uniformity. First, the node-centered kernels used to blend the polynomial fields from one element to another must collectively cover the entire mesh domain. If these supports fail to entirely cover the mesh, the RKEM basis functions will not have the partition of unity property. More importantly, the basis functions will be zero in regions uncovered by a kernel. This situation can be seen in Fig. 3(a).

Second, the supports of the kernels can be so large that they include other nodes. This also corresponds to a loss of a RKEM basis function property, the Kronecker-δ property, depicted in Fig. 3(b). These concepts imply that a balance of support sizes must be struck as shown in Fig. 3(c).

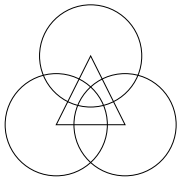
2.1 Definitions

The following definitions concisely state the RKEM quasi-uniform condition and form the basis to check a mesh for quasi-uniformity. While this work focuses on two-dimensional triangular meshes with circular supports, these definitions are in-



(a) Too small, loss of partition of unity

(b) Too large, loss of Kronecker-δ



(c) Just right, preserves both properties

Figure 3. Quasi-uniformity for a RKEM element

dependent of spatial dimension, element geometry, or kernel support shape. The first of these definitions address the isolation concept, which ensures the Kronecker-δ property.

1. Definition 2.1.1 (Nodal Isolation) A node is *isolated* if it is not in the support of any other node.

2. Definition 2.1.2 (Element Coverage) Let E_i represent the domain of an element $i \in N_e$, where N_e is the index set of elements contained in a mesh, M. Let C_j represent the domain of the support whose center corresponds to a node j. Denote by N_I the index set of all node indices of mesh M. An element, E_i, is *covered* if

$$\bigcup_{j \in N_I} C_j \cap E_i = E_i.$$

3. Definition 2.1.3 (Mesh Coverage) A mesh M is *covered* if all its elements are covered:

$$\bigcup_{i \in N_e} \left\{ \bigcup_{j \in N_i} C_j \cap E_i \right\} = M.$$

4. Definition 2.1.4 (RKEM Quasi-Uniformity) A mesh M is said to be *quasi-uniform* in the context of RKEM if

   (a) All mesh nodes are isolated (Def. 2.1.1)

   (b) M is covered (Def. 2.1.3).

The first condition in Def. 2.1.4 guarantees the Kronecker-δ property is met at all nodes, and the second condition ensures the partition of unity, and consequently the reproducing properties of RKEM basis functions everywhere in the mesh M.

Though first noted in [25], it is worth re-emphasizing that nodal isolation, Def. 2.1.1, is not required for RKEM basis functions to be used for general function approximation, or in Galerkin weak forms. However it is beneficial in applying essential boundary conditions in the latter case, and makes the RKEM basis functions interpolating, in the mathematical sense, in the former situation.

2.2 Previous Understanding

In previous work, [25], the mesh regularity condition was specified as some constant times the diameter of an element. Specifically, in [32], the mesh regularity condition for triangular meshes with circular supports was expressed as

$$\frac{1}{2} \max_{j \in \Lambda_i} d_{ij} \leq \rho_i \leq \min_{j \in \Lambda_i} d_{ij} \tag{23}$$

where ρ_i is the radius of support of node i, Λ_i is the set of all node indices sharing an edge with node i, and d_{ij} is the distance from node i to node j. To ensure the Kronecker-δ property, it suffices to require that

$$\rho_i = f \min_{j \in \Lambda_i} d_{ij} \tag{24}$$

where $0.5 \leq f < 1$ is the scale factor. Then Eq. 23 may be restated as

$$\rho_i \geq \frac{1}{2} \max_{j \in \Lambda_i} d_{ij} \tag{25}$$

The use of a scale factor is a convenient way to set all the support radii as some fraction of their known maximum value. Furthermore, requiring $f < 1$ ensures that the Kronecker-δ property is maintained because no other node will be in another node's support. This definition is not satisfactory as can be seen in the following example.

Consider the case of a single element mesh consisting of an equilateral triangle as shown in Fig. 4. According to Eqs. 24 and 25 it is valid to make $f = 0.5$. This would make

$$\rho_i = \frac{1}{2} d_{ij} \tag{26}$$

which will be constant for all nodes since all sides are equal length in this example. The condition in Eq. 25 is satisfied; however, as depicted in Fig. 4, the condition that every point lie within the kernel support of at least one node is not satisfied. Thus, the condition, Eq. 23 or equivalently Eq. 25, is not sufficient to guarantee coverage.

Past work may have avoided this issue in one or more of the following ways.

1. The condition expressed in Eq. 23 is sufficient for one-dimensional work.

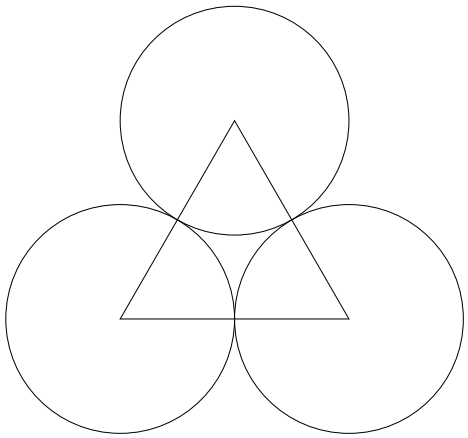


Figure 4. Element passes Eq. 25 but the supports fail to cover the entire mesh

2. While the case shown in Fig. 4 demonstrates a weakness in the condition, it is an extreme case. If a large scale factor, f, is chosen, it is possible that a mesh passes the coverage criteria.

3. If the Kronecker-δ property is not needed, then the scale factor can be set ($f > 1$) such that the supports are large enough to completely cover the domain.

4. In cases where RKEM are used in a Galerkin weak form, it is possible to avoid this issue as long as the Gauss points used in integration are covered by the support of a node's kernel.

2.3 Fundamental Difficulty

While the RKEM quasi-uniformity condition is simple in concept, it is not so simple to check. Quasi-uniformity is a global property of the mesh, not a local one in the sense that it cannot be ascertained from considering an isolated element and its nodes. Unless a node lies on the boundary it always belongs to more than one element. Since a node's support cannot cover another node, the nodes of its neighboring elements must lie outside the support. For a single element, it may be possible to set the support sizes of its nodes to achieve coverage, but the diameters of neighboring elements sharing the nodes may restrict the support size in order to

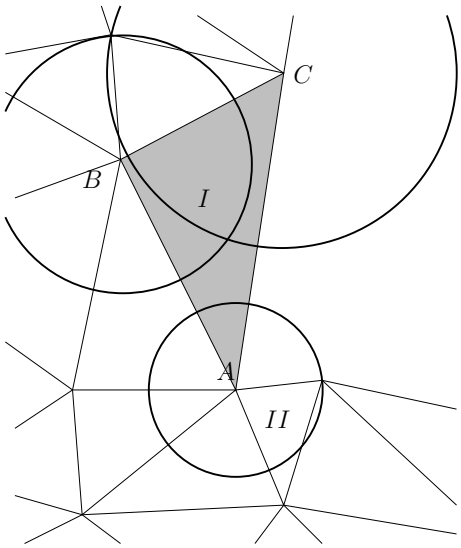


Figure 5. The proximity of smaller element II to element I causes a failure of quasi-uniformity

achieve nodal isolation. For example, consider the portion of a mesh depicted in Fig. 5. The supports of the element's nodes are drawn as circles and clearly do not cover the entire element. If the node A of element I could have its support increased, the element could be covered. However, this node has its support limited by a node in element II. To increase the support further would mean the loss of the Kronecker-δ property. The RKEM quasi-uniformity condition states that the mesh domain must be completely covered by at least one support kernel from any node. In the context of a single element, this does not imply that the element's nodes alone must provide the coverage. Consider the mesh shown in Fig. 6. Note that if only the supports of the nodes of element I are considered, the element fails to be completely covered, as shown with cross hatching. However, when the supports of nodes on the neighboring element II are also included, element I is covered because the large support of node D sufficiently covers the area of element I that the kernels of its own nodes did not. The previous expression of quasi-uniformity (Eq. 23) was not able to capture this effect, and as a result was overly restrictive by attempting to assure coverage using only the vertex nodes of each element.

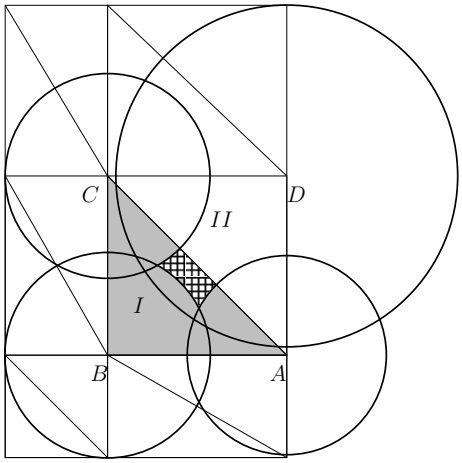


Figure 6. The support of element II covers the otherwise deficient element I

2.4 Support Shapes

While this dissertation focuses on circular supports, Def. 2.1.4 is independent of support shape. Any support shape may be used provided that a window function may be constructed with the required properties [25].

Rectangular window functions are a frequent choice, primarily because they can be obtained by a tensor product of a one-dimensional window function. In meshfree methods they are also advantageous because the integration domain can be aligned with the background integration cells. Here, the coverage problem is not aided by the rectangular shape of the support. For example, consider the triangular mesh shown in Fig. 7 and the node labeled A. Determining the maximum possible support is not a unique process. If one first determines the maximum horizontal dimension and then the vertical, the rectangular support labeled I is obtained. If this is reversed, then the support labeled II is obtained. Such ambiguities can lead to undesirable bias in the resulting basis functions.

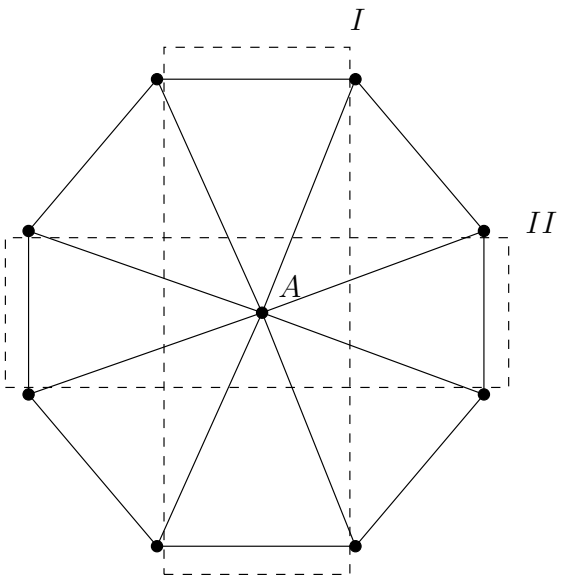


Figure 7. Two possible maximum rectangular supports

2.5 Algorithm Development

Existing mesh generation addresses the quality constraints for interpolation error but does not address Def. 2.1.4. This section introduces an algorithm to check mesh coverage while preserving nodal isolation.

2.5.1 General Algorithm

The definitions in §2.1 are amenable to direct implementation into an algorithm for checking RKEM quasi-uniformity. There are two parts to the general algorithm. The first part ensures the Kronecker-δ property by computing the nearest-neighbor distance for each node and an application of a scale factor, $f < 1$ ensures that Def. 2.1.1 is satisfied. The second part checks that the resulting supports cover the domain using Def. 2.1.3.

2.5.2 Nodal Isolation

Technically, this step is not a check, but rather a computation to determine support sizes for each node, a process guaranteed to succeed as long as there are no

coincident nodes. Conceptually, one merely computes all pair-wise distances and for each node retains the minimum. In general this is an $\mathcal{O}(N^2)$ algorithm, where N is the number of nodes in the mesh, though it is likely that algorithms similar to those described in [8] could improve on this. While it might be tempting to use the mesh to reduce the number of nodes examined, for example examining only nodes that share an element edge with the node being tested, this does not work as shown in Fig. 8. Here the node labeled A is closest to node B even though node B is separated from node A by several elements. Another mesh may have an arbitrary number of elements separating nodes A and B. This reflects the fact that the connectivity of the mesh does not contain any distance information that can be used to simplify the search. One motivation to require the Kronecker-δ property only for a subset of nodes

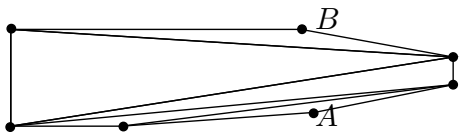


Figure 8. Edge and elements do not help find closest nodes

is that the complexity reduces to $\mathcal{O}(N_\delta \cdot N)$, where N_δ is the number of nodes with the Kronecker-δ property.

2.5.3 Coverage

The second part of the general algorithm tests for coverage. In this case, since the mesh already consists of disjoint elements, the coverage test can be performed on an element-by-element basis. This may be beneficial for at least two reasons. First, this is a so-called embarrassingly parallel computation and can trivially benefit from multiple processors. Secondly, as exploited in §2.5.4, the computational geometry may be easier. The element coverage test will generally involve any number of nodes, and ones that are not necessarily located at the element vertices or on adjacent elements, as shown in Fig. 8. However such cases are not good elements from the

FEM quasi-uniform point of view either, and are generally avoided, if possible. Thus, in practice, element coverage will often be achieved using only vertex nodes of the current element, or those of adjacent elements. To improve efficiency, a three-step process is used, noting that if element coverage is achieved at any step, the remaining steps are unnecessary.

1. Restrict the set N_i to be the nodes at the vertices of the element and check Def. 2.1.2.

2. Expand the set N_i in step 1 to include the vertices of elements adjacent to the element being tested. Re-check Def. 2.1.2.

3. Expand the set N_i to include all nodes in the mesh and re-check Def. 2.1.2.

As long as each element is covered by its own nodes or nodes of its neighboring elements, the cost of this portion of the algorithm will be $\mathcal{O}(n)$. The algorithm becomes significantly more expensive if all nodes must be included in coverage test. For meshes consisting of regular elements, such as the quality meshes that typical algorithms (such as those in §1.3.4) generate, the first two steps are sufficient.

2.5.4 Two-dimensional, Circular Support Algorithm

For the special case of triangular meshes and circular supports, the algorithm presented in §2.5.1 can be specialized and is particularly simple. The nodal supports are computed as in the general case, but the element coverage algorithm is simplified. The concept is to start with a triangular element and sequentially cut away the the portion of the element that is covered by the support of each of the nodes in the index set N_i. Each region is bounded by an ordered set of edges that are either line segments, or circular arcs. The intersection of a circular support with the region then consists of finding the intersection of a circle with line segments and/or circular arcs. The portion of the region covered by the support is then trimmed away yielding a

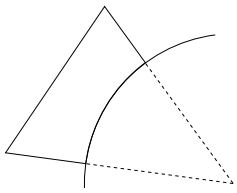


Figure 9. Triangle trimmed by circular support

smaller region that is not covered. This process continues until the element is covered, or all the nodes have been exhausted.

2.5.5 Circular Support Trimming Algorithm

The following steps were used to trim triangular elements with circular supports.

1. Find all intersections of the circular support with segments
2. Categorize each segment into 1 of 3 categories
   (a) *No Support Intersections.* If the edge endpoints are both inside the support, then the edge is removed. If both are outside, then the edge is left unaltered.
   (b) *1 Support Intersection.* In this case, one of the segment end points will be covered by the support and is trimmed to the intersection point. Note that although it is possible to have the case where an edge is tangent to the support, and hence no end point is inside the support. This possibility is detected and the segment is split at the tangent point. Figure 9 shows a typical first trimming operation on a triangle.
   (c) *2 Support Intersections.* This case results in two separated line segments as shown in Fig. 10 and shows a single region being divided into two regions.
3. Once a region has been trimmed, circular arcs that are portions of the support boundary are inserted into the regions to maintain the list of ordered segments.

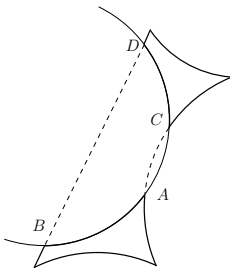


Figure 10. Arbitrary region trimmed by circular support

Figure 10 shows such an arc between points A and B, and also between points C and D.

The first step of this process involves finding the intersections of the segments defining the boundary of the untrimmed area and the circular support. Initially, this will involve circle-line intersections only, but after some supports are trimmed away from the element area, there will be intersections with circular arcs as well. These intersections are found by first assuming the circular arc is a complete circle and performing a circle-circle intersection. The circle-circle intersection algorithm used is depicted in Fig. 11. The known quantities are the radii of the circles, r_1, r_2, as well as the center-to-center distance, l. There are two intersections symmetric about, and located perpendicular to, the center-to-center line segment at a distance d. Let the distances from each center to the common secant formed by the intersections be denoted l_1 and l_2. Then the unknown distances l_1, l_2, and d may be found by

$$l_2 = \frac{r_2^2 - r_1^2 + l^2}{2l} \tag{27}$$

$$l_1 = l - l_2 \tag{28}$$

$$d = \sqrt{r_2^2 - l_2^2} \tag{29}$$

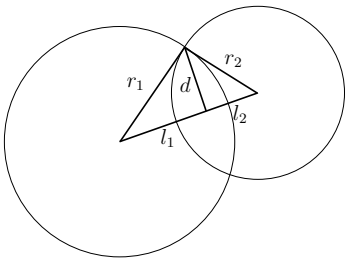


Figure 11. Circle-circle intersection

As each element is examined, a list of segments is maintained representing uncovered areas. If any region remains after the coverage algorithm has completed, then the mesh is not covered. If the mesh is not covered, the algorithm returns a list of regions that remain. This list is useful in a mesh mending procedure, as described in §2.7.

2.5.6 Three-dimensional, Spherical Support Algorithm

For the special case of tetrahedral meshes with spherical supports, the general algorithm in §2.5.1 can be specialized. This case is logically similar to the two-dimensional counterpart described in §2.5.4 but a dimension higher and more complex.

As before the nodal supports are computed to exclude all other nodes and then the element coverage is computed by removing the spherical intersections from the tetrahedral volumes. The tetrahedron consists of a list of surfaces, segments and vertices. Each surface also keeps track of the segments which consist of its boundary. If the surface is spherical, then the center and radius is also stored. As in the circular algorithm a segment can either be a straight line or a circular curve. In the three-dimensional case, however, the circular curves also contain a normal vector which indicates the plane in which the curve is defined. This normal in conjunction with the end points and radius makes the curve unique in three-dimensional space.

2.5.7 Spherical Support Trimming Algorithm

The following steps were followed to trim tetrahedral elements with spheres.

1. Find all intersections of the spherical support and the boundary edges of the tetrahedra as shown in Fig. 12(a).

2. Categorize each segment exactly as done in the circular-support algorithm and trim the qualifying segments. Note that while this step is equivalent to that of the circular support, the intersection algorithms must be general enough to work on two- and three- dimensional data. This trimming is shown in Fig. 12(b).

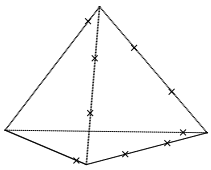
3. Once all segments have been trimmed, examine the solid surface by surface. For each surface, insert arcs again to maintain an ordered list of segments. These segments are shown connected in Fig. 12(c).

4. If a segment was split by the support, what was one logical surface will become two surfaces. Find these and create new surfaces.

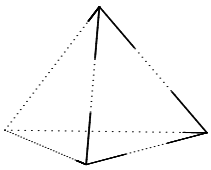
5. If for a given surface there were no segment intersections, the surface still could be intersected by the spherical support. If the plane of the surface intersects the spherical support yet no segment intersections are found, add a circular bounding curve that lies in the plane of intersection. This case is shown in Fig. 12(d).

6. Once all the trimmed segments on all surfaces are connected, take the added segments and create new spherical surfaces from connected combinations.

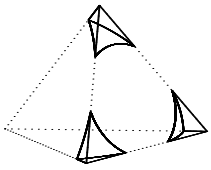
As before with the two-dimensional algorithm, the post-trimming uncovered volumes are returned and can be used as a basis for mesh mending.



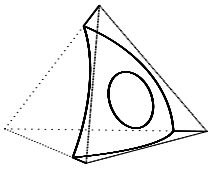
(a) Find the intersections of the support with the bounding segments



(b) Trim the segments to their intersections, split where needed



(c) Create new curves and surfaces where needed



(d) A support can intersect the surface without intersecting any bounding segments

Figure 12. The spherical support intersection algorithm at different stages with original tetrahedron shown with a light line for reference

2.6 Examples

Figure 13 depicts an RKEM mesh representing a slice from a CT scan of a tooth from a bullshark. The sequence shown in Fig. 14 shows the uncovered regions of the mesh as determined by the algorithm given in §2.5.4 for three different scale factors. The pictures shown display the nodes of the mesh as points, and the solid lines are the segments bounding the regions of the mesh that remain uncovered. In Fig. 14(a), the scale factor was set to 0.525 with a large portion of the mesh uncovered. In Fig. 14(b)-(c), the increasing scale factor leads to increased mesh coverage. Finally, a scale factor, $f = 0.92$ yields a fully covered mesh (not shown). Thus, any scale factor $0.92 \leq f < 1$ will fully cover the mesh while retaining the Kronecker-δ property

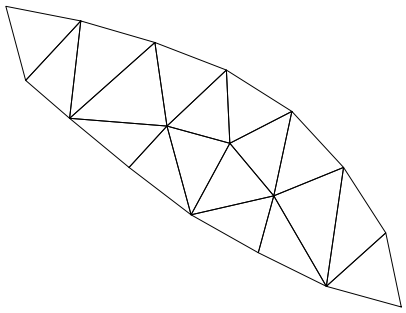


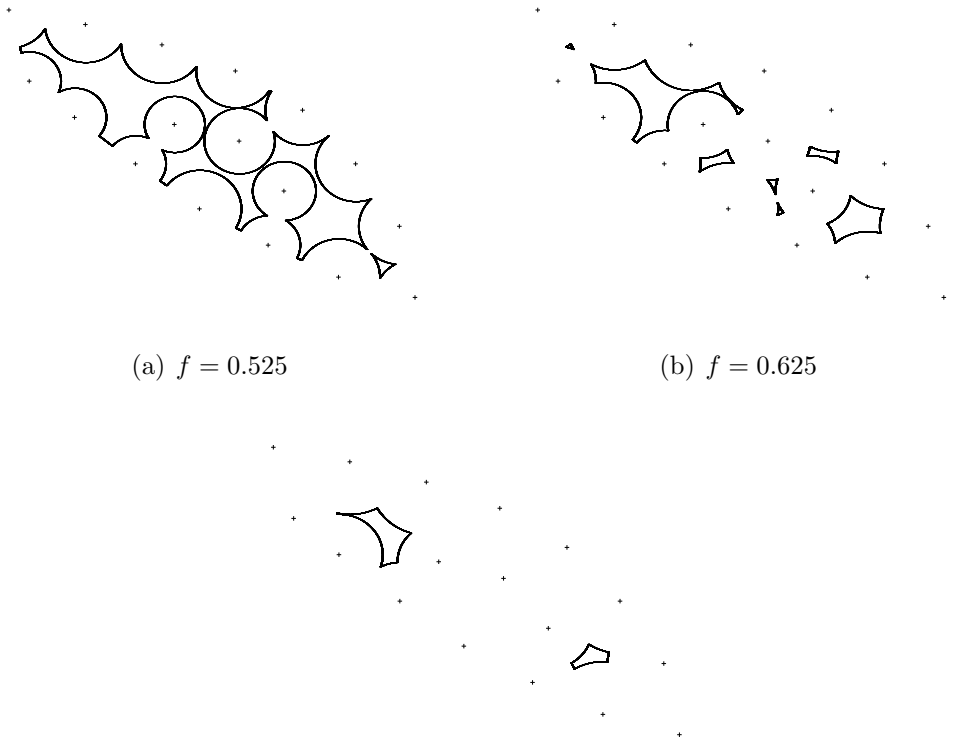
Figure 13. Triangular mesh representing a slice of a bullshark tooth

at every node. To demonstrate that the RKEM quasi-uniformity condition is not necessarily restrictive, the well-graded mesh shown in Fig. 15 was generated using a standard FEM quality mesh generator. This mesh is RKEM quasi-uniform with no modifications.

A series of nine images (Fig. 16) show a single tetrahedral element intersected with a increasingly large spherical support factor. A single element is shown here due to difficulties in visualization of large amounts of three-dimensional data. In this particular example, the largest scale factor shown is $f = 0.6$. The tetrahedron is fully covered when larger scale factors are used. Here the smaller values are chosen to demonstrate algorithmic capability.

2.7 Mesh Mending

A key feature of the algorithm is that it returns the uncovered regions. This important information can be used to add vertices to the mesh to increase coverage. Adding the vertices, called Steiner points in the meshing literature, combined with mesh smoothing to achieve RKEM quasi-uniformity, is termed mesh mending. Many approaches may be useful, but as a first step a bounding box of each connected region is calculated and a node inserted at the bounding box center. Following insertion, the points are re-tessellated and a smoothing technique applied. Currently, a physics-based smoothing technique detailed in [26] is used.



(a) $f = 0.525$ (b) $f = 0.625$

(c) $f = 0.725$

Figure 14. Uncovered regions of bullshark tooth mesh for different scale factors, f

Consider the following two-dimensional example mesh depicted in Fig. 17(a). This mesh was generated by evenly distributing points along the boundary and then randomly inserting interior points. These points were then triangulated without any quality constraints. This mesh cannot be covered for any value of $f < 1$; the uncovered regions are shown in Fig. 17(b). The mesh mending procedure is applied to this mesh. After one refinement, the mesh shown in Fig. 18(a) is generated and the resulting coverage shown in Fig. 18(b). One more application of mesh mending leads to the mesh shown in Fig. 19, which now meets the RKEM quasi-uniformity condition.

While this procedure is independent of dimension, two-dimensional problems were successfully mended while three-dimensional problems were not. When the uncovered regions of the three-dimensional meshes were used as a basis for node insertion, the

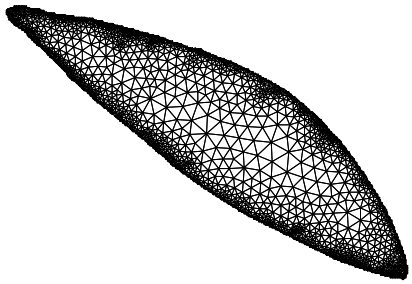


Figure 15. Well graded RKEM quasi-uniform mesh

resulting tetrahedral mesh did not result in a RKEM quasi-uniform mesh. Subsequent applications of this technique did not satisfy the condition as in the two-dimensional meshes.

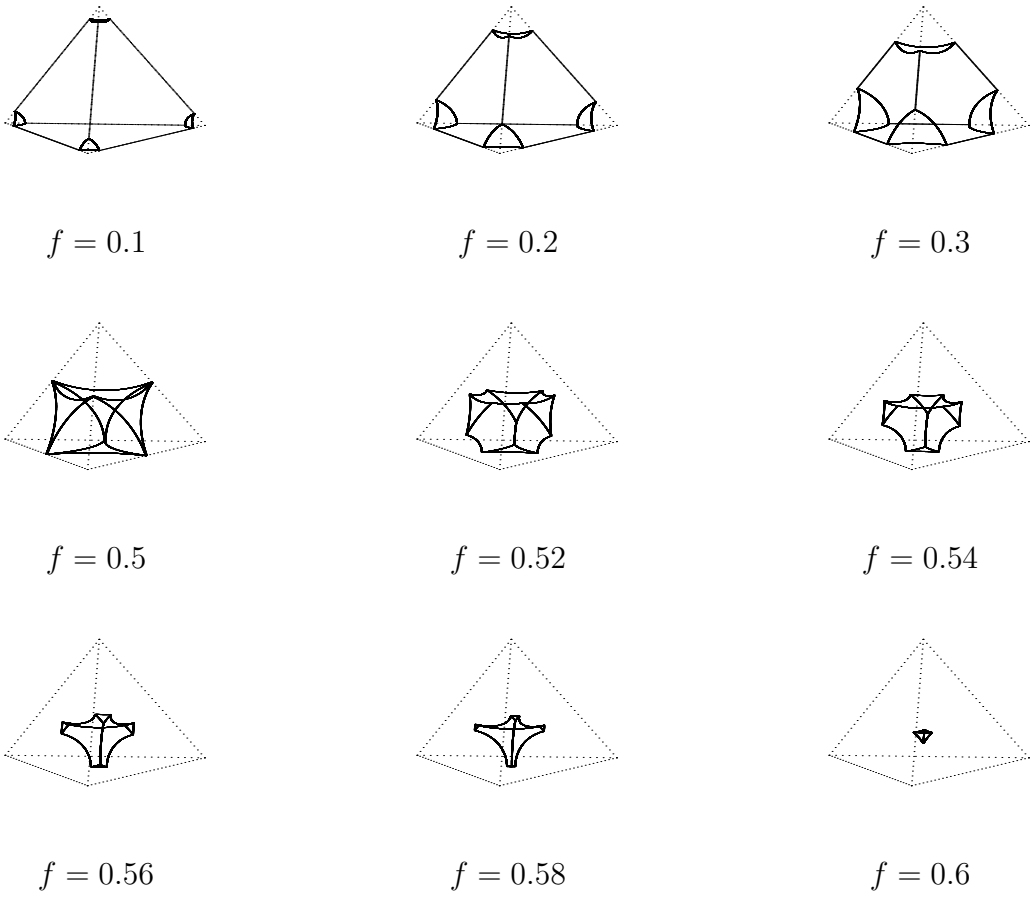
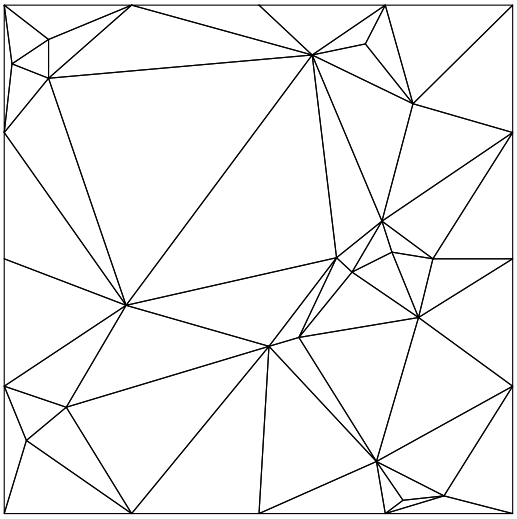
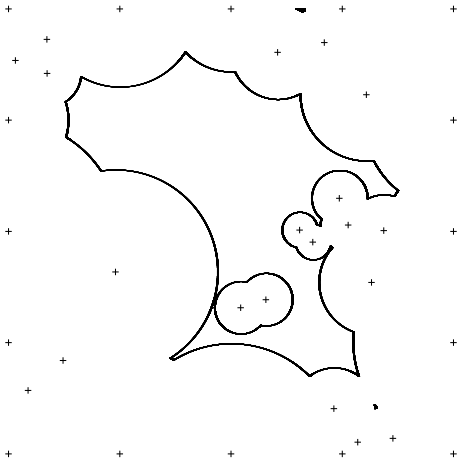


Figure 16. Sample tetrahedral element, shown in a series of increasing support sizes

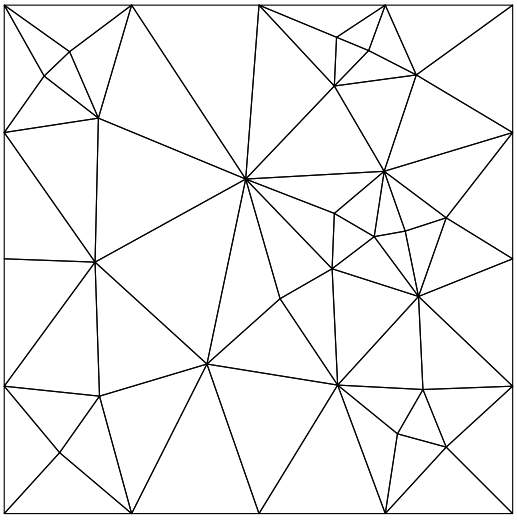


(a)

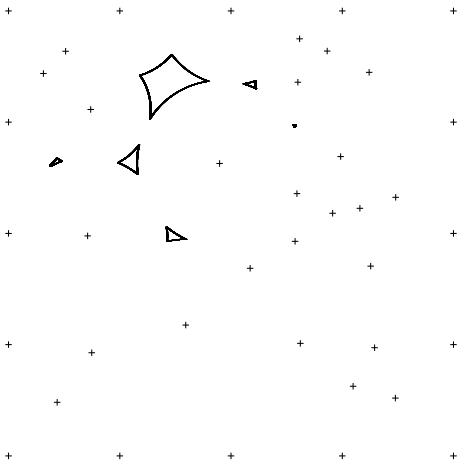


(b)

Figure 17. Random mesh (a) and accompanying uncovered areas (b)



(a)



(b)

Figure 18. First refinement mesh (a) and accompanying uncovered areas (b)

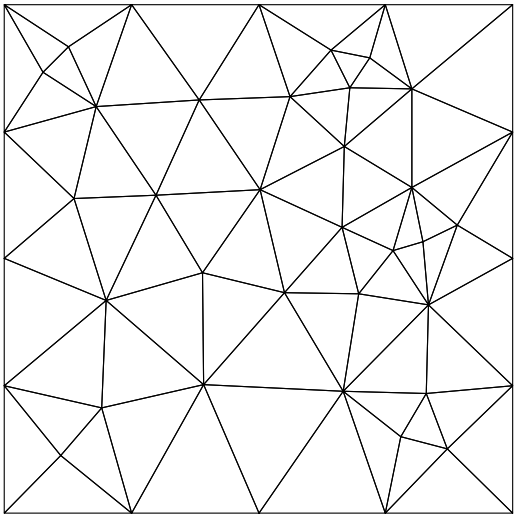


Figure 19. Second refinement which now passes quasi-uniformity

CHAPTER 3

GEOMETRY REPRESENTATION

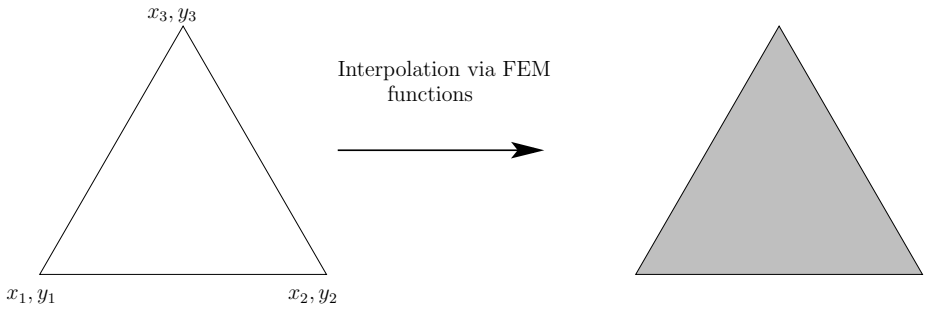
As discussed in the introduction, the main aim of studies in computational mechanics is to improve the design process by numerically approximating governing differential equations. For any kind of significant computation, this is done by computer. The implicit consequence of this is that the domain of the problem must be represented discretely in some fashion within the computer memory.

While not an emphasized aspect of finite elements, the geometry is approximated by the mesh whose elements are used to create the interpolation field. Finite elements provides an approximated weak-form solution to an approximated geometry. This geometry is a polygonal or polyhedral approximation. While for a large number of problems this approximation is of no consequence, provided that small enough elements are used to capture major geometric features, in other problems this can lead to difficulties. Fluid-structure interaction, nano-mechanics, and biomedical problems all could benefit from a smoother, precise geometry. Such problems are sensitive to sharp discontinuities in the geometry.

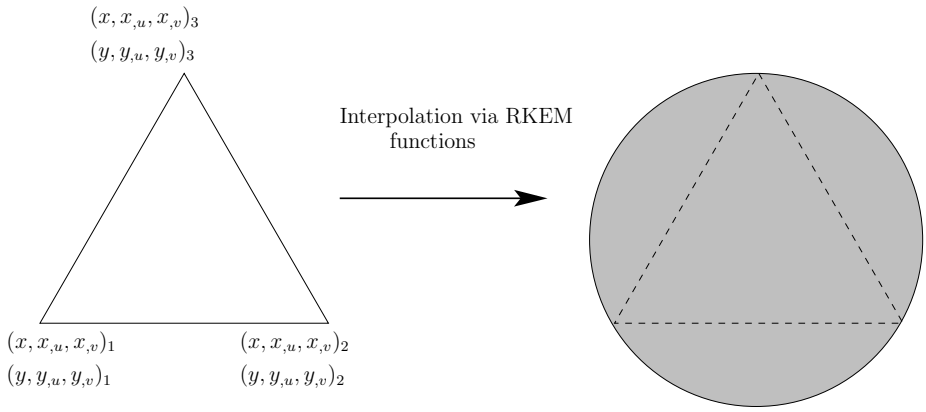
3.1 RKEM Geometry Representation

The Reproducing Kernel Element Method's higher-order globally smooth interpolants are ideal for use in the representation of geometries, first explored in [33]. In FEM, the geometry can be thought of as an interpolation of weights located at the element nodes. Since FEM basis functions possess the Kronecker-δ property, these geometry nodal weights will be equal to the geometric location of the nodes as shown

in Fig. 20(a) on the left. Typical finite element functions are linear (in multiple dimensions bi- or tri-linear), so the finite element interpolation of the geometric nodal weights is the element itself. Here, a circle is being represented with a single triangular element. For the finite element case, the geometry is also a triangle. While a single triangle is not a practical mesh for a circular problem, here it is used for demonstration purposes.



(a) FEM Geometry Representation



(b) RKEM Geometry Representation

Figure 20. Differences in the use of interpolants to represent a circle with a single triangular element

As briefly discussed in §1.2.3, RKEM interpolation can be enriched by adding nodal weights and accompanying element polynomial fields. While this can also be done in FEM, it is in RKEM that the node centered kernels enforce compatibility making the creation of such fields trivial.

In the case of RKEM (using a T9P2I1 element, two-dimensions), the geometry may be interpolated by the basis functions with both primary variable and derivative geometric nodal weights. In Fig. 20(b) these nodal weights are written as $(x, x_{,u}, x_{,v})$, one set for each spatial dimension and for each node. Derivative weights are with respect to the u and v coordinates, which are local mesh coordinates. This can be thought of as a generalized Hermite interpolation. The RKEM interpolants still possess the Kronecker-δ property, so the primary variable nodal weights are the nodal locations just as in FEM. Here, if derivative nodal weights can be determined, a more complex geometry may be computed. Figure 20(b) represents conceptually how more degrees of freedom and higher degree functions can map a single triangular element mesh into a different geometry. The geometry represented is that of a circle where the original mesh is included as a dotted line.

The geometry representation process can be conceptualized as shown in Fig. 21. Here a point on the actual geometry, $\mathbf{X}$, is determined to lie at a point $\mathbf{U}$ on the RKEM mesh. The nodal parameters are then determined such that $\mathbf{U}$ interpolates to a point $\tilde{\mathbf{X}}$ which coincides with $\mathbf{X}$. The RKEM interpolation serves as a mapping from the mesh domain to the representation domain much the way the the deformation function maps points from the Lagrangian to Eulerian frames in continuum mechanics.

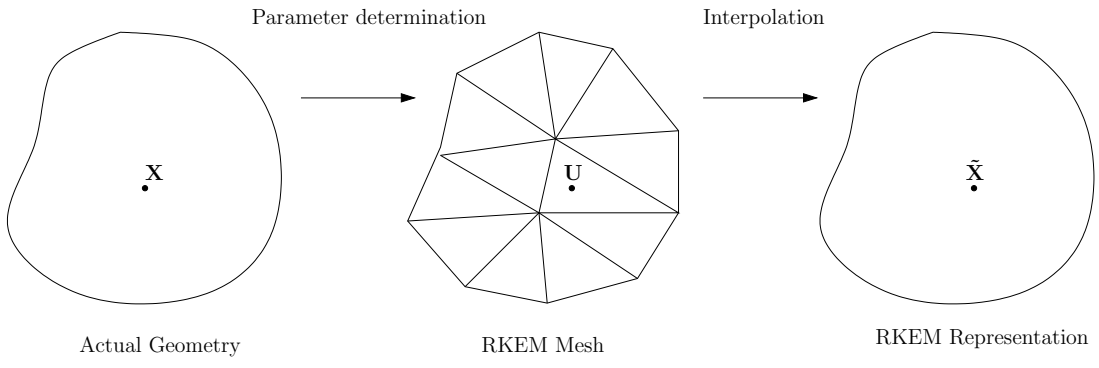


Figure 21. The RKEM geometry representation process

This idea was published originally in [34] where boundary and volume representations were shown in two-dimensions for trivial and non-trivial geometries and a surface representation was shown in three-dimensions for a sphere. Further study has been impeded by a lack of the ability to generate and assess RKEM quasi-uniform meshes.

3.1.1 Isogeometric Analysis

The representation of geometry with RKEM then allows for analysis of a problem where a smooth geometry may be interpolated on the same mesh and with the same basis functions as the weak form solution. This has been termed isogeometric analysis and received much attention in recent years [2, 3, 9, 18, 30]. The research group publishing these papers has used Non-Uniform Rational B-Splines (NURBS) [29], commonly used in computer graphics to represent smooth geometries, to solve Galerkin weak form problems. These references are included here for completeness, but will not be reviewed in detail.

The vision of the use of RKEM is in applications where smooth geometries are difficult to obtain, e.g. imaging-based data in biomedical or nano-technology. In engineering design, the geometries are mathematical descriptions in CAD systems and subsequently manufactured. These CAD systems are often based on NURBS and therefore analysis with NURBS makes sense in these applications. Obtaining geometries suitable for analysis is still an open problem for applications where the input data come in the form of discrete images. In applications where the geometry is not a connected system of basic geometry primitives, even NURBS has difficulty in determining a representation. A framework for the use of RKEM in this manner was published in [10].

3.2 Geometry Representation Procedure

The procedure used is outlined here and specific tasks are explained in more detail. To simplify the discussion, only the procedure used to obtain surface representations is discussed here. The extension to volume representations is straightforward.

1. *Obtain an image or series of images depicting the object of interest.* For work in two-dimensions, extract an ordered list of segments representing the boundary of the object. For three-dimensions, a closed list of faces fully representing the object boundary is needed. Current segmentation software already handles this operation for both dimensions.

2. *Obtain a coarsening of these boundary data.* While it is possible to simply triangularize/tetrahedralize this data, the goal of this process is to obtain smooth representations with RKEM meshes with significantly fewer elements than the tessellated original data set. Image data usually returns pixel locations as the geometric data, so there can exist a great deal of information, some of which may be extraneous. This coarsened data are then tessellated to obtain a RKEM mesh. Details of the methods used to achieve this will be discussed in §3.2.1.

3. *Check that the coarsened mesh is RKEM quasi-uniform.* This is done using the algorithms specified in this dissertation. The mesh needs to be mended until it passes this condition.

4. *Determine a mesh point for each geometry point.* Finding RKEM representations of smooth data is an inverse problem. When the actual geometry is known discretely, one must find the RKEM interpolation which will fit these known points. *Geometry points* are the coordinates of the original data obtained in Step 1. Each geometry point must have a corresponding location on an edge/face belonging to the coarsened mesh. These locations on the mesh

will be termed *mesh points*. These points are critical because the method of determining the unknown derivative nodal weights will involve the evaluation of RKEM basis functions, which do not have value outside of the mesh. Details of this procedure will be given in §3.2.2.

5. *Solve for the unknown derivative nodal parameters.* The unknown derivative nodal weights are needed to fully represent the geometry using RKEM. The goal is to determine these nodal weights in such a way as to ensure that each mesh point interpolates to its geometry point. Thus, the following equation can be written for each geometry point, $\mathbf{X}$, and its corresponding mesh point $\mathbf{U}$.

$$\mathbf{X} = \sum_{i=1}^{n} \Psi_i^{00}(\mathbf{U})x + \Psi_i^{10}(\mathbf{U})x_{,u} + \Psi_i^{01}(\mathbf{U})x_{,v} \tag{30}$$

where n is the number of RKEM nodes, Ψ_i is the RKEM basis function corresponding to node i, and x, $x_{,u}$, $x_{,v}$ are the nodal weights. The superscripts on the basis functions refer to which nodal weight each function belongs: 00 belongs to primary variable, 10 to the derivative nodal weight in the u direction, and 01 to the derivative nodal weight in the v direction where u and v are local coordinates of the mesh element.

The key point here is that the only unknowns are the derivative nodal weights, $x_{,u}$ and $x_{,v}$. This equation may be written for each geometry point, and for each spatial dimension. A system of equations can then be written to minimize the error between the interpolated value and the actual value, the coordinate of the geometry point.

$$Error = \mathbf{X} - \sum_{i=1}^{n} \Psi_i^{00}(\mathbf{U})x + \Psi_i^{10}(\mathbf{U})x_{,u} + \Psi_i^{01}(\mathbf{U})x_{,v} \tag{31}$$

The large amount of data typically generated by medical imaging guarantees that the least-squares system will always be overdetermined. The amount of imaging data needed is a function of the number of mesh elements and of the degrees of freedom of each element node. The least-squares solution of this system results in a complete set of nodal weights which may be interpolated by the RKEM basis functions to render a more complex geometry.

3.2.1 Data Coarsening Techniques

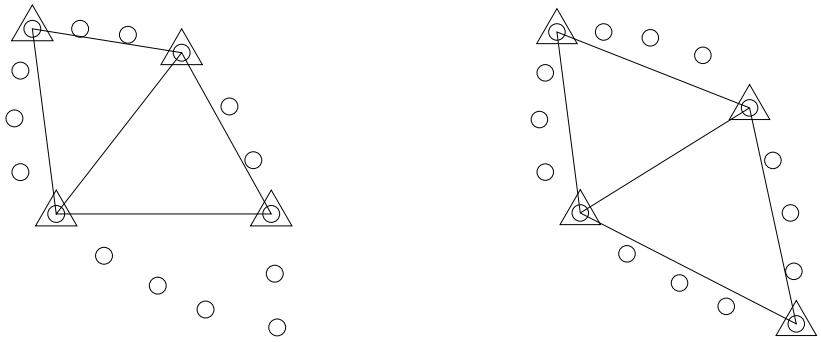
Step 2 of the RKEM geometry representation procedure requires a coarsening of the original data set. This section will describe the techniques used. The data that are obtained by CT imaging or MRI pixel intensities are a single or series of images. The extraction of a geometric object from these data is usually achieved by segmentation. What results is a set of points representing edge or face boundaries of the imaged object. High resolution data will contain many data points. Consider the tip of a Tiger shark tooth, shown in Fig. 30(b). This surface consists of 74,188 triangular faces and 37,096 vertices. In this three-dimensional case, the geometry consists of these nodal locations which requires storage of 111,288 double precision variables as well as face connectivity. While smooth representations are in and of themselves a desirable goal, another goal is to obtain this smooth representation with less data storage.

For this reason, the original data set needs to be coarsened. For data in two-dimensions this is straightforward. Essentially, the goal is to find a subset of nodes, $\mathcal{N}_c \in \mathcal{N}$, where $\mathcal{N}$ is the set of node indices representing the nodal locations of the original data set. This subset $\mathcal{N}_c$ should chosen be such that

$$\|d_{i,i+1} - d_{i,i-1}\| < \delta \tag{32}$$

where $d_{i,j}$ is the node-to-node distance between neighboring nodes i and j and δ is some prescribed tolerance. What this means is that neighboring edges should not vary in length more than an amount δ. Otherwise this can lead to tessellations that are not RKEM quasi-uniform.

It may be important to force certain nodes of $\mathcal{N}$ to be part of $\mathcal{N}_c$. If the set of points is chosen according to Eq. 32 with no regard for key features in the geometry, a tessellation of these data can result in a mesh that must represent unnecessarily high amounts of curvature. This phenomenon is depicted in Fig. 22(a). A better set includes nodes defining key geometry features, as in Fig. 22(b). While the RKEM interpolants are higher order and capable of resolving changes in curvature, this can produce negative effects in other areas of the representation. This is particularly important in volume representations, as illustrated and explored in [10].



(a) Poor choice of $\mathcal{N}_c$ (b) Better choice of $\mathcal{N}_c$

Figure 22. Two selections of $\mathcal{N}_c$ and resulting mesh, original data points shown as circles, coarsening shown as triangles

Three-dimensional data-coarsening techniques can borrow concepts from the computer science community. In this context, data coarsening is known as mesh decimation and has been widely studied [7, 14, 16]. For computer scientists this problem is important in simplifying a mesh-based geometry for ease in rendering, especially when the geometry is over-sampled compared to the amount of data required to represent the geometry.

While it is beneficial to review work done in graphics and visualization, this should be done with careful consideration. While many times the challenges faced in computer science and computational mechanics are similar, particular solutions may not be suitable for both fields. For example, in the case of decimation, the goal of the computer scientist is to reduce the complexity of the mesh-based geometry while maintaining the overall appearance of the object. In this dissertation's application, the need is to find a coarsening that is a proper subset of the original geometry points. Creation and moving of face vertices are not helpful, and the appearance of the final mesh is of no consequence. For this reason, algorithms must be carefully weighed to determine what is suitable for the particular application.

The techniques reviewed in mesh decimation rely on the relocation and deletion of mesh entities. This makes them unsuitable for use in this application. A successful mesh decimation technique for use in representation of geometries in RKEM will have the following characteristics.

1. Generates a simpler mesh whose nodes are a subset of the original mesh.

2. Resulting mesh faces should also pass RKEM quasi-uniformity or some similar metric. A coarsening that will not lead to a RKEM quasi-uniform mesh is not desirable.

3. Minimize curvature represented by each mesh face. This is similar to the ideas presented in two-dimensional coarsening. Points should be chosen such that key geometric features are maintained.

3.2.2 Projection Techniques

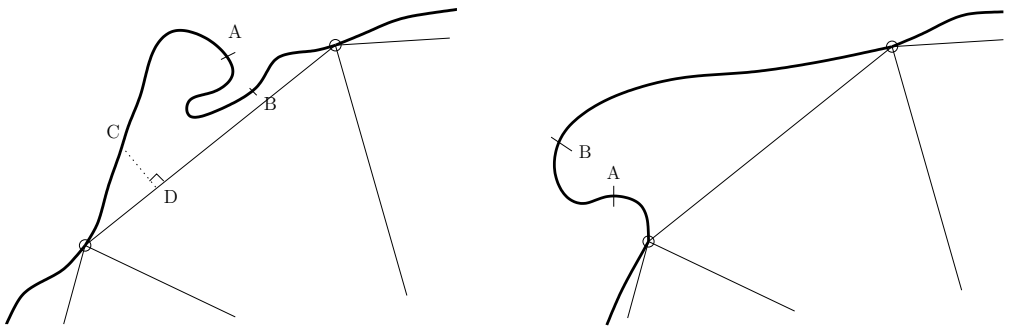
Once the coarsened meshes are obtained, a point on the boundary of this mesh (mesh point) must be found for every point on the original data set (geometry point). This is part of a process that will eventually use these mesh points to find the unknown

derivative nodal weights and complete the geometry representation. The determination of these mesh points will involve projecting the geometry point along certain directions, determined from different criteria. This step is key to obtaining suitable representations, and while several methods are presented, none is shown to be optimal for the general case.

In two-dimensions, the geometry point may be projected in a direction perpendicular to a mesh edge to determine the corresponding mesh point. In Fig. 23(a) this is shown where point D is the perpendicular projection of point C. While this is a simple method of determining a mesh point, it is not without pitfalls. It is not sufficient to merely find a corresponding mesh point for every geometry point. These points are used in the determination of the geometry. Sets of points whose projections cross each other or lie off the mesh edge can cause nonsensical results in the process of determining unknown derivative nodal weights.

Consider the case where a series of geometry points contains a subset of points which project to the same mesh point as shown in Fig. 23(a). In this figure, the darkened line represents a fine sampling of geometry points, and the region between A and B contains points that could share a mesh point with other geometry points. This can cause difficulties in the determination of the unknown nodal parameters, because a single mesh point maps to multiple geometry points, violating the definition of a function. These points are detected and filtered to eliminate this possibility.

Another potential pitfall is depicted in Fig. 23(b). Here, the geometry points between A and B should have corresponding mesh points on the mesh edge shown. However, the curvature of the geometry causes their perpendicular projections to fall off the edge. These points are also omitted as not only are the projected mesh points off the edge, they are off the mesh and have no basis function values. In both cases the points between A and B are removed to eliminate potential difficulties.



(a) Multiple geometry points contain the same mesh point

(b) Mesh point falls off the mesh edge

Figure 23. Cases were data are eliminated when determining mesh points

A better method, perhaps, is to fit all geometry points parametrically between the two edge nodes. While this method causes no information to be lost, it is not necessarily optimal. Both filtering cases discussed in the preceding two paragraphs address the removal of information when the geometry points represent a highly varying curvature along a certain edge. Including all such points could cause negative effects in the final representation such as an inverted mapping. This effect is not yet clear and requires more study to fully understand.

In three-dimensions, projection methods are far less straight-forward. For two-dimensional problems, an added efficiency is the knowledge that the ordered list of geometry points must lie between two nodes of the coarsened RKEM mesh boundary edge. This pre-knowledge is useful in finding projections. For three-dimensional data, this same efficiency is not present and cannot be exploited.

While the data are a dimension higher, the overall goal is the same in three-dimensions as for two-dimensions: find a mesh point for each geometry point. Three separate methods of finding these mesh points are explored and detailed here:

1. *Average Normal Projection* This method was used in [34] to determine the mesh points for the sphere published in that work. The idea was to average the inward facing normals of all faces which contain a particular geometry point. The mesh

point was then determined to be the intersection of this direction with the first face encountered. This can be graphically seen in Fig. 24.

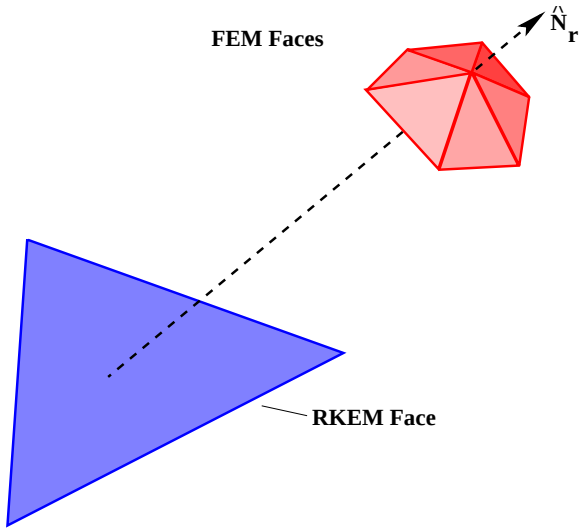


Figure 24. Average normal projection

While this method is convenient is the sense that only information local to the geometric point is used to calculate a projection direction, the resulting mesh points are not likely to generate well behaved geometries. This is because neighboring mesh points could map to geometry points which are far away and on different sides, inverting the mapping. Even though this effect is undesired, it could be used as an indicator that the mesh needs refined to adequately represent the geometry.

This undesirable effect can be seen in Fig. 25 in a two-dimensional analog. A series of geometry points are shown along with their neighboring faces as heavy lines. The average normal is approximately drawn and shows where the corresponding mesh points would appear. Note that the normal projection of the geometry point labeled A is not even on the edge closest to the point. Especially undesirable is the effect of the series of points B, C, and D. Note that the projections of the geometry points to the mesh points actually cross.

This will cause the representation to invert at this point, which is not a desired behavior.

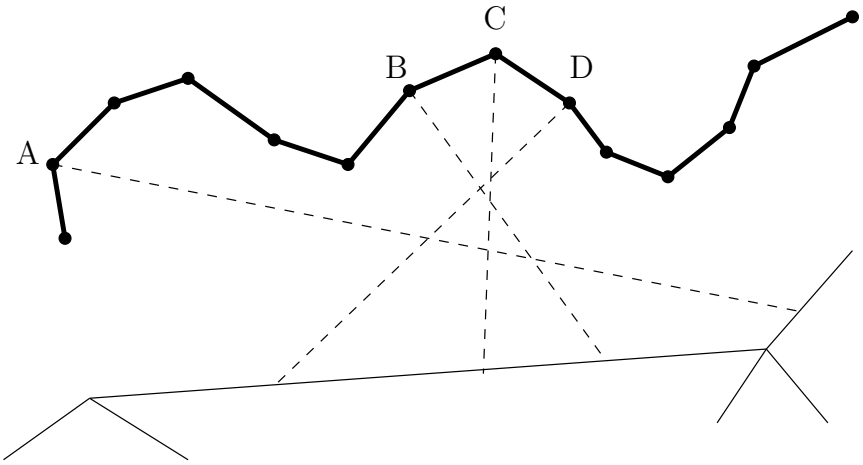


Figure 25. Problem with the average normal projection method

For the sphere, this method worked perfectly because the curvature changes uniformly along the solid. This method is unsuitable for any general geometry.

2. *Mesh Center Projection* As the name suggests another method of determining mesh points is to determine a mesh center and find the intersection of the ray beginning at a geometry point and in the direction of this center with a mesh face. This method branches into two sub-methods at this point. The intersection may be found with the nearest face along the ray pointing to the mesh center in which case, each geometry point will have a mesh point and no data is filtered. This method is used to generate the results shown in §3.3.

   The second sub-method uses a filter. While the face on which the geometry points must find mesh points is not known as in the two-dimensional case, a simplification may me made where a geometry point must find a mesh point at its nearest face. The nearest face will be the face whose perpendicular projection to the geometry point is both on the face and the minimum for all mesh faces.

This projection technique will generate differing results and will filter out mesh points which project to a face not its closest.

This method also has a pitfall in the computation of the mesh center. If the mesh center does not lie inside the mesh, this method will produce non-sensual results. The location of the center is also a factor in the quality of projection. However, for round objects, the simplification is perfectly valid.

3. *Perpendicular Projection* The last method considered involves determining the perpendicular projection to the nearest face. This mesh point must lie on the triangular mesh face to be valid, otherwise it is filtered. This method is similar to what is done in two-dimensions.

As in the two-dimensional case, it is unclear which technique is optimal. The quality of the choice of these mesh points can greatly affect the end geometry and needs to be studied in more detail.

3.3 Three-dimensional Results

In this section, the RKEM geometry representation procedure is used to represent an analytic geometry, a sphere, as well as geometry obtained from imaging, a shark tooth tip. In both cases, the original data set is shown as well as the underlying meshes and representations.

3.3.1 Sphere

A progression of spherical meshes was generated by an analytical code and checked for RKEM quasi-uniformity. A data set, shown in Fig. 26, was generated and considered the original geometry. The series of increasingly refined representations may be seen in Figs. 27-29. In each figure, the underlying RKEM mesh is shown on the left and the accompanying representation on the right. Since this geometry is analytic, the error can be computed, for which the maximum is listed in Table 1. The error metric used was the Euclidean distance between a geometric point and the interpolation of the corresponding mesh point. Note that while this is one possible metric, other metrics could be used. This metric is beneficial because it not only measures the effect of refinement but also measures the performance of the projection method chosen. Refinement may not be as beneficial if a poor method of projecting geometry points was chosen.

While the results of Table 1 demonstrate a clear convergence, the maximum value of error is larger than expected. In the NURBS representation of a sphere [29], a numerically exact representation is determined with a 26 control point mesh. These results for the RKEM sphere demonstrate slower convergence that expected.

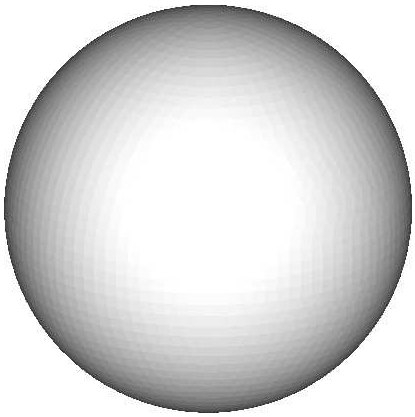
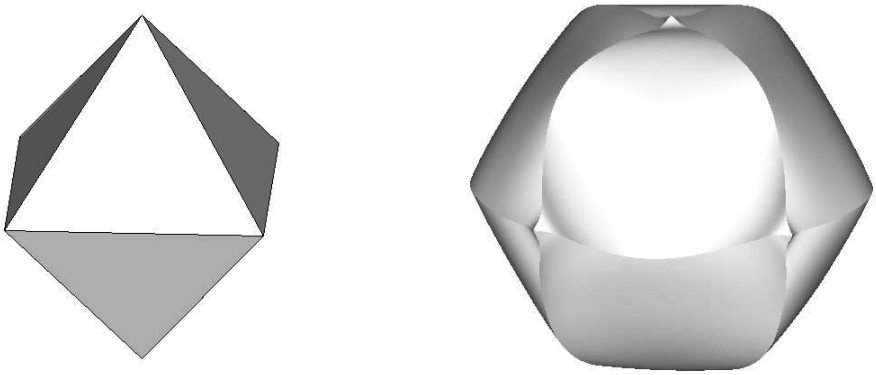
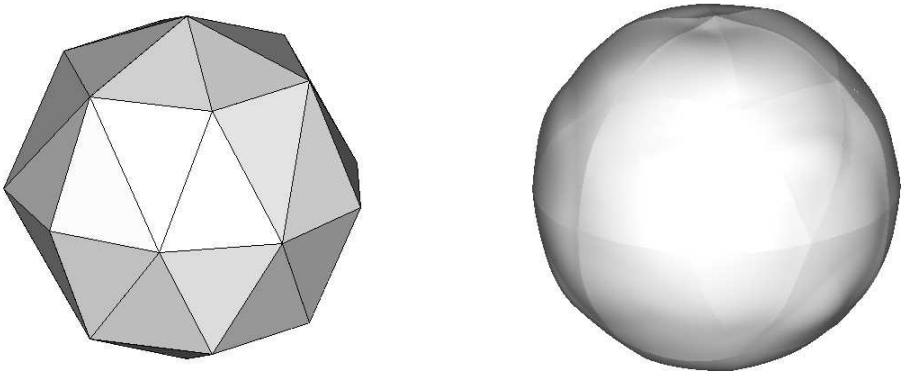


Figure 26. Original sphere geometry



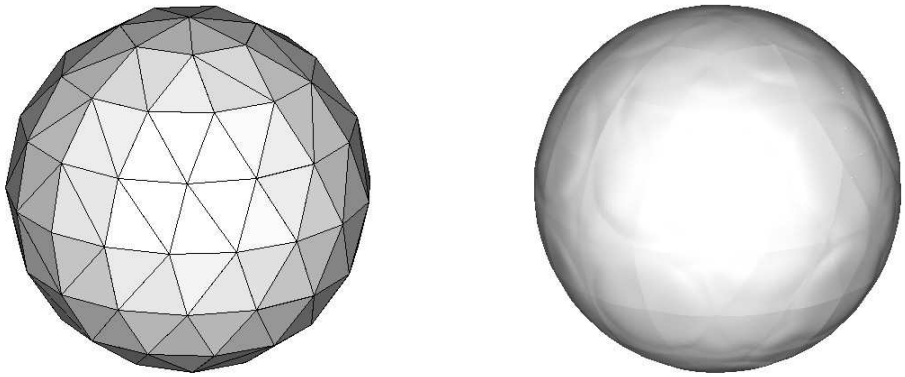
(a) Mesh (b) Representation

Figure 27. Sphere representation with 8 faces



(a) Mesh (b) Representation

Figure 28. Sphere representation with 42 faces



(a) Mesh (b) Representation

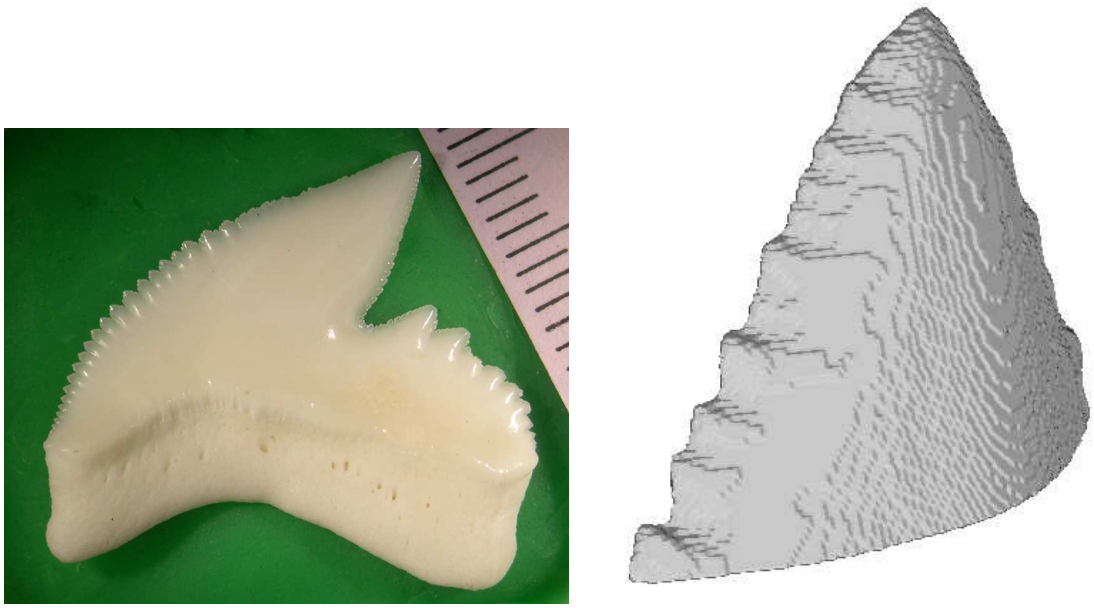
Figure 29. Sphere representation with 170 faces

Table 1. Convergence of sphere representation

Boundary Faces	Elements	Max Error
4	1	1.15
8	8	0.59
42	51	0.22
170	358	0.06

3.3.2 Shark Tooth Tip

The shark tooth tip comes as a list of triangular faces assembled from CT imaging data. This data set comes from a study on the evolution of shark teeth, in [37]. The original geometry has many terraced features (Fig. 30(b)). These terraces are not an actual tooth feature as can be seen in Fig. 30(a), but created from the discrete method used to obtain the geometry. The extracted geometry is but the tip of the original tooth. These features are undesirable in the final representation. Alternatively, the tooth has several serrations on its edge. These are features that should be captured. The goal is that this procedure can capture large features and smoothen the small undesirable artifacts.

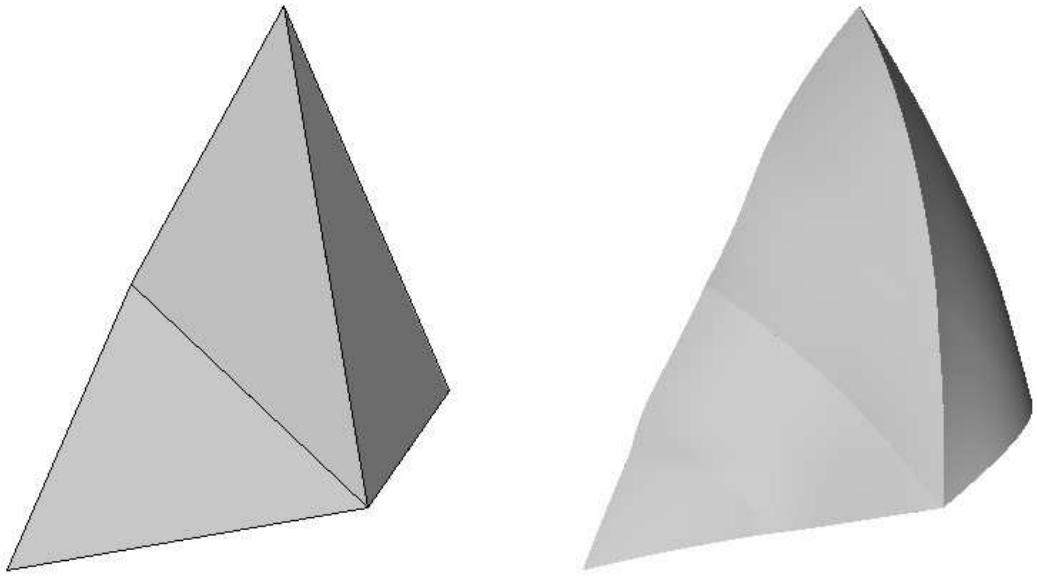


(a) Tiger shark tooth (b) Original Tooth Tip Geometry

Figure 30. Tiger shark tooth and its tip (used with permission from [37])

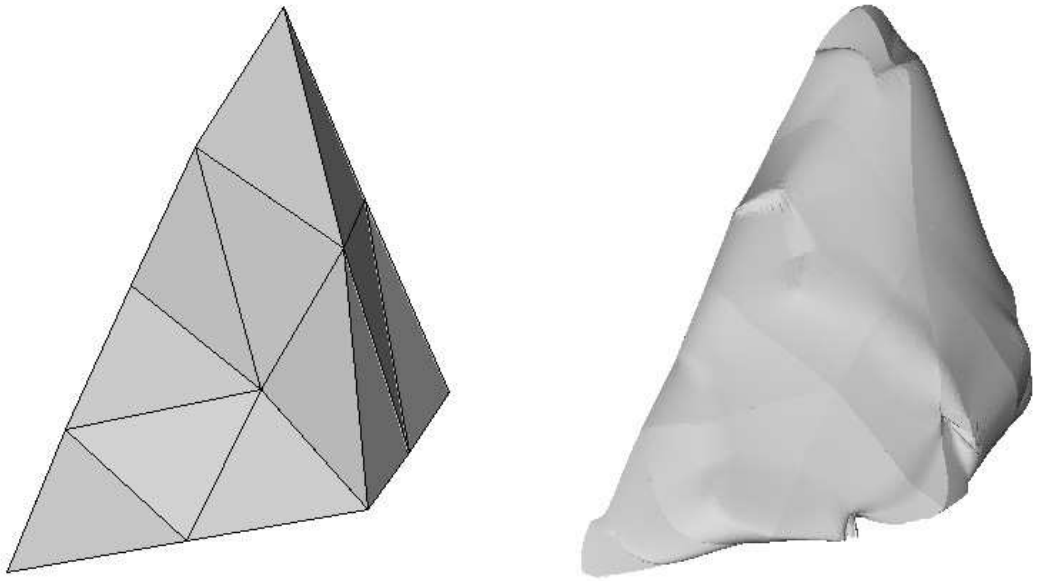
The representations shown (Figs. 31 and 32) are for meshes, hand picked and tested to be RKEM quasi-uniform. Note that while the mesh is coarse, the representation is faithful to the original shape of the tooth. Subsequent refinements were not possible during the duration of this study. The mesh mending technique detailed in

§2.7 worked well for two-dimensional data, but in practice did not work well in three-dimensions. The inserted nodes into the uncovered volumes did not result in RKEM quasi-uniform meshes. For further work in geometry representation, it is important to further develop meshing techniques that are guaranteed to be RKEM quasi-uniform. This is beyond the scope of this work. This problem was obviated in the case of the sphere because the geometry was analytical and RKEM quasi-uniform refinements easy to generate.



(a) Mesh (b) Representation

Figure 31. Tip representation with 3 elements



(a) Mesh (b) Representation

Figure 32. Tip representation with 31 elements

3.4 Derivative Discontinuities Across Element Boundaries

As stated several times throughout this dissertation, RKEM interpolants are higher-order globally smooth. This means that the function interpolated possesses an arbitrary number of smooth derivatives. The tetrahedral elements used to generate the results of this section (Tet16P2I1, [34]) are C^2 continuous, yet the results have an apparent C^0 continuity along places where element boundaries exist. This is particularly apparent in the spherical cases of Figs. 27-29.

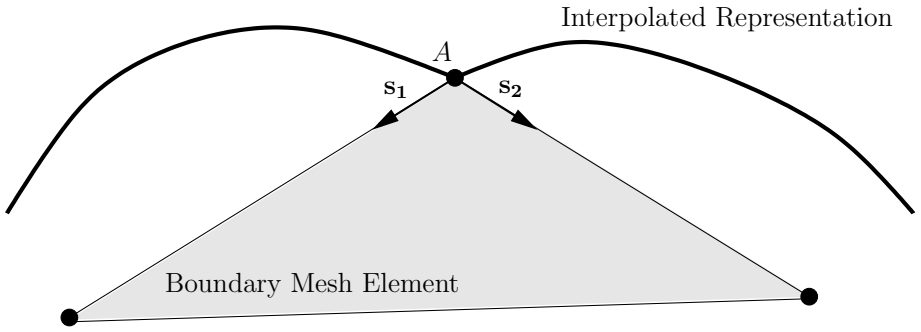


Figure 33. Derivatives at boundary elements are not well defined

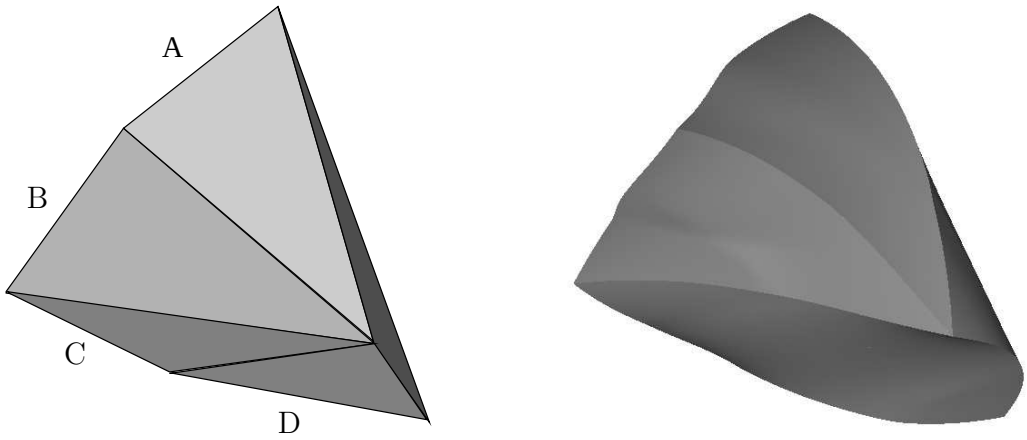
This is, in fact, not a contradiction of the RKEM interpolant properties. Consider the two-dimensional analog shown in Fig. 33. Here the geometry is interpolated and shown as a heavy black line with a derivative discontinuity at a point labeled A. The underlying mesh is shown as a shaded triangle. The issue is related to the fact that determining the unknown parameters at the nodes involves taking data that are fundamentally smooth and projecting these data to mesh edges/faces that are not smooth. The interpolated representation, expressed as a function f, would be continuous at A if

$$\frac{\partial f}{\partial \mathbf{s_1}} = -\frac{\partial f}{\partial \mathbf{s_2}} \tag{33}$$

where $\mathbf{s_1}$ and $\mathbf{s_2}$ are vectors as shown in Fig. 33. It is illogical to expect derivatives with respect to different variables to be equal at general locations. For example, let $f(x, y) = 1 - x^2 - y$. In this case $\frac{\partial f}{\partial x} = -2x$ and $\frac{\partial f}{\partial y} = -1$. Although at $x = 0.5$

these derivatives are equal, for a general (x, y) the derivatives will be distinct. This does not indicate that the function f does not possess as much continuity as its order would indicate. This is the cause of the seams seen in the surface representations and not a contradiction of the RKEM higher order, globally smooth property.

Consider the shark tooth tip originally shown in Figs. 30(b)-31 and repeated here in Fig. 34 with a different view angle. This view is tilted upward to show the underside of the tooth. The mesh in Fig. 34(a) has labels for different faces. Faces labeled A, B and C all have differing planar normals indicating that an interpolation from one element face to another must change parametric directions. This effect can be seen in Fig. 34(b) by clear lines in the vicinity of element edges. However, the planar normals of faces C and D are equal, meaning that an interpolation from face C to D is possible without change in parametric direction. In the area where faces C and D meet, there is no such sharp edge. However, in general, these apparent discontinuities will exist unless an alternate method of determining nodal weights which eliminates them can be determined.



(a) Mesh (b) Representation

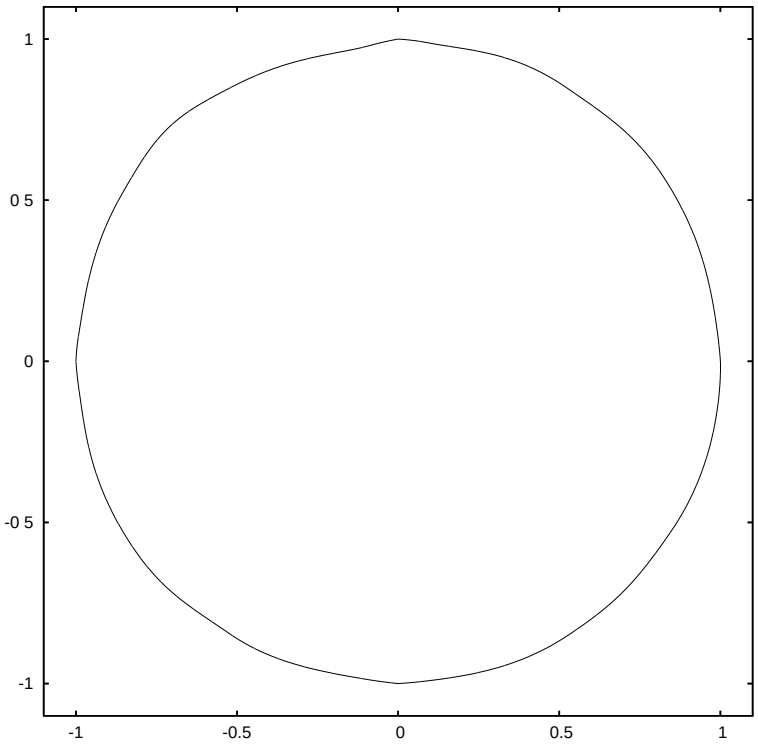
Figure 34. Tip representation with 3 elements, tilted view

The same effect is seen, albeit less pronounced, in the two-dimensional results previously published [34]. Consider the circle representation with 4 elements as seen in Fig. 35. The full circle is shown in Fig. 35(a), with the top of the circle shown

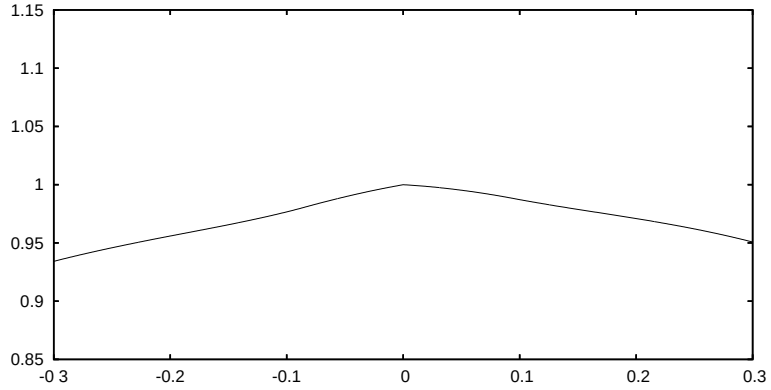
in greater detail in Fig. 35(b). Note clear derivative changes going from the left side of the circle to the right. This is at an element boundary, which demonstrates the same effect as seen in three-dimensions. This was overlooked in two dimensions because of its apparent disappearance under refinement. This effect is more obvious in three-dimensions because it is far more visible. This is due to

1. The affected area is a line (element edge). In the two-dimensional case, this effect was only seen at a point (element nodes).

2. The three-dimensional representations are rendered with shading. This highlights sharp changes in curvature because neighboring pixel intensities vary.



(a) Full circle



(b) Zoomed in to show top portion of circle

Figure 35. The four element circle representation has discontinuities

CHAPTER 4
CONCLUSIONS

This dissertation has addressed a wide variety of topics. First was a discussion on the Galerkin weak form procedure and how the Finite Element Method can be used to approximate a solution. This lead to discussions of methods that have emerged to address weaknesses in FEM, of which RKEM is one. A fundamental property of a RKEM mesh is that it be quasi-uniform. Precise definitions were developed based on general set intersections and an algorithm was written and shown to function in two- and three-dimensions. Procedures and techniques for healing offending meshes were also shown to function for typical and random data.

In a second part, the motivation for the development of RKEM quasi-uniformity was revealed showing how RKEM interpolants can be used to represent geometries. While the process is complex, the end result is a higher order smooth representation which can contain the same underlying mesh as a typical mesh for use in finite elements. Finally, results were shown in three-dimensions for the surface representation of a sphere and a shark tooth.

While a significant result in and of itself, the aim of this work is not to merely represent geometries with RKEM interpolants. The use of these representations to solve problems isogeometrically is of great importance. This aspect was not discussed at length in this dissertation. This is an active area of research, for which this work is part of a larger scope.

4.1 Future Work

This document represents what has been done with RKEM during my studies, but there remains a lot of possible work to expand and improve these ideas.

1. The description of the quasi-uniformity condition for RKEM meshes is important for the existence of basis function properties. However, the element and support sizes will have an effect on the convergence of RKEM when used in the context of a Galerkin weak form approximation. It is important to understand this effect to assure proper results and convergence.

2. While coverage is necessary, nodal isolation is only needed for nodes on which the Kronecker-δ property is desired. The loss of this property for a set of nodes will affect the basis functions, making them smoother. It remains a open problem as to what is the optimal arrangement to provide the best interpolants.

3. The check detailed in chapter 2 is a first step in a greater effort to generate meshes that are guaranteed to be RKEM quasi-uniform. This capability is essential to pursue areas in three-dimensions. Two specific ideas may prove useful:

   (a) Use the radius of supports of the nodes as a means of node insertion, similar to what is already done for FEM quality meshes.

   (b) A smoothing technique called bubble mesh [31] could be used where the bubble centers are the node centers and the bubble radii are the radii of support. If the bubbles are sufficiently packed before determining the element connectivity, this technique could be used to first determine acceptable nodal locations and support sizes and subsequently tessellated for the corresponding element connectivity.

4. The breaks in continuity on the surface representations (§3.4) are undesirable. If smooth representations across boundary element faces are to be obtained this needs included in the system of equations determining unknown nodal weights.

LIST OF REFERENCES

[1] Atluri SN, Zhu T (1998) A new meshless local petrov-galerkin (MLPG) approach in computational mechanics. Computational Mechanics 22:117–127

[2] Bazilevs Y (2006) Isogeometric fluid-structure interaction analysis with applications to arterial blood flow. Computational Mechanics 38:310

[3] Bazilevs Y, Beirao Da Veiga L, Cottrell JA, Hughes TJR, Sangalli G (2006) Isogeometric analysis: approximation, stability and error estimates for h-refined meshes. Mathematical Models Methods in Applied Sciences 16(7):1031

[4] Belytschko T, Lu Y, Gu L (1994) Element free Galerkin methods. International Journal for Numerical Methods in Engineering 37:229–256

[5] Chen J, Pan C, Wu C, Liu W (1996) Reproducing kernel particle methods for large deformation analysis of nonlinear structures. Computer Methods in Applied Mechanics and Engineering 139(1–4):195

[6] Chew PL (1989) Guaranteed-quality triangular meshes. PhD thesis, Cornell University, Ithaca, NY

[7] Cignoni P, Montani C, Scopigno R (1998) A comparison of mesh simplification algorithms. Computers & Graphics 22(1):37 – 54

[8] Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms, Second Edition. The MIT Press

[9] Cottrell J, Reali A, Bazilevs Y, Hughes T (2006) Isogeometric analysis of structural vibrations. Computer Methods in Applied Mechanics and Engineering 195(41-43):5257

[10] Daniel C Simkins J, Collier N, Juha M, Whitenack LB (2008) A framework for studying the RKEM representation of discrete point sets. In: Griebel M, Schweitzer MA (eds) Meshfree Methods for Partial Differential Equations IV, Springer-Verlag, Lecture Notes in Computational Science and Engineering, pp 301–314

[11] Dolbow J, Belytschko T (1998) An introduction to programming the meshless element free Galerkin method. Archives of Computational Methods in Engineering 5(3):207–241

[12] Duarte CA, Oden JT (1996) An h-p adaptive method using clouds. Computer Methods in Applied Mechanics and Engineering 139(1-4):237–262

[13] Field D (1988) Laplacian smoothing and delaunay triangularizations. Communications in Applied Numerical Methods 4(6):709–712

[14] Garland M, Heckbert PS (1997) Surface simplification using quadric error metrics. In: SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp 209–216

[15] Gingold RA, Monaghan JJ (1977) Smoothed particle hydrodynamics: theory and application to non-spherical stars. Monthly Notices of the Royal Astronomical Society 181:375–389

[16] Heckbert PS, Garland M (1997) Survey of polygonal surface simplification algorithms. In: SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co., New York

[17] Hughes T (2000) The finite element method. Dover, Mineola

[18] Hughes T, Cottrell J, Bazilevs Y (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering 194:4135–4195

[19] Idelsohn SR, Onate E (2006) To mesh or not to mesh. That is the question... Computer Methods in Applied Mechanics and Engineering 195:4681–4696

[20] Li S, Liu WK (2002) Meshfree and particle methods and their applications. Applied Mechanics Review 55(1):1–34

[21] Li S, Liu WK (2004) Meshfree particle methods. Springer, Berlin

[22] Li S, Lu H, Han W, Liu WK, Simkins DC Jr (2004) Reproducing kernel element method, Part II. Global conforming I^m/C^n hierarchy. Computer Methods in Applied Mechanics and Engineering 193:953–987

[23] Liu W, Jun S, Zhang Y (1995) Reproducing kernel particle methods. International Journal for Numerical Methods in Fluids 20:1081–1106

[24] Liu W, Li S, Belytschko T (1997) Moving least square reproducing kernel method. Part I: methodology and convergence. Computer Methods in Applied Mechanics and Engineering 143:422–453

[25] Liu WK, Han W, Lu H, Li S, Cao J (2004) Reproducing kernel element method: Part I. Theoretical formulation. Computer Methods in Applied Mechanics and Engineering 193:933–951

[26] Lohner R, Morgan K, Zienkiewicz OC (1986) Adaptive grid refinement for compressible euler equations. In: Accuracy Estimates and Adaptive Refinements in Finite Element Computations, Wiley, Chichester Sussex

[27] Lu H, Li S, Simkins DC Jr, Liu WK, Cao J (2004) Reproducing kernel element method Part III. Generalized enrichment and applications. Computer Methods in Applied Mechanics and Engineering 193:989–1011

[28] Owen S (1998) A survey of unstructured mesh generation technology. In: Proceedings of the 7th International Meshing Roundtable

[29] Piegl L, Tiller W (1995) The NURBS book. Monographs in visual communication, Springer, New York

[30] Reali A (2006) An isogeometric analysis approach for the study of structural vibrations. Journal of Earthquake Engineering 10(1):1

[31] Shimada K (1993) Physically-based mesh generation: automated triangulation of surfaces and volumes via bubble packing. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA

[32] Simkins DC, Li S, Lu H, Liu WK (2004) Reproducing kernel element method Part IV. Globally compatible $C^n(n \geq 1)$ triangular hierarchy. Computer Methods in Applied Mechanics and Engineering 193:1013–1034

[33] Simkins DC Jr (2004) General reproducing kernel element hierarchies. PhD thesis, University of California, Berkeley, CA

[34] Simkins DC Jr, Kumar A, Collier N, Whitenack L (2007) Geometry representation, modification and iterative design using RKEM. Computer Methods in Applied Mechanics and Engineering 196:4304–4320

[35] Vartziotis D, Athanasiadis T, Goudas I, Wipper J (2008) Mesh smoothing using the geometric element transformation method. Computer Methods in Applied Mechanics and Engineering 197:3760–3767

[36] Watson DF (1981) Computing the delaunay tesselation with application to voronoi polytopes. The Computer Journal 24(2):167–172

[37] Whitenack L (2008) The biomechanics and evolution of shark teeth. PhD thesis, University of South Florida, Tampa, FL

[38] Zienkiewicz O, Taylor R (2000) The finite element method, vol 1, 5th Edition. Butterworth-Heinemann, Oxford

ABOUT THE AUTHOR

Nathan got his B.S. in Mechanical Engineering from the University of South Florida in 2001 and his masters also in 2003 working under Dr. Autar Kaw. From 2003 to 2006 he worked with Dr. Stan Kranc as a research associate on projects for the Florida Department of Transportation until May of 2006 when he began his Ph.D. studies with Dr. Simkins in the Civil and Environmental Department at USF. His area of study is in Computational Mechanics, with emphasis in meshfree and hybrid methods.