Stable Mesh Decimation

Chandrajit Bajaj
Department of Computer
Sciences
University of Texas at Austin
bajaj@cs.utexas.edu

Andrew Gillette
Department of Mathematics
University of Texas at Austin
agillette@math.utexas.edu

Qin Zhang
Department of Computer
Sciences
University of Texas at Austin
zqyork@ices.utexas.edu

ABSTRACT

Current mesh reduction techniques, while numerous, all primarily reduce mesh size by successive element deletion (e.g. edge collapses) with the goal of geometric and topological feature preservation. The choice of geometric error used to guide the reduction process is chosen independent of the function the end user aims to calculate, analyze, or adaptively refine. In this paper, we argue that such a decoupling of structure from function modeling is often unwise as small changes in geometry may cause large changes in the associated function. A stable approach to mesh decimation, therefore, ought to be guided primarily by an analysis of functional sensitivity, a property dependent on both the particular application and the equations used for computation (e.g. integrals, derivatives, or integral/partial differential equations). We present a methodology to elucidate the geometric sensitivity of functionals via two major functional discretization techniques: Galerkin finite element and discrete exterior calculus. A number of examples are given to illustrate the methodology and provide numerical examples to further substantiate our choices.

1. INTRODUCTION

For function computations carried out on large meshes, mesh decimation is an essential first step. Mesh decimation techniques are distinguished by the *cost function* they attempt to minimize as they collapse edges in the mesh. In this paper, we show that given a particular partial differential equation (PDE) problem, an analysis of the geometric sensitivity of the functions involved should guide the choice of cost function for pre-computation decimation.

We consider such function-guided decimation in two realms: adaptive finite element methods (AFEM) and discrete exterior calculus (DEC) methods. There are three main steps in such methods: formulating a weak version of the governing PDEs, discretizing the problem, and solving the resulting linear system. Each step introduces a different type of error to the process. Formulating a weak problem may create what is known as *model error*. Reducing to a linear system causes *discretization error*. Implementing the numerical method inevitably causes some *solver error*.

Adaptive finite element methods aim to control solver error by selective local refinement of the input mesh (h-adaptivity), the degree of the basis functions (p-adaptivity), or both (hp-adaptivity). While each flavor of adaptive method has met success in particular applications, we will show that their applicability does not immediately transfer to problems that require mesh decimation as a pre-processing step. Mesh decimation causes a certain loss of geometric information while adaptive refinement is an approximation of missing function information. For this reason, it is important that the loss of *function* information accrued during mesh decimation be bounded a priori so that the adaptive method can have a hope of converging to a meaningful result.

Discrete exterior calculus methods control solver error by discretizing the functions and operators of the PDE with respect to their algebraic relationships. This type of analysis leads to specific conclusions about where values of the load data and solution data should be assigned or computed; in many cases, values belong most naturally somewhere other than mesh vertices, e.g. on mesh edges or at the circumcenters of triangles. Therefore, an error bound on function information loss for mesh decimation prior to a DEC method must, by necessity, take into account the locations of the samples of the various variables in the problem.

We describe a framework for selecting an appropriate mesh decimation technique given a PDE and an approach to solving it. The discretization from the AFEM or DEC method yields a linear system of the form $\mathbb{A}\mathbf{x} = \mathbf{b}$ whose solution requires inverting the matrix $\mathbb{A}$. Our contention is that mesh decimation should be guided by an attempt to avoid large entries in the matrix $\mathbb{A}$ which can make $\mathbb{A}$ ill-conditioned and hence destabilize the numerical method.

In Section 2, we discuss prior work on AFEM, DEC, and mesh decimation. In Section 3, we first give a general overview of AFEM and DEC methods and then explain how each can suggest a mesh decimation technique through a variety of examples. In Section 4, we describe the cost functions associated to two existing techniques as well two novel cost functions for use in molecular solvation energetics computations. In Section 5, we present initial experimental results comparing our technique to prior ones.

2. PRIOR WORK

We discuss the three main topics of prior work related to our approach: adaptive finite element methods, discrete exterior calculus methods, and mesh decimation methods.

Finite element methods (FEM) have witnessed an explosive growth both in the literature and in industrial application in the past few decades. Adaptive methods [2] have gained traction for their ability to increase local accuracy in a solution. Beginning with a coarse mesh, AFEM refine by subdividing certain elements into smaller pieces (h-adaptivity) [16], increasing the degree of polynomial approximation on some elements (p-adaptivity) [3], or a combination of the two (hp-adaptivity) [14]. Recently AFEM have been applied to computational biology disciplines. Baker et al. have worked on a parallel implementation of an AFEM to solve the Poisson-Boltzmann equation [8] which generates a mesh of the molecular surface via a subdivision scheme. Recent work by Chen et al. [12] provides a FEM for the nonlinear Poisson-Boltzmann equation with rigorous convergence estimates.

Discrete Exterior Calculus (DEC) is an attempt to create from scratch a discrete theory of differential geometry and topology whose definitions and theorems mimic their smooth counterparts. This theoretical foundation allows the canonical prescription of a discretization scheme for a given PDE problem that enforces topological constraints combinatorially instead of numerically, thereby providing for increased robustness in implementation. This approach has been employed by an increasing number of authors in recent years to develop multigrid solvers [9], solve Darcy flow problems [22], and geometrize elasticity [29]. For a complete introduction to DEC theory, see Hirani [21] and Desbrun et al. [15]. In this paper, we give a brief introduction to the theory in Section 3 and an example in Section 3.3.

Mesh decimation techniques are abundant in geometry processing literature. A useful survey of many methods was given by Heckbert and Garland [20] and a more recent book by Luebke et al. [27] provides a thorough overview of mesh simplification techniques. Bajaj and Schikore [6] have given an error bounded mesh decimation technique for 2D scalar field data. We focus on mesh decimation of unstructured surface meshes via edge contraction including the approaches of Garland and Heckbert [18] and Lindstrom and Turk [23, 24]. These are explained in Section 4.

3. METHODOLOGY FOR ELUCIDATING GEOMETRY SENSITIVE FUNCTIONALS

We begin with a general problem: find $u \in V$ such that

$$Lu = f \quad \text{on } \Omega, \qquad (1)$$

where L is a linear operator, V is the appropriate solution space for the problem, and Ω is a simplicial complex embedded in $\mathbb{R}^3$. There are two main techniques used to discretize this into a linear system: Galerkin methods and Discrete Exterior Calculus (DEC) methods. We describe how each could be used and how a mesh decimation technique should be chosen accordingly.

The Galerkin finite element method begins by putting (1)

into the weak form: find $u \in V$ such that

$$a(u, v) = (f, v)_{L^2} \quad \forall v \in V, \qquad (2)$$

where a is the operator L phrased as a bilinear form (usually symmetric) and f is treated as a functional $(f, \cdot)_{L^2}$ on V. An appropriate finite dimensional subspace $V_h \subset V$ is chosen and an answer to the following discretized problem is sought: find $u \in V_h$ such that

$$a(u, v) = (f, v)_{L^2} \quad \forall v \in V_h.$$

Since V_h is finite-dimensional, we can fix a basis $\{\phi_i : 1 \leq i \leq n\}$ of V_h. The size of the basis is proportional to the number of elements in the mesh Ω. Write $u = \sum_{j=1}^{n} U_j \phi_j$, $K_{ij} = a(\phi_j, \phi_i)$ and $F_i = (f, \phi_i)$. Set $\mathbf{U} = (U_j)$, $\mathbb{K} = (K_{ij})$ $\mathbf{F} = (F_i)$. Then solving (2) over V_h is the same as solving the matrix equation

$$\mathbb{K}\mathbf{U} = \mathbf{F}. \qquad (3)$$

For a proof and detailed discussion, see [11].

A significant amount of care goes into the selection of V_h to ensure that the method is both well-posed and stable. "Well-posed" means the system has a unique solution and "stable" means there exists a constant $C > 0$ independent of h such that

$$||u - u_h||_V \leq C \inf_{w_h \in V_h} ||u - w_h||_V.$$

In other words, a method is stable if the error between the true solution u and approximate solution u_h is bounded above uniformly by a constant multiple of the minimal approximation error for V_h. The famous Babuska inf-sup condition [1] is often used to simultaneously prove both well-posedness and stability of a FEM and hence provide an a priori bound on solver error. We note, however, that the stability error bound does not account for error due to naive mesh decimation.

If the mesh Ω has too many elements, (3) will be too large for the solver, making decimation necessary. For decimation to be useful, however, it must not create very large or small entries in $\mathbb{K}$ which might make $\mathbb{K}$ ill-conditioned. Since the entries of $\mathbb{K}$ are a functional $a(\cdot, \cdot)$ on the basis functions ϕ_i, mesh decimation must be guided by the geometry-sensitive components of a as opposed to the geometry of Ω alone. Such components are necessarily problem-specific as we elucidate in examples presented in Sections 3.1 and 3.2.

The method of Discrete Exterior Calculus is an alternative approach which focuses on correctly discretizing the operator L instead of the solution space V. The viewpoint provided by differential geometry and topology reveals how this ought to be done. Common operators such as grad, curl, and div are all manifestations of the exterior derivative operator d in dimensions 1, 2, and 3, respectively. Equations relating quantities of complementary dimensions, such as the constitutive relations in Maxwell's equations, involve a Hodge star operator $*$ which provides the canonical mapping. The Laplacian operator Δ can be written as $\delta d + d\delta$ where δ is the coderivative operator, defined by $\delta := *d*$. Each operator has a discrete version designed to mimic the properties of its smooth counterpart. The discrete versions of the operators are written as matrices whose entries depend only on the topology and geometry of the mesh Ω.

To solve (1), an analysis is made as to the dimension of u as a k-form based on either the problem context or the type of operator acting on it. The div operator in 3D, for example, acts on 2-forms while grad acts on 0-forms. The variable u is replaced by a vector $\vec{u}$ with one entry for each k-simplex in the mesh of Ω and the operator L is replaced by its discrete counterpart, written as a matrix $\mathbb{L}$. The load data f is converted to a vector $\vec{f}$ accordingly. This yields the equation

$$\mathbb{L}\vec{u} = \vec{f}, \qquad (4)$$

which can then be solved by linear methods.

Again, it may be necessary to decimate Ω so that the linear system (4) is tractable on a computer. The size of the entries of $\mathbb{L}$ depend heavily on the geometry-sensitive operators such as $*$ and δ and less on the topology sensitive operators such as d. Therefore, to prevent an ill-conditioned $\mathbb{L}$, mesh decimation must be guided based on the definition of the discrete operators as opposed to the definition of the solution space. We discuss an example in Section 3.3.

3.1 Poisson-Boltzmann Electrostatics

The Poisson-Boltzmann equation (PBE) describes the attraction between solvated molecules. We describe its linearized version according to the formulation given by Lu, Zhang, and McCammon in [26], which is believed to be a sufficient approximation for electrostatics computations. Let $\Omega \subset \mathbb{R}^3$ be a domain indicating interior molecular regions with point charges $q_1, \ldots, q_N$ located at $r_1, \ldots, r_N \in \Omega$. The linear PBE is

$$\nabla^2 \phi^{\text{int}}(r_p) = -\frac{1}{\epsilon_{\text{int}}} \sum_{k=1}^{N} q_k \delta(r_p - r_k), \quad p \in \Omega, \qquad (5)$$

$$\nabla^2 \phi^{\text{ext}}(r_p) = \kappa^2 \phi^{\text{ext}}(r_p), \quad p \in \Omega, \qquad (6)$$

where ϕ^{int} and ϕ^{ext} are the electrostatic potentials on the interior and exterior of Ω, ϵ_{int} is the interior dielectric constant, δ is the Dirac distribution, and κ is the inverse of the Debye-Hückel screening length. Values for the constants ϵ_{int} and κ are determined experimentally. At the boundary $\partial\Omega$, the surface potential f should satisfy $f = \phi^{\text{int}} = \phi^{\text{ext}}$ with normal derivative $h = \partial \phi^{\text{ext}}/\partial n$. The equations are put into integral form and discretized, reducing the problem to a set of linear equations of the form

$$\begin{pmatrix} \mathbb{B} & \mathbb{A} \\ \mathbb{D} & \mathbb{C} \end{pmatrix} \begin{pmatrix} f \\ h \end{pmatrix} = \begin{pmatrix} Q \\ 0 \end{pmatrix} \qquad (7)$$

where Q is the initial data of point charges and locations. The four (sub)matrices $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$ and $\mathbb{D}$ have entries

$$\sum_t \int_{E_t} \mathcal{I} dA$$

where E_t is a facet of Ω, t indexes over a small neighborhood of mesh elements, and the integrand $\mathcal{I}$ is one of the following:

$$\mathcal{I} \in \left\{ G(x_i, x_j), \frac{\partial G(x_i, x_j)}{\partial n}, u(x_i, x_j), \frac{\partial u(x_i, x_j)}{\partial n} \right\}.$$

The functions G and u are the Green functions for (5) and (6), respectively. They are given by

$$G(x_i, x_j) = \frac{1}{4\pi r_{ij}} \quad \text{and} \quad u(x_i, x_j) = \frac{\exp(-\kappa r_{ij})}{4\pi r_{ij}},$$

where $r_{ij} = |x_i - x_j|$. Hence, the terms of the submatrices in (7) decay like $1/r_{ij}$ at worst. To avoid a blowup of these terms, we use a cost function f_{pb} that attempts to preserve mesh element quality as a way of avoiding small r_{ij} values. We describe f_{pb} in Section 4.3.

3.2 Generalized Born Electrostatics

A recent approach by Bajaj and Zhao [7] computes molecular solvation energetics and forces by using a Generalized Born (GB) model instead of a Poisson Boltzmann model of electrostatic solvation. While the PB model begins with the PBE, a description of the electrostatic potential over the whole domain, the GB model begins with a model of the solvation energy of a single atom in a given medium. The electrostatic solvation energy of a molecule is defined in terms of the pairwise interaction between these atomic energies:

$$G_{\text{pol}} = -\frac{\tau}{2} \sum_{i,j} \frac{q_i q_j}{[r_{ij}^2 + R_i R_j \exp(-r_{ij}^2/4R_i R_j)]^{1/2}}.$$

Here, $\tau = \frac{1}{\epsilon_p} - \frac{1}{\epsilon_w}$ where ϵ_p and ϵ_w are the solute and solvent dielectric constants, q_i and R_i are the charge and effective Born radius of atom i, and r_{ij} is the distance between atom i and atom j. The success of a GB method hinges upon efficient and accurate approximation of the effective Born radii R_i. Bajaj and Zhao use the surface integration technique given in [19]. This gives the expression

$$R_i^{-1} = \frac{1}{4\pi} \int_{\Gamma} \frac{(\mathbf{r} - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} dS, \quad i = 1, \ldots, M,$$

where Γ is the solvent-molecular interface, $\mathbf{x}_i$ is the center of atom i, and $\mathbf{n}(\mathbf{r})$ is the unit normal of the surface at $\mathbf{r}$. The position vector $\mathbf{r}$ ranges over Γ. The integral is approximated by

$$R_i^{-1} \approx \frac{1}{4\pi} \sum_{k=1}^{N} w_k \frac{(\mathbf{r}_k - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4} dS, \quad i = 1, \ldots, M,$$

where the $\mathbf{r}_k$ are the Gaussian quadrature nodes with weights w_k lying on a triangular mesh approximating the surface Γ. Therefore, the computation of R_i^{-1} will be sensitive to changes in the position of Gauss points relative to the nearest atomic centers, i.e. changes in the computed values of $|\mathbf{r} - \mathbf{x}_i|$. Accordingly, we design a cost function f_{gb} to minimize the cumulative change in $|\mathbf{r} - \mathbf{x}_i|$ values. This function is described in Section 4.4 and compared experimentally to other cost functions in Section 5.

3.3 Darcy Flow

Recent work by Hirani et al. [22] uses a DEC method to model Darcy flow, a description of the flow of a viscous fluid in a permeable medium. The governing equations under the assumption of no external body force are given by

$$\begin{aligned} f + \tfrac{k}{\mu} \nabla p &= 0 & \text{in } \Omega, \\ \operatorname{div} f &= \phi & \text{in } \Omega, \\ f &= \psi & \text{on } \partial\Omega, \end{aligned}$$

where f is the volumetric flux, $k > 0$ is the coefficient of permeability, $\mu > 0$ is the coefficient of viscosity, $\phi : \Omega \to \mathbb{R}$ is the prescribed divergence of velocity, and $\psi : \partial\Omega \to \mathbb{R}$ is the prescribed normal component of the velocity across the
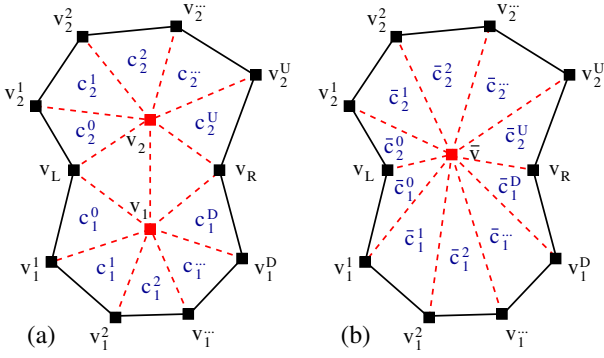
Figure 1: Notation for collapse of edge (v_1, v_2) to $\overline{v}$. Black vertices and edges are unchanged in the collapse while red vertices and dashed edges may change.

boundary. They discretize the operators ∇ and div based on DEC theory to arrive at the linear system

$$\begin{bmatrix} -(\mu/k)\mathbb{M}_{n-1} & \mathbb{D}_{n-1}^T \\ \mathbb{D}_{n-1} & 0 \end{bmatrix} \begin{bmatrix} f \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ \phi \end{bmatrix}.$$

Here, $\mathbb{D}_k$ is the discrete exterior derivative operator that acts on k-cochains and $\mathbb{M}_k$ is a diagonal matrix representing the Hodge Star operator on k-cochains.

We now consider the effect of a priori mesh decimation for this scheme. The DEC analysis used to derive this method requires that the solution $[f \; p]^T$ provide values of the flux f on $(n-1)$-simplices (i.e. edges in triangle meshes and triangles in tetrahedral meshes) and values of the pressure p at the circumcenters of n-simplices.

For the pressure values to have any meaning, the mesh must be *well-centered* meaning the circumcenter of each simplex must lie in the interior of the simplex. Since this criterion is often violated by meshing schemes (e.g. an obtuse triangle is not well-centered), pressure is assigned instead to the barycenters of n-simplices. The authors point out that the error introduced by this modification prevents the exact representation of linear variation of pressure over the domain. Therefore, an appropriate cost function for this method should be weighted to favor the creation of simplified meshes with good quality elements (e.g. elements with good aspect ratios). This would minimize the distance between the barycenter and circumcenter, making the calculations more robust.

4. DESCRIPTION OF COST FUNCTIONS
4.1 Quadratic Error Cost Function
The quadric error measure we use comes from Garland and Heckbert [18]. First, an error metric $\Delta(\mathbf{v})$ is established for each vertex $\mathbf{v}$, based on the planes $P(\mathbf{v})$ that contain the triangles incident to $\mathbf{v}$. A plane $\mathbf{p} \in P(\mathbf{v})$ given by $ax + by + cz + d = 0$ is represented as $[a \; b \; c \; d]^T$. The vertex $\mathbf{v}$ is represented as $[v_x \; v_y \; v_z \; 1]^T$. Then the error metric $\Delta(\mathbf{v})$ is defined by

$$\Delta(v) = \mathbf{v}^T \left(\sum_{\mathbf{p} \in P(\mathbf{v})} \mathbf{p}\mathbf{p}^T \right) \mathbf{v}.$$

For a vertex $\mathbf{v}_i$, let $\mathbb{Q}_i = \sum_{\mathbf{p} \in P(\mathbf{v}_i)} \mathbf{p}\mathbf{p}^T$. Then the cost of collapsing edge $(\mathbf{v}_1, \mathbf{v}_2)$ to some point $\overline{\mathbf{v}}$ is

$$f_{qe}(\mathbf{v}_1, \mathbf{v}_2; \overline{\mathbf{v}}) := \overline{\mathbf{v}}^T (\mathbb{Q}_1 + \mathbb{Q}_2) \overline{\mathbf{v}}.$$

We use the publicly available software QSlim [17] to implement this cost function. For a given edge, the program attempts to find an optimal placement of $\overline{\mathbf{v}}$ by solving a certain linear system derived from the $\mathbb{Q}_i$. If the matrix associated to this system is not invertible, it tries to place $\overline{\mathbf{v}}$ optimally on $(\mathbf{v}_1, \mathbf{v}_2)$. If this fails, it sets $\overline{\mathbf{v}}$ to be either $\mathbf{v}_1$, $\mathbf{v}_2$, or the midpoint of the edge, whichever minimizes f_{qe}.

4.2 Volumetric Error Cost Function
The volumetric error measure we use comes from Lindstrom and Turk [23, 24]. First, we establish the notation for the signed volume V of a tetrahedron bounded by a vertex $\mathbf{v}$ and the vertices $\mathbf{v}_0^{t_i}, \mathbf{v}_1^{t_i}, \mathbf{v}_2^{t_i}$ of a triangle t_i. As before, vertices are written as four component vectors, e.g. $\mathbf{v}$ is represented by $[v_x \; v_y \; v_z \; 1]^T$. Then V is defined by

$$V(\mathbf{v}, \mathbf{v}_0^{t_i}, \mathbf{v}_1^{t_i}, \mathbf{v}_2^{t_i}) = \frac{1}{6} \det(\mathbf{v} \; \mathbf{v}_0^{t_i} \; \mathbf{v}_1^{t_i} \; \mathbf{v}_2^{t_i}) =: \frac{1}{6} \mathbf{G}_{t_i} \mathbf{v}$$

where $\mathbf{G}_{t_i}$ is a 1 by 4 matrix defined by the above equation. The cost of collapsing edge $(\mathbf{v}_1, \mathbf{v}_2)$ to some point $\overline{\mathbf{v}}$ is

$$f_{vol}(\mathbf{v}_1, \mathbf{v}_2; \overline{\mathbf{v}}) := \frac{1}{2} \overline{\mathbf{v}}^T \left(\frac{1}{18} \sum_i \mathbf{G}_{t_i}^T \mathbf{G}_{t_i} \right) \overline{\mathbf{v}},$$

where i indexes over triangles t_i incident to at least one of $\{\mathbf{v}_1, \mathbf{v}_2\}$. This cost function is ultimately quite similar to f_{qe} except that f_{qe} weights the distance between $\overline{\mathbf{v}}$ and a plane by the area of the triangle defining the plane while f_{vol} weights it by the *square* of the triangle area. As is explained in [23, 24], this weighting better serves the goal of volume preservation. We use a package from the publicly available software TeraScale Browser [25] to implement this cost function.

4.3 Poisson Boltzmann Cost Function
In Section 3.1 we discuss how Poisson Boltzmann (PB) computations are sensitive to Gaussian quadrature points coming into close proximity. Hence, we define a cost function f_{pb} which penalizes such occurrences.

We fix the following notation for the collapse of edge $(\mathbf{v}_1, \mathbf{v}_2)$ to the point $\overline{\mathbf{v}}$. Consider the union of triangles incident to $\mathbf{v}_1$ or $\mathbf{v}_2$. These are the only vertices, edges, and triangles whose geometry may be changed by the edge collapse. Although all these objects lie in $\mathbb{R}^3$, their generic connectivity information is captured in $\mathbb{R}^2$ by Figure 1 (a). Taking $(\mathbf{v}_1, \mathbf{v}_2)$ to be vertical with $\mathbf{v}_1$ at the bottom, the triangle to the left (resp. right) of the edge has $\mathbf{v}_L$ ($\mathbf{v}_R$) as its third vertex. We proceed from $\mathbf{v}_L$ to $\mathbf{v}_R$ along the upper (resp. lower) vertices labeling them $\mathbf{v}_2^1, \mathbf{v}_2^2, \dots, \mathbf{v}_2^U$ ($\mathbf{v}_1^1, \mathbf{v}_1^2, \dots, \mathbf{v}_1^D$ (D for down)). Set $\mathbf{v}_2^0 := \mathbf{v}_1^0 := \mathbf{v}_L$ and $\mathbf{v}_2^{U+1} := \mathbf{v}_1^{D+1} := \mathbf{v}_R$. We denote the centers of the triangles not adjacent to $(\mathbf{v}_1, \mathbf{v}_2)$ as

$$c_2^u := \frac{1}{3} \left(\mathbf{v}_2 + \mathbf{v}_2^u + \mathbf{v}_2^{u+1} \right), \quad u = 0, 1, \dots, U,$$

$$c_1^d := \frac{1}{3} \left(\mathbf{v}_1 + \mathbf{v}_1^d + \mathbf{v}_1^{d+1} \right), \quad d = 0, 1, \dots, D.$$

The collapse operation moves $\mathbf{v}_1$ and $\mathbf{v}_2$ to $\overline{\mathbf{v}}$. The result is shown in Figure 1 (b). The $\{\mathbf{v}_1^d\}$ and $\{\mathbf{v}_2^u\}$ are unchanged, but the new centers are $\{\bar{c}_2^u\}$ and $\{\bar{c}_1^d\}$ where $\overline{\mathbf{v}}$ replaces $\mathbf{v}_2$ or $\mathbf{v}_1$ in the expression of the center. Re-index c_1^d and c_2^u as c_i.

If the triangles indexed by the c_i are of good quality, meaning nearer-to-equilateral, their Gaussian quadrature points will be better spaced. We approximate the quality of a triangle T by

$$q(T) := \frac{l(T)}{s(T)} + \frac{maxa(T)}{mina(T)},$$

where $l(T)$ (resp. $s(T)$) denotes the longest (shortest) side and $maxa(T)$ (resp. $mina(T)$) denotes its maximum (minumum) angle. The best quality triangles have the minimum q value of 2. We denote triangles in Figure 1 by their center c_i or $\bar{c}_i$. The cost function is then defined to be

$$f_{pb}(\mathbf{v}_1, \mathbf{v}_2; \overline{\mathbf{v}}) := \sum_i q(T_{\bar{c}_i}) - q(T_{c_i}).$$

To further improve element quality, we decimate in stages and run a quality improvement code based on geometric flow [28] in between stages.

4.4 Generalized Born Cost Function

In Section 3.2 we discuss how the Generalized Born (GB) computations are sensitive to changes in the location of Gaussian quadrature points $\{c_i\}$ relative to the atomic centers $\{x_j\}$ of the molecule in question. Hence, we define a cost function f_{gb} which penalizes edge collapses with a larger cumulative change in $|c_i - x_j|$ values. Since it would be too computationally expensive to compute the complete change for every edge collapse, we use a restricted set of pertinent $\{c_i\}$ and $\{x_j\}$ values described below.

We take the $\{c_i\}$ described in Section 4.3 as our set of pertinent Gaussian quadrature points. We set $\{x_j\}$ to be those atomic centers lying within a fixed distance ρ of either v_1 or v_2. Since Born radii are on the order of 1-2 Å, ρ should be set between 2 and 5 to capture a manageable, non-empty set of nearby atoms. In the future, we will devise a parameter sweep to optimize the value of ρ.

We want to minimize the atomic center functional f_{ac}

$$f_{ac} := \left| \sum_{i,j} |c_i - x_j|^2 - |\bar{c}_i - x_j|^2 \right|,$$

which can be re-written as

$$f_{ac} = \left| \sum_{i,j} c_i^T c_i - \bar{c}_i^T \bar{c}_i + 2(\bar{c}_i^T - c_i^T) x_j \right|.$$

All the variables in the above expression are known, save for the $\bar{c}_i$ which are linear functions of $\overline{\mathbf{v}}$. We define the GB-dependent cost function to be

$$f_{gb}(\mathbf{v}_1, \mathbf{v}_2; \overline{\mathbf{v}}) := |\mathbf{v}_1 - \mathbf{v}_2| + \lambda f_{ac}(\mathbf{v}_1, \mathbf{v}_2; \overline{\mathbf{v}}).$$

The first term is used to promote the collapse of shorter edges and thereby improve triangle quality. The weight factor λ is chosen so that the two terms are of the same order of magnitude.
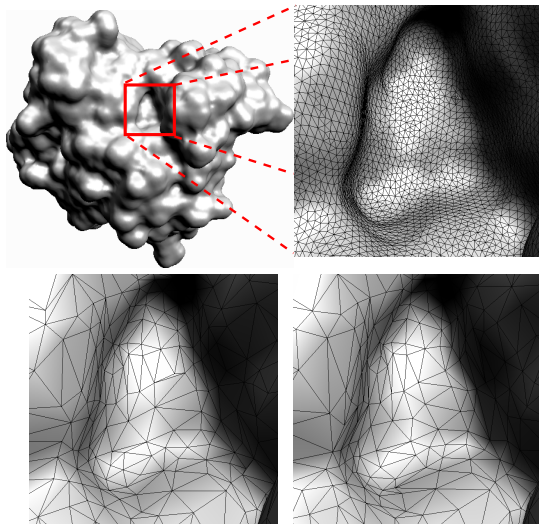

Figure 2: *Top:* **Surface rendering of mAChE before decimation. The original mesh of the surface has about 200,000 vertices and 400,000 faces. In the inset, the fine mesh is visible. This pocket region of the molecule aids in its biological function.** *Bottom:* **The mesh of the pocket after decimation to 25,500 faces using f_{qe} (left) and a modified version of f_{gb} (right).**

5. EXPERIMENTAL RESULTS AND CONCLUSIONS

To test the validity of our claims, we work with Mouse Acetylcholinesterase (mAChE). This macromolecule serves an important regulatory function as it terminates the action of the neurotransmitter acetylcholine (ACh). The initial mesh of the molecular surface is generated from a Protein Data Bank (PDB) [10] file of the molecule using our in-house software TexMol [13]. We show a picture of the initial mesh in Figure 2. We decimate this mesh using the cost function f_{qe} and a modified version of f_{gb} which uses f_{qe} instead of $|\mathbf{v}_1 - \mathbf{v}_2|$. We use a Dynamic Packing Grid data structure [5] to efficiently compute the set $\{x_j\}$ of nearby centers for each mesh edge. We set $\rho = 5$ and $\lambda = 10^{-8}$ and compute the polarized and non-polarized energy for each mesh using the nFFGB code described in [7]. The meshes are only marginally different as shown in Figure 2 and thus produce similar energy values as shown in the charts in Figure 3. With further experimentation and parameter sweeps, we believe f_{gb} will begin to out-perform f_{qe}. Still, Figure 3 shows that f_{vol} is a decidedly worse choice for non-polarized energy computations as it does not respect the functional sensitivity of the problem. In future work, we will also implement f_{pb} and use PB-CFM code by Bajaj and Chen [4] to compute and compare PB energetics.
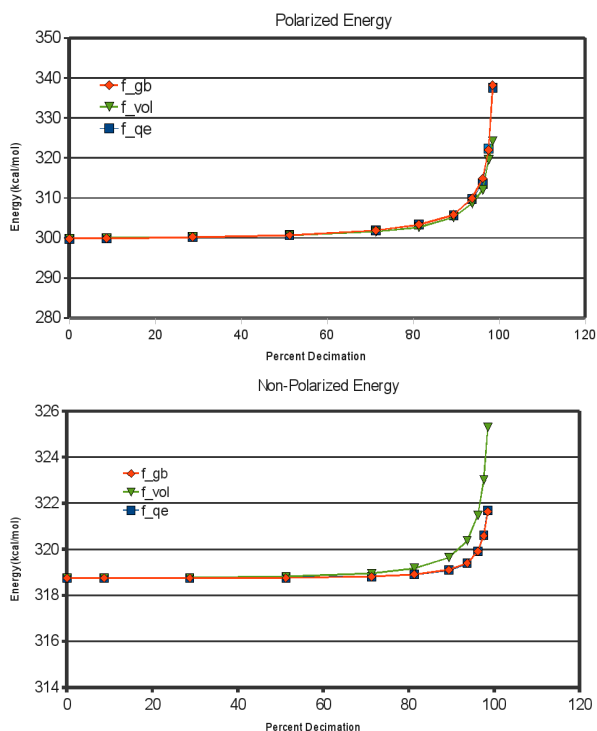
6. ACKNOWLEDGMENTS

Figure 3: Effect of cost function used for decimation on computed GB energy values for mAChE.

7. REFERENCES

[1] I. Babuska and A. Aziz. Survey lectures on the mathematical foundations of the finite element method. In *The Mathematical Foundations of the FEM with Applications to PDEs, Proc. Sympos.*, 1972.

[2] I. Babuška, J. Chandra, and J. E. Flaherty. *Adaptive Computational Methods for Partial Differential Equations.* SIAM, Philadelphia, PA, USA, 1983.

[3] I. Babuška, O. C. Zienkiewicz, J. Gago, and E. R. de A. Oliveira. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations.* John Wiley and Sons, Chichester, 1986.

[4] C. Bajaj and A. Chen. Efficient and accurate higher-order fast multipole bem for poisson-boltzmann electrostatics. *SIAM J. on Sci. Comp.*, Submitted.

[5] C. Bajaj, R. Chowdhury, and M. Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. In *SIAM/ACM GDSPM09*, Accepted.

[6] C. Bajaj and D. Schikore. Topology preserving data simplification with error bounds. *Computers and Graphics*, 22:3–12(10), 25 February 1998.

[7] C. Bajaj and W. Zhao. Fast molecular solvation energetics and forces computation. *SIAM J. Sci. Comp.*, Submitted.

[8] N. A. Baker, D. Sept, M. J. Holst, and J. A. McCammon. The adaptive multilevel finite element solution of the Poisson-Boltzmann equation on massively parallel computers. *J. Comput. Chem*, 21, 2000.

[9] W. N. Bell. Algebraic multigrid for discrete differential forms (dissertation). Technical report, University of Illinois at Urbana-Champaign, 2008.

[10] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The Protein Data Bank. *Nucleic Acids Research*, pages 235–242, 2000.

[11] S. Brenner and L. Scott. *The Mathematical Theory of Finite Element Mehtods.* Springer-Verlag, New York, 2002.

[12] L. Chen, M. J. Holst, and J. Xu. The finite element approximation of the nonlinear poisson-boltzmann equation. *SIAM J. Numer. Anal.*, 45(6):2298–2320, 2007.

[13] CVC. TexMol. *http://ccvweb.csres.utexas.edu/ccv/ projects/project.php?proID=8*.

[14] L. Demkowicz. *Computing with hp-adaptive finite elements.* Chapman and Hall / CRC, 2007.

[15] M. Desbrun, A. N. Hirani, M. Leok, and J. E. Marsden. Discrete Exterior Calculus. *arXiv:math/0508341*, 2005.

[16] J. E. Flaherty, M. Shepard, P. Paslow, and D. Vasilakis. *Adaptive Methods for Partial Differential Equations.* SIAM, Philadelphia, PA, USA, 1989.

[17] M. Garland. QSlim. *http://graphics.cs.uiuc.edu/ ∼garland/software/qslim.html*, 2004.

[18] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97*, pages 209–216, New York, NY, USA, 1997.

[19] A. Ghosh, C. S. Rapp, and R. A. Friesner. Generalized born model based on a surface integral formulation. *The Journal of Physical Chemistry B*, 102(52):10983–10990, 1998.

[20] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical report, Carnegie Mellon University, 1995.

[21] A. N. Hirani. Discrete exterior calculus (dissertation). Technical report, Cal Tech, 2003.

[22] A. N. Hirani, K. B. Nakshatrala, and J. H. Chaudhry. Numerical method for Darcy flow derived using Discrete Exterior Calculus. *arXiv:0810.3434*, 2008.

[23] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *VIS '98*, pages 279–286, 1998.

[24] P. Lindstrom and G. Turk. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):98–115, 1999.

[25] LLNL. TeraScale Browser. *https://computing.llnl.gov/vis/terascale.shtml*, 2007.

[26] B. Lu, D. Zhang, and J. A. McCammon. Computation of electrostatic forces between solvated molecules determined by the poisson–boltzmann equation using a boundary element method. *Journal of Chemical Physics*, 122(21):214102–1–7, 2005.

[27] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney. *Level of Detail for 3D Graphics.* Elsevier Science Inc., New York, 2002.

[28] G. Xu, Q. Pan, and C. L. Bajaj. Discrete surface modelling using partial differential equations. *CAGD*, 23(2):125–145, 2006.

[29] A. Yavari. On geometric discretization of elasticity. *Journal of Mathematical Physics*, 49(2):1–36, 2008.