On Interpolation Errors over Quadratic Nodal Triangular Finite Elements

Shankar P. Sastry and Robert M. Kirby

University of Utah, Salt Lake City, UT 84112
{sastry,kirby}@sci.utah.edu

Summary. Interpolation techniques are used to estimate function values and their derivatives at those points for which a numerical solution of any equation is not explicitly evaluated. In particular, the shape functions are used to interpolate a solution (within an element) of a partial differential equation obtained by the finite element method. Mesh generation and quality improvement are often driven by the objective of minimizing the bounds on the error of the interpolated solution. For linear elements, the error bounds at a point have been derived as a composite function of the shape function values at the point and its distance from the element's nodes. We extend the derivation to quadratic triangular elements and visualize the bounds for both the function interpolant and the interpolant of its derivative. The maximum error bound for a function interpolant within an element is computed using the active set method for constrained optimization. For the interpolant of the derivative, we visually observe that the evaluation of the bound at the corner vertices is sufficient to find the maximum bound within an element. We characterize the bounds and develop a mesh quality improvement algorithm that optimizes the bounds through the movement (r-refinement) of both the corner vertices and edge nodes in a high-order mesh.

1 Introduction

Interpolation techniques are used to estimate the solutions of partial differential equations (PDEs) at those points for which the equations have not been explicitly solved. Mesh generation is partly driven by the estimation of the errors for those interpolation techniques[1]. In the finite element method (FEM), linear, quadratic, or higher-order basis expansions of the solution are used

[1] The optimization of the discretization error and improvement of the conditioning of the stiffness matrix are some of the other factors that play a vital role in mesh generation.

J. Sarrate & M. Staten (eds.), *Proceedings of the 22nd International Meshing Roundtable*,
DOI: 10.1007/978-3-319-02335-9_20, © Springer International Publishing Switzerland 2013

on an appropriate mesh, and a corresponding interpolation technique is used to estimate the solution within an element. Thus, a characterization of error bounds for all types of mesh elements is necessary to aid high-quality mesh generation. In this paper, we derive and characterize the errors bounds associated with quadratic triangular straight-sided finite elements and provide an algorithm to optimize the error bounds.

Numerous papers [1, 2, 3, 4] that discuss *a priori* and *a posteriori* error bounds or their estimates have been published for linear elements. Notably, Shewchuk [5] provided a comprehensive characterization of quality metrics for linear elements. We discuss some of these prior results in Section 2. For quadratic elements, however, only a limited understanding of the behavior [6] of error bounds is known. In this paper, we extend the understanding of the error bounds to include the analysis of interpolation error for quadratic straight-sided triangular elements. We discuss the background concepts used in this paper in Section 3 and derive the error bounds in Section 4.

We visualize the error bounds within a quadratic triangular element and also the quality metrics obtained by normalizing of the error bounds with respect to the area of the triangle. The visualizations help us develop an algorithm to compute the error bounds and improve it through optimization-driven vertex movement (r-refinement) techniques. We provide the visualizations in Section 5 and the details of the implementation of the mesh quality improvement algorithm in Section 6.

We carry out some numerical experiments to determine the effects of the optimization-driven vertex movement algorithm on the error bounds. We discuss the results of the experiments in Section 7. Finally, we conclude the paper with possible future work in Section 8.

2 Related Work

The estimation of error bounds can be broadly classified into the following two main categories: *a priori* and *a posteriori* error estimation. An *a priori* estimation is carried out before any numerical simulation of a physical phenomenon takes place, and *a posteriori* error estimate is computed after some knowledge of the solution is obtained through any method. Both forms of error estimation drive the mesh generation and refinement process. In this paper, we focus on an *a priori* error estimation technique. Sometimes, the error bound is estimated through the means of a *quality metric* that is defined as a function of the geometry of an element. The quality metric may also capture the effect of the element on the conditioning of the stiffness matrix.

For linear elements, some examples of *a priori* analysis include the works of Babuška and Aziz [1], Knupp [2], Munson [3], and Baker [4]. A comprehensive analysis of the associated error estimates and quality metrics in these papers and many other such metrics is present is Shewchuk's manuscript [5].

For high-order elements, some of the quality metrics for linear elements described above are used for the purpose of mesh quality evaluation. Examples of this work include Lu *et al.* [7, 8] and Lowrie *et al.* [9]. In [7], the quality of a curved high-order element is considered to be the product of the following two quantities: (a) the quality of a straight-sided linear element with the identical location of the corner vertices, and (b), the ratio of the largest and the smallest value of the determinant of the Jacobian matrix of the physical coordinate system with respect to the parametric coordinate system. Lowrie *et al.*'s [9] paper looks at the correlation between the quality metric for a linear element and the solution accuracy for high-order meshes through a series of numerical experiments.

Although Lowrie *et al.'s* paper establishes a strong correlation between quality of a linear element and the corresponding high-order element, a theoretical basis has not been provided for the claim. Besides, the location of the additional nodes in high-order elements is fixed with respect to the barycentric coordinate system in their analysis. We believe that movement of the edge nodes in a quadratic triangular element and other additional nodes in higher-order elements can reduce the error bound in certain contexts.

For elements of very high order, several node placement techniques such as electrostatic points [10], Fekete points [11], Chen-Babuška points [12], etc. have been proposed. These techniques have not been derived from an explicit error bound formulation, but have been shown to lower the Lebesgue constant [13], which bounds the error of an interpolated function with respect to the most accurate (measured in L_∞ norm) polynomial interpolation of the same function. The vertex placement techniques have been derived by minimizing certain properties such as the electrostatic potential, condition number of the Vandermonde matrix, etc. In these papers, the analysis is carried out on a equilateral triangular high-order element. A natural question is: would the node placement be different for other triangles? We attempt to answer this question by deriving error bound for quadratic, straight-sided triangular elements and studying its characteristics.

3 Background

In this section, we present some background concepts that are used in this paper.

3.1 Barycentric Coordinates and Shape Functions

A barycentric coordinate system, denoted by $\omega_i(x)$, $1 \leq i \leq 3$, is used to describe every point in a triangle as a weighted sum of the coordinates of the triangle's vertices. In order to compute the weights, the point is joined to each of the vertices by straight lines, and the areas of three smaller triangles are computed. The weight of a vertex for the point is given by the ratio of

the signed area of a smaller triangle (that does not contain the vertex) with the total signed area of the triangle. The barycentric coordinates sum to one for every point in the plane.

Linear interpolation can be carried out using the barycentric coordinate system. Let the value of the function be f_i at vertex i of a triangle. For a point P in the triangle, let the barycentric coordinates be $\omega_i(x_P)$, $1 \leq i \leq 3$. The linearly interpolated value at P is given by $\sum_{i=1}^{3} \omega_i(x_P) f_i$.

Typically, Lagrange polynomial interpolation over a triangle is carried out by assigning weights to each of its vertices. These weights vary for every point in the triangle and are known as shape functions. The shape function for a vertex i is equal to 1 at that vertex and 0 at all other vertices of the triangle. As we infer from above, the shape functions for a linear triangle is equivalent to the barycentric coordinate system. For a quadratic triangle, the shape functions, denoted by $\lambda_i(x)$, $1 \leq i \leq 6$, are as given below:

$$\begin{aligned}
\lambda_1(x) &= \omega_1(x) * (1 - \gamma_4 * \omega_2(x) - \gamma_3 * \omega_3(x)),\\
\lambda_2(x) &= \omega_2(x) * (1 - \gamma_5 * \omega_3(x) - \gamma_1 * \omega_1(x)),\\
\lambda_3(x) &= \omega_3(x) * (1 - \gamma_6 * \omega_1(x) - \gamma_2 * \omega_2(x)),\\
\lambda_4(x) &= \gamma_1 * \gamma_4 * \omega_1(x) * \omega_2(x),\\
\lambda_5(x) &= \gamma_2 * \gamma_5 * \omega_2(x) * \omega_3(x),\\
\lambda_6(x) &= \gamma_3 * \gamma_6 * \omega_3(x) * \omega_1(x),
\end{aligned}$$

where (see Fig 1) $\gamma_1 = \frac{||A_4 - A_2||}{||A_1 - A_2||}$, $\gamma_2 = \frac{||A_5 - A_3||}{||A_2 - A_3||}$, $\gamma_3 = \frac{||A_6 - A_1||}{||A_3 - A_1||}$, $\gamma_4 = \frac{||A_4 - A_1||}{||A_1 - A_2||}$, $\gamma_5 = \frac{||A_5 - A_2||}{||A_2 - A_3||}$, and $\gamma_6 = \frac{||A_6 - A_3||}{||A_3 - A_1||}$. In this paper, vertices A_1, A_2, and A_3 called corner vertices, and vertices A_4, A_5, and A_6 are called edge nodes.

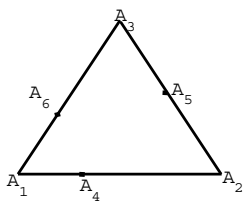


Fig. 1 A quadratic triangle with the three edge nodes that purposefully do not lie at the mid-points of the edges. In this paper, vertices A_1, A_2, and A_3 called corner vertices, and vertices A_4, A_5, and A_6 are called edge nodes.

3.2 Mesh Quality Improvement by Vertex Movement

Mesh quality improvement is carried out by moving vertices (r-refinement), by swapping edges, or by adding and deleting vertices (h-refinement). These

operations help improve the one or more of the following properties: a) interpolation error, b) conditioning of the associated stiffness matrix for solving a PDE using the FEM, c) discretization error, etc. We restrict ourselves to mesh quality improvement through vertex movement for the purpose of minimizing the bounds on the interpolation error. The movement of the vertices is dictated by a numerical optimization algorithm such that the quality of the mesh, computed using some composite function of the qualities of all the triangles in the mesh, is improved. We use the bounds on the interpolation error as the quality metric, and the nonlinear conjugate gradient algorithm is used to minimize the objective function given by $\sum_{i=1}^{m} q_i^p$, where m is the number of elements in the mesh, q_i is the quality of element i, and p is some integer. Note that a larger p penalizes poor elements more; and, thus, the quality of the worst element is more likely to be improved.

3.3 Nonlinear Conjugate Gradient Method

The nonlinear conjugate gradient method [14] is used to solve unconstrained optimization problems. We use this method in following two contexts: a) in the computation of bounds on the interpolation error within a triangle, and b) in the movement of vertices for mesh quality improvement. In both the contexts, we employ a line search technique in order to compute an optimal solution.

3.4 Active Set Method

The active set method [14] is used on solve constrained optimization problems. In our computation of the error bounds over a triangle, the optimal value may lie on the boundary of the triangle. When such an instance is found, the active set method is used to solve the optimization problem. We have described the optimization algorithm in Section 6 for both the error bound computation and mesh quality improvement.

4 Interpolation Error Bound

In this section, we derive the bounds on the interpolation error for second-order triangular elements for both the function and its derivative. The proofs for the error bounds closely follow the techniques employed by Johnson [6]. We denote a function by $v(x)$ and its quadratic approximation over a triangle by $\pi v(x)$. In order to establish an error bound on some approximation of a function, we assume that the third derivative of the function is bounded by some constant k. The bound k is necessary because the bound on $\|v(x) - \pi v(x)\|$ or $\|\nabla v(x) - \nabla \pi v(x)\|$ would behave arbitrarily if $v(x)$ behaves arbitrarily.

In [6], barycentric coordinates were used to establish the interpolation error bound on a linear triangular element. In order to derive the bounds for high-order triangles, we use the shape functions associated with Lagrangian polynomial interpolation used in FEM. In our analysis, we denote the barycentric coordinates as $\omega_i(x)$ for $i \in \{1, 2, 3\}$ and shape function parameters as $\lambda_i(x)$ for $i \in \{1, ..., 6\}$ for straight-sided quadratic triangular elements. Each of the vertices of the quadratic triangle is denoted by a_i, $1 \leq i \leq 6$. In the derivation below, x and y are 2D vectors, (x_1, x_2) and (y_1, y_2), representing points on a 2D plane. Also, e_1 and e_2 are orthogonal unit vectors along the coordinate axes.

In general, a quadratic approximation of a function $v(x)$ over a triangle K has the representation

$$\pi v(x) = \sum_{i=1}^{6} v(a_i)\lambda_i(x), \tag{1}$$

where $\pi v(x)$ is the quadratic approximation and $x \in K$. The Taylor series expansion at $x \in K$ is given by $v(y) = v(x) + L(x, y) + Q(x, y) + C(x, y)$, where

$$L(x, y) = \sum_{j=1}^{2} \frac{\partial v}{\partial e_j}(y_j - x_j),$$

$$Q(x, y) = \frac{1}{2} \sum_{i,j=1}^{2} \frac{\partial^2 v}{\partial e_i \partial e_j}(x)(y_i - x_i)(y_j - x_j),$$

$$C(x, y) = \frac{1}{3!} \sum_{i,j=1}^{2} \frac{\partial^3 v}{\partial e_i \partial e_j^2}(\zeta)(y_i - x_i)(y_j - x_j)^2,$$

and ζ is a point on the line segment between x and y. By choosing $y = a_i$, we have $v(a_i) = v(x) + L(x, a_i) + Q(x, a_i) + C(x, a_i)$. By substituting for $v(a_i)$ in the quadratic representation (Eq. (1)), we obtain

$$\pi v(x) = \sum_{i=1}^{6} \lambda_i(x) \left(v(x) + L(x, a_i) + Q(x, a_i) + C(x, a_i) \right). \tag{2}$$

As it has been shown for linear interpolation over a triangle [6], we will show that the following lemma holds true for shape functions associated with quadratic interpolation of a solution over a triangle.

$$\sum_{i=1}^{6} \lambda_i(x) = 1 \tag{3}$$

$$\sum_{i=1}^{6} \lambda_i(x) L(x, a_i) = 0 \tag{4}$$

$$\sum_{i=1}^{6} \lambda_i(x) Q(x, a_i) = 0 \tag{5}$$

By substituting the equations above into Eq. (2) and then rearranging the terms, we obtain

$$\pi v(x) - v(x) = \sum_{i=1}^{6} \lambda_i(x) C(x, a_i).$$

Since the third derivative is bounded by k, we obtain the following inequality:

$$|\pi x(v) - v(x)| \leq \frac{k}{3!} \sum_{i=1}^{6} |\lambda_i(x)| ||x - a_i||_2^3. \tag{6}$$

In order to prove the lemma stated above, we construct a new function $v^*(x)$ whose value, gradient, and the Hessian at a given x^* is identical to that of $v(x)$. Since the left-hand side of the equalities stated in the lemma are only a function of the coordinates of x^*, replacing $v(x)$ with $v^*(x)$ will evaluate the left-hand side to the same value. Since a function v^* can be constructed for every point x^* in the triangle, we claim that the left-hand side of the lemma evaluates to the same value for all points in the triangle.

Consider a constant function $v^*(x) = c = v(x^*)$. The quadratic model is given by $\pi v^*(x) = \sum_{i=1}^{6} \lambda_i(x) v^*(a_i)$. Since $v^*(x)$ is a constant function, $\pi v^*(x) = v^*(x)$. Thus, $\sum_{i=1}^{6} \lambda_i(x) = 1$, which prove Eq. (3). Consider a linear function $v^*(x)$ whose gradient is identical to gradient of $v(x)$ at some x^*. Again $\pi v^*(x) = v^*(x) = \sum_{i=1}^{6} \lambda_i(x) v^*(a_i)$. Also, since the second and the third derivative vanish at all points for a linear function, $Q(x, a_i) = 0$ and $C(x, a_i) = 0$. By substituting for Q and C in Eq. (2), we obtain $\pi v^*(x) = \sum_{i=1}^{6} (\lambda_i(x) v^*(a_i) + \lambda_i(x) L(x, a_i))$. Since $\pi v^*(x) = v^*(x)$, $\sum_{i=1}^{6} \lambda_i(x) L(x, a_i) = 0$, which proves Eq. (4). Now consider a quadratic function $v^*(x)$ whose value, gradient, and the Hessian are identical to $v(x)$ at x^*. As in the previous case, $\pi v^*(x) = v^*(x)$, and this function can be uniquely represented by the value of $v^*(x)$ at a_i, $1 \leq i \leq 6$. Thus, $\sum_{i=1}^{6} \lambda_i(x) Q(x, a_i) = 0$, which proves Eq. (5).

For linear triangle, the global maximum for interpolation error bounds is present at the center of the mincontainment circle[2]. Unfortunately, we were unable to find such an expression that would provide the local maxima for quadratic triangles. Thus, we resort to numerical optimization techniques to compute the location of the local maxima.

We also derive the error bounds for the gradient of a finite element solution in a similar way. By differentiating Eq. (1) and using the Taylor series expansion at x, we obtain the gradient of the quadratic model function as shown here:

[2] Mincontainment circle is the circumcircle for an acute triangle; for an obtuse triangle, it is the circle with the longest side as the diameter.

$$\frac{\partial \pi v(x)}{\partial e_j} = \sum_{i=1}^{6} \frac{\partial \lambda_i(x)}{\partial e_j} \left(v(x) + L(x, a_i) + Q(x, a_i) + C(x, a_i) \right). \tag{7}$$

As described above, it is can shown by constructing functions $v^*(x)$ that the following lemma holds:

$$\sum_{i=1}^{6} \frac{\partial}{\partial e_j} \lambda_i(x) = \frac{\partial}{\partial e_j} \sum \lambda_i(x) = 0, \tag{8}$$

$$\sum_{i=1}^{6} \frac{\partial}{\partial e_j} \lambda_i(x) L(x, a_i) = \frac{\partial v}{\partial e_j}, \tag{9}$$

$$\sum_{i=1}^{6} \frac{\partial}{\partial e_j} \lambda_i(x) Q(x, a_i) = 0, \tag{10}$$

for $j = \{1, 2\}$. By substituting this into Eq. (7), we obtain

$$\frac{\partial \pi v(x)}{\partial e_j} - \frac{\partial v(x)}{\partial e_j} = \sum_{i=1}^{6} \frac{\partial \lambda_i(x)}{\partial e_j} C(x, a_i),$$

and the inequality

$$\left| \frac{\partial \pi v(x)}{\partial e_j} - \frac{\partial v(x)}{\partial e_j} \right| \leq \frac{k}{3!} \sum_{i=1}^{6} \left| \frac{\partial \lambda_i(x)}{\partial e_j} \right| ||x - a_i||_2^3$$

is obtained by using the bound k on the third derivative of $v(x)$. This can be simplified to

$$\left| \frac{\partial \pi v(x)}{\partial e_j} - \frac{\partial v(x)}{\partial e_j} \right| \leq \frac{k}{3!} \sum_{i=1}^{6} |\nabla \lambda_i(x)|_2 \, ||x - a_i||_2^3,$$

for $j = \{1, 2\}$. Thus,

$$|\nabla (\pi v(x) - v(x))|_2 \leq \frac{\sqrt{2}k}{3!} \sum_{i=1}^{6} |\nabla \lambda_i(x)|_2 \, ||x - a_i||_2^3. \tag{11}$$

5 Characteristics of the Interpolation Error Bounds

In this section, we describe some of the characteristics of the bounds we obtained for interpolation error of a function and its gradient. We use these characteristics to design an algorithm to compute the bounds for a triangle and to develop an algorithm to improve the bounds.

5.1 Interpolation of the Function

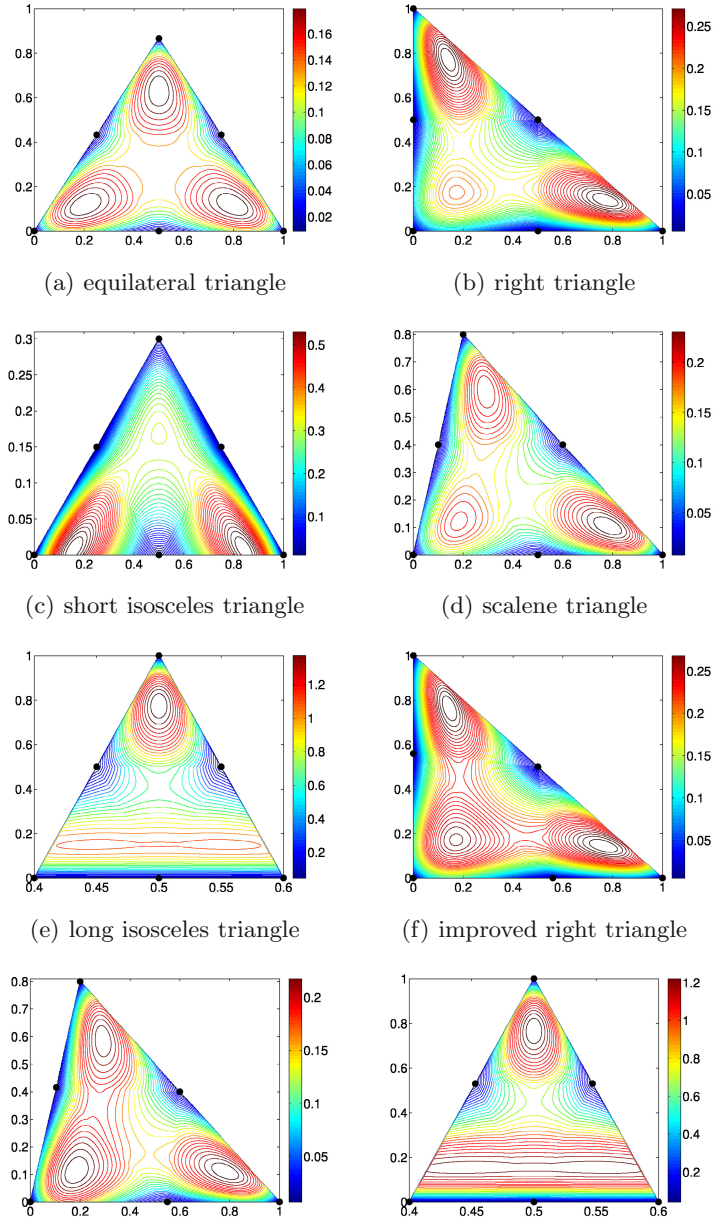
In order to gain an understanding of the error bounds provided in Eq. (6), we plot the error bounds for various triangles as shown in Fig. 2. The figure provides the plots for the error bounds within a triangle. The black dots in the figure denote the vertices of the quadratic triangles. Notice that the error bounds are the low near the vertices, and they grow as we move away from them. Also notice that the bounds are discontinuous at points where the shape function vanish, *i.e.*, $\lambda_i(x) = 0$, due to the presence of the absolute function, $|\lambda_i(x)|$, in the equation. For a straight-sided quadratic triangle, the bounds are discontinuous along the lines joining the edge nodes. There are three-four local maxima in each plot, and three higher maxima are present near the three vertices.

In Fig. 2(a)-(e), the edge nodes are chosen to be the mid points of the respective edges. For an equilateral triangle, the error bound are symmetric with respect to all its vertices, but for other triangles, the global maximum is found near the vertex with the smallest angle. In Fig 2(f)-(h), we have moved the edge nodes from the midpoints towards the smaller angles on the respective sides of the triangle. Intuitively, we move the edge nodes towards the higher maximum in order to reduce the error in the neighborhood. In all the cases, the higher maximum is present near the smaller angle on the side of a triangle. We were able to improve the error bounds only for the right triangle, scalene triangle, and long isosceles triangle by this operation. The equilateral triangle and short isosceles triangle did not respond positively to the movement of the vertices on the edge, *i.e.*, the bounds could not be improved by vertex movement because the local optimum had already been reached. For the right triangle, scalene triangle, and long isosceles triangle, we observed an improvement of 3%, 7%, and 11%, respectively.

5.2 Interpolation of the Gradient of the Function

As in the section above, we visualize the error bounds for the magnitude of the gradient of the interpolated function by plotting the contours of Eq. (11). For linear triangles, the corresponding bounds yield a result that show that the bounds are locally maximum at the vertices of the triangle[3]. For quadratic triangles, too, the bounds are locally maximum at the three vertices of the triangle. The plots are show in Fig. 3. Since the error was observed to be maximum at the vertices, computation of the bound does not require the use of numerical optimization algorithms. A formal proof for this claim is not available at this point. We have not shown the plot for equilateral triangle because they are similar to these plots and also very symmetric. A closer

[3] The error bounds are locally maximum because they are constrained to be inside the triangle. By locally maximum, we mean that the Kahn-Karush-Tucker (KKT) conditions are satisfied.



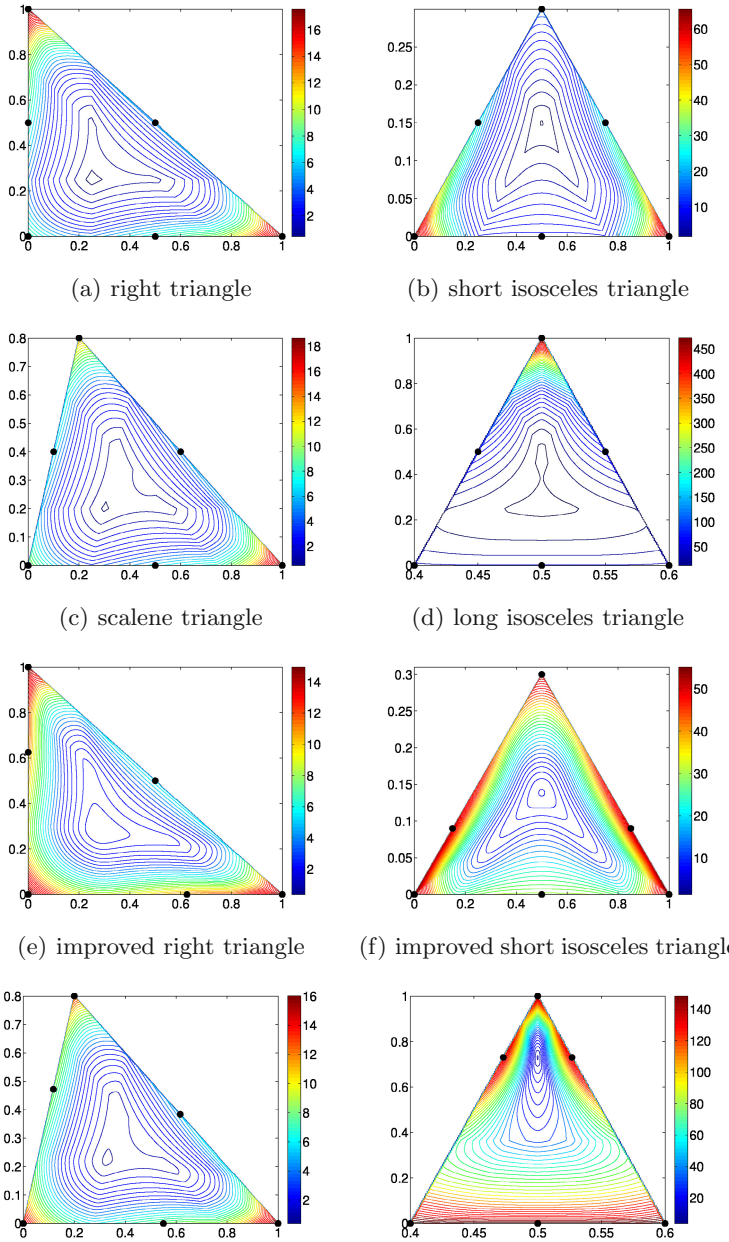
(a) equilateral triangle (b) right triangle

(c) short isosceles triangle (d) scalene triangle

(e) long isosceles triangle (f) improved right triangle

(g) improved scalene triangle (h) improved long isosceles triangle

Fig. 2 A contour plot of the error bounds of the interpolant (as given in Eq. (6)) for various triangles. Note that the scaling of the axes and the colors are different for every plot. We moved the edge nodes for only those triangles for which improvement was observed. This improvement comes at a cost of the increasing the bounds on the error in other parts of the triangles.



(a) right triangle (b) short isosceles triangle

(c) scalene triangle (d) long isosceles triangle

(e) improved right triangle (f) improved short isosceles triangle

(g) improved scalene triangle (h) improved long isosceles triangle

Fig. 3 Contour plots of the error bounds of the gradient of the interpolant (as given in Eq. (11)) for various triangles after being normalized with respect to their area. Note that the scaling of the axes and the colors are different for every plot. The plots are smooth, but appear nonsmooth due to insufficient sampling and rendering.

observation of the plots reveal that greater improvements in the bounds can be achieved by moving the edge nodes than in the previous case. For the right triangle, scalene triangle, short isosceles triangle, and long isosceles triangle, we observed an improvement of 14.5%, 15.9%, 20.9%, and 67.7%, respectively.

5.3 Saddle Point in the Movement of the Edge Vertices

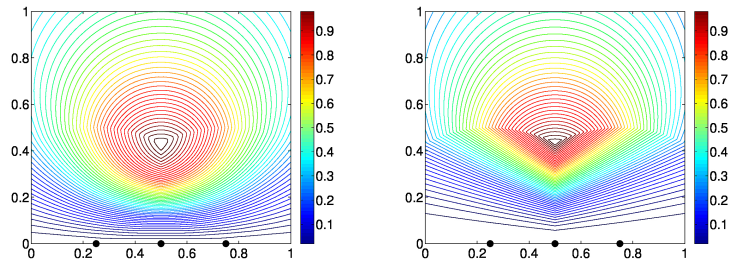
Another characteristic of the error bound is the existence of saddle points with respect to the location of the edge nodes. For instance, a typical quadratic right triangle contains the edge nodes at the mid point of the respective edges. We observe that moving both the edge nodes on the sides forming the right angle improves the error bound of the interpolant and the gradient. But moving only one of the edge nodes results in the worsening of the maximum error bounds. Analytical or finite difference computation of the gradient of the error bound with respect to the location of the edge node will not provide the desired descent direction. Heuristic techniques have to be designed to determine the direction and the magnitude of the movement of edge nodes for the optimization of the error bounds.

5.4 Scale Invariance

Eqns. (6) and (11) are not normalized for the area of the triangle. In order to convert them to a shape-based metric, the equations can be divided by $(\mathrm{Ar})^{\frac{3}{2}}$, where Ar is the area of the triangle. The inverse of the above metric is usually plotted so that the quality is maximum for an equilateral triangle and the quality is 0 for a degenerate triangle. The contour plots for the scale invariant quality metrics associated with the interpolation error is shown in Fig. 4. Two of the three corner vertices of the triangle are fixed at (0.25, 0.00) and (0.75, 0.00), and the third corner vertex is free to move in the plane. The edge nodes are fixed to be the mid points of the corresponding edges. The interpolation error bound contours are plotted as a function of the position of the third corner vertex. The contours are very similar to corresponding bounds for linear elements as shown in [5], but the relative magnitudes of the bounds are different because they have been derived from different formulations.

6 Implementation

In this section, we discuss the implementation of the algorithm to compute the bound on the interpolation error for a quadratic triangle and an algorithm to improve the bounds. Both the computation of the bounds and its improvement are numerical optimization problems. While both are constrained



(a) function interpolation error (b) gradient interpolation error

Fig. 4 Contour plots of the normalized scale-invariant quality metrics associated with the interpolation error bound and the gradient interpolation error bounds. The contours are very similar to the corresponding bounds for linear elements as shown in [5].

optimization problems, the latter can be solved using unconstrained optimization algorithms. In the latter problem, the vertices are constrained to define a triangle of positive orientation, but the cost of the objective function associated with near-degenerate triangle is very high. This is because either the quality of the triangle is normalized to take the area of the triangle into account or a near-degenerate triangle leads to bigger triangle in the neighborhood whose bounds are very large. Thus, the problem behaves like an unconstrained optimization problem.

6.1 The Active Set Method for Interpolation Error Bound Computation

From the contour plots in Fig. 2, we know that we need to compute three maxima that may be located inside the triangle or on its boundary. The active set method requires a starting iterate, and by positioning the initial iterate at strategic locations, we may be able to compute all three maxima.

In order to compute the initial iterates, we join the edge nodes so that the triangle is divided into four parts. The centroids of the three of the four triangles that contain the corner vertices are chosen as initial iterates. The active set method described in Algorithm 1 is used to find the maxima near each of the three iterates. Note that the gradients are continuous within each of four parts of the triangle. Thus, the gradient computation can be done analytically. The maximum of the three bounds returned by the active set method is chosen as the quality for a triangle.

Algorithm 1. The active set method for computing the maximum bound of the interpolation error.

```
start the iteration with point x_0 and i = 0.
while the KKT conditions are not satisfied do
  if x_i is on the border then
    compute the gradient
    if the gradient points outside the domain then
      maximize the function along the border
    else
      carry out a line search in the usual manner as described below
    end if
  else
    compute the gradient and decent direction
    carry out a line search along the decent direction
    if the line search takes x_i outside the triangle then
      snap x_i back to the edge
    end if
  end if
  update x_i and increment i
end while
```

6.2 The Nonlinear Conjugate Gradient Method for Mesh Quality Improvement

The nonlinear conjugate gradient method for mesh quality improvement is implemented as it is implemented in Mesquite [15]. We implemented a Polak-Ribière variant of the algorithm, and the gradient computation is carried out using finite differences. The p-norm of the qualities for all triangles was chosen as the objective function because the worst element is more likely to be optimized for a high p. A global optimization technique is used in which the gradient (and the descent direction) of the objective function is computed for all the corner vertices, and the vertices are moved simultaneously. The magnitude of the gradients dictate the relative distances by which the vertices move in an iteration.

For an edge node, however, a direction was chosen by observing how the lowest of the three maxima changes as a result of the movement of the edge node along an edge[4]. Note that a typical edge node affects the quality of two triangles. Thus, it is necessary to examine the effect of its movement on both triangles and then choose a direction to move. We chose the direction that increases the p-norm of the lowest maxima of the two triangles. We hope that that the increase in the lowest maxima translates to a decrease in the highest maxima. The magnitude of the increase in the p-norm helps us determine the relative distances by which all the edge nodes should be moved for effective

[4] Recall that the error bound function has a saddle point with respect to the location of an edge node if the node at the mid point of the edge.

quality improvement. This heuristic technique enables global optimization of edge nodes through a line-search technique. Note that the actual element quality, the highest maxima, is optimized in the line search.

6.3 Possible Acceleration Techniques

We have to carry out numerical optimization iterations to compute the quality associated with the interpolation error. This may be expensive if implemented naively. Techniques such as memorization of the location of the maximum bound may be used, but they are likely to improve the time only by a limited factor. A practical technique is to store a precomputed table for various location of the triangle vertices, and to refer to the table for quality evaluation. Simple interpolation techniques may be used to interpolate for vertex locations that are not present in the table.

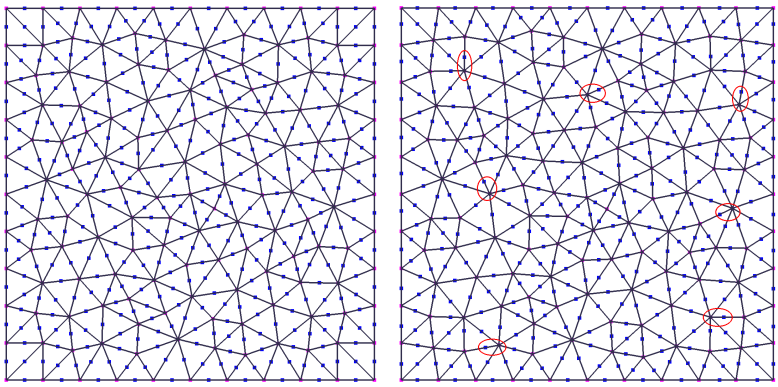
For quality metric associated with the error in the *gradient* of the function, such acceleration techniques are useful, but not necessary as we have observed that the maximum bounds are found at the vertices of the triangle. Thus, a constant number of computations are necessary to compute the quality.

7 Experiments

In order to assess the impact of our mesh quality improvement algorithm, we carry out numerical experiments and examine the improvement of the bounds. The purpose of this section is to demonstrate the possible mesh quality improvement through the use of the an appropriate quality metric. We find that small meshes were sufficient for this purpose. The experiments can be carried out for large meshes as well, but we believe no additional insights can be gathered from them.

The algorithm is implemented in C++ as described in the previous section. Gmsh [16] was used to generate a small mesh with 252 element and 545 vertices. The in-built mesh quality optimization routine in Gmsh was used to improve the mesh before its quality was improved by our algorithm. Gmsh generates a mesh with edge nodes at the mid point of the respective edges. We improve the mesh by moving both the corner vertices and edge nodes.

In our first experiment, we seek to optimize the error bound for the interpolated function. Since we carry out a numerical optimization routine just to compute a quality of an element in this experiment, the implementation is not yet optimal. A practical implementation, however, may use some of the acceleration techniques described at the end of the previous section. In our experiment, the maximum error bound in the initial mesh was about 293 units. By moving just the corner vertices, the maximum bound was improved to 268 units (8.5% improvement). Our algorithm was able to improve it to about 263 units (10.0% improvement) by moving both the edge nodes and corner vertices. The root mean square (RMS) bound was marginally improved from 172



(a) initial mesh (b) optimized mesh

Fig. 5 The initial and optimized quadratic mesh. The initial mesh was generated using Gmsh [16], and the mesh was optimized to minimize the bound on the error gradient of an interpolated function. Notice the movement of the edge nodes from the mid points of the respective edges, especially for triangles with very small or large angles. Encircled regions show a large movement of the edge nodes from the mid point of their respective edges.

to 171 units (0.5% improvement) in both the cases. Note that the bounds were not normalized for the areas of the triangles and also that the mesh quality was improved in Gmsh before it was improved here.

In our second experiment, we seek to optimize the error bound in the gradient of the interpolated function. The maximum bound in the initial mesh was 721 units, and the RMS bound was about 456 units. First, by moving only the corner vertices, we were able to improve the maximum bound to 658 (8.7% improvement), and the RMS bound to 426 units (6.5% improvement). Next, we moved the edge nodes and corner vertices, but only after the corner vertices had been optimized. In this case, the maximum bound was improved to 557 units (22.7% improvement), and the RMS bound was improved to 380 units (16.7% improvement). Finally, we moved both the edge nodes and corner vertices in all the iterations. The maximum bound improved to 539 units (25.2% improvement), and the RMS bound improved to 378 units (17.1% improvement). The initial and the final meshes for the last experiment are shown in Fig. 5.

From the above experiments, we infer that using the r-refinement technique to improve the bounds for the gradient of the interpolated function is more effective than using the technique to improve the bounds for the function interpolation. The first experiment is prohibitively expensive without any acceleration technique. It should be used sparingly and only when necessary. A quality metric used for linear elements may suffice for quadratic elements. One possible application of using the error bound is the placement of vertices on a surface mesh to improve the geometric fidelity. As the number of surface

elements are low compared to the number of elements in a volume mesh, the error bound quality metric would work well for this purpose. The second experiment takes about as much time as an implementation in which any other linear element quality metric used to optimize the mesh. This is because we compute the bound on the gradient only at the three corner vertices. This was possible because we observed that the location of maximum bound is always at the corner vertices. Note that a formal proof for this claim is not available at this point.

8 Conclusions and Future Work

In this paper, we adapted a framework to bound the errors observed in function interpolation and its derivatives for a quadratic triangle. The framework can be extended to quadratic tetrahedra or other high-order elements used in FEM. Visualization of the bounds has revealed some of its characteristics that can enable high-quality mesh adaptation for numerical solvers. The study of the characteristics has also helped us develop a numerical technique to optimize the error bounds, and we have shown the improvement in a mesh through its use.

This work can be easily extended to include other quadratic elements such as quadrilaterals, tetrahedra, and hexahedra and also to include cubic or higher-order triangles. We have only considered straight-sided triangles in this paper. Curvilinear triangles are also present in contemporary high-order meshes, and the behavior of the error bounds over such elements can shed light on appropriate ways to define a quality metric. Such definitions [17] may also help in untangling high-order meshes efficiently. We assumed that the third order derivatives were isotropic in our study. We also plan to study anisotropy and its effect on vertex placement.

It would be interesting to compare some of the vertex placement techniques mentioned in Section 2 for high order elements [10, 11, 12] with the technique described in this paper.

Geometric fidelity is also an important factor in numerical simulations. Our interpolation bounds could be used to place vertices on the surface meshes to best approximate the underlying geometry. Its effect on numerical simulation can also be studied by carefully-constructed experiments.

Other important factors in determining the mesh quality include the conditioning of the stiffness matrix constructed from the mesh and discretization errors. These facets of high-order finite element quality metrics should be studied in detail as it has been studied for linear elements [5].

Acknowledgment. The work of the first author was supported in part by the NIH/NIGMS Center for Integrative Biomedical Computing grant 2P41 RR0112553-12 and DOE NET DE-EE0004449 grant. The work of the second author was supported in part by the DOE NET DE-EE0004449 grant and ARO W911NF1210375 (Program Manager: Dr. Mike Coyle) grant.

References

1. Babuška, I., Aziz, A.K.: On the angle condition in the finite element method. SIAM J. Numer. Anal. 13(2), 214–226 (1976)
2. Knupp, P.: Algebraic mesh quality metrics. SIAM J. Sci. Comput. 23(1), 193–218 (2001)
3. Munson, T.: Mesh shape-quality optimization using the inverse mean-ratio metric. Math. Program. 110(3), 561–590 (2007)
4. Baker, T.J.: Element quality in tetrahedral meshes. In: Proc. of the 7th International Conference on Finite Element Methods in Flow Problems, pp. 1018–1024 (1989)
5. Shewchuk, J.R.: What is a good linear element? Interpolation, conditioning, and quality measures (2002) (unpublished)
6. Johnson, C.: Numerical Solution of Partial Differential Equations by the Finite Element Method. Cambridge University Press, Cambridge (1987)
7. Lu, Q., Shephard, M.S., Tendulkar, S., Beall, M.W.: Parallel curved mesh adaptation for large scale high-order finite element simulations. In: Jiao, X., Weill, J.-C. (eds.) Proceedings of the 21st International Meshing Roundtable, vol. 123, pp. 419–436. Springer, Heidelberg (2013)
8. Lu, Q.: Development of parallel curved meshing for high-order nite element simulations. Master's thesis, Rensselaer Polytechnic Institute, Troy, NY, USA (2011)
9. Lowrie, W., Lukin, V.S., Shumlak, U.: A priori mesh quality metric error analysis applied to a high-order finite element method. J. Comput. Phys. 230(14), 5564–5586 (2011)
10. Hesthaven, J.S.: From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex. SIAM J. Numer. Anal. 35, 655–676 (1998)
11. Taylor, M.A., Wingate, B.A., Vincent, R.E.: An algorithm for computing fekete points in the triangle. SIAM J. Numer. Anal. 38, 1707–1720 (2000)
12. Chen, Q., Babuska, I.: Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle. Comput. Methods Appl. Mech. Engrg. 128(3-4), 405–417 (1995)
13. Bos, L.: Bounding the lebesgue function for Lagrange interpolation in a simplex. J. Approx. Theory 38, 43–59 (1983)
14. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (2006)
15. Brewer, M.L., Diachin, L.F., Knupp, P.M., Leurent, T., Melander, D.J.: The mesquite mesh quality improvement toolkit. In: The 12th International Meshing Roundtable (2003)
16. Geuzaine, C., Remacle, J.F.: Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. Inter. J. Numer. Meth. Eng. (2009)
17. Sastry, S.P., Shontz, S.M., Vavasis, S.A.: A log-barrier method for mesh quality improvement and untangling. Eng. Comput., 1–15 (2012)