Available online at www.sciencedirect.com
SCIENCE DIRECT®
Finite Elements in Analysis and Design 41 (2005) 686–702
FINITE ELEMENTS IN ANALYSIS AND DESIGN
www.elsevier.com/locate/finel

Adaptive computations on conforming quadtree meshes

A. Tabarraei, N. Sukumar*

Department of Civil and Environmental Engineering, University of California, One Shields Avenue, Davis, CA 95616, USA

Received 21 August 2004; accepted 28 August 2004
Available online 24 January 2005

Abstract

In this paper, the quadtree data structure and conforming polygonal interpolants are used to develop an *h*-adaptive finite element method. Quadtree is a hierarchical data structure that is computationally attractive for adaptive numerical simulations. Mesh generation and adaptive refinement of quadtree meshes is straight-forward. However, finite elements are non-conforming on quadtree meshes due to level-mismatches between adjacent elements, which results in the presence of so-called *hanging nodes*. In this study, we use meshfree (natural-neighbor, *nn*) basis functions on a reference element combined with an affine map to construct conforming approximations on quadtree meshes. Numerical examples are presented to demonstrate the accuracy and performance of the proposed *h*-adaptive finite element method.

Keywords: Barycentric coordinates; Meshfree methods; Natural neighbors; Laplace interpolant; Quadtree data structure; Hanging nodes

1. Introduction

Uniform meshes are not a computationally viable choice for solving system of partial differential equations in which steep gradients, singularities, or discontinuities need to be captured. Using large elements will lead to non-satisfactory results in the vicinity of such regions and the use of small elements in the whole domain is not economical. The purpose of adaptive refinement strategy is to automatically adjust mesh resolution in order to obtain better accuracy where it is needed most. Today, the most advanced refinement techniques are based on natural refinement of elements. Quadtree data structure,

* Corresponding author. Tel.: +1 530 7546415; fax: +1 530 7527872.
E-mail address: nsukumar@ucdavis.edu (N. Sukumar).

0168-874X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.finel.2004.08.002

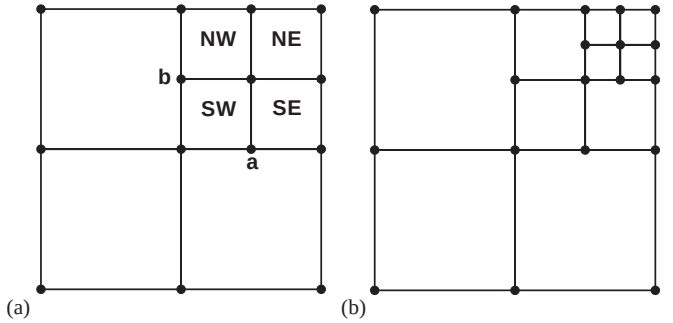

Fig. 1. Quadtree data structure. (a) Level 1 and (b) Level 2.

which is based on the principle of recursive decomposition [1] provides a simple, fast, and efficient way for *h*- and *p*-refinement. As a spatial data structure, efficient storage and fast data retrieval in a quadtree (or octree in 3D) are unmatched, which renders it particularly attractive for adaptive simulations. The generation of *hanging nodes*, however, has been a significant impediment in using quadtree meshes in finite element methods. Hanging nodes are generated after each refinement, if the adjacent elements are not of the same size. In Fig. 1, quadtree meshes after one and two levels of refinements are shown. The *hanging nodes a* and *b* that are generated in neighboring elements at different levels of refinement are indicated. Since classical finite element methods are non-conforming on quadtree meshes, complicated algorithms and interpolation functions are needed to handle the presence of hanging nodes. Therefore for better efficiency, the number of hanging nodes over an element face is minimized. A common method that is widely used to minimize the number of hanging nodes is the *restricted quadtree* [2–4]. In a restricted quadtree, the maximum difference between the level of refinement of adjacent elements cannot be more than one (this is also known as the 2:1 rule).

There are three classical methods, which are often used to handle the presence of hanging nodes on quadtree meshes. The first approach is to add temporary triangular or rectangular elements to those elements that have a hanging node to obtain compatible meshes [4,5]. On using the 2:1 rule, the variation of neighbor arrangements is limited to five cases and therefore a compatible mesh can be easily obtained as shown in Fig. 2. Finding the elements that have hanging nodes and adding temporary triangles to them renders the problem to be expensive, and results in meshes that are locally over-refined and are far from optimal. The second method is to constrain the hanging nodes to corner nodes. Transformation of classical shape functions to constrained shape functions is required by this method. In the third approach, Lagrange multipliers or penalty methods are used to treat the incompatibility, which leads to algorithmic complexities. Recently, conforming approximations over quadtree meshes have been obtained using hierarchical enrichment [6,7] as well as B-splines [8]. These methods also rely on the 2:1 rule.

The key contribution in this study is the adaptation of a recently proposed polygonal interpolant [9] to enable one to conduct numerical computations on quadtree decompositions of any finite element mesh. We use a natural neighbor (Laplace) interpolant [10–12] to obtain $C^0(\Omega)$ admissible approximations along edges that include *hanging nodes*. The shape functions are defined over the reference elements and through an affine map, conforming basis functions are constructed on quadtree meshes. These basis functions satisfy all the desirable properties of finite element basis functions, and therefore they can be easily incorporated into existing FE codes.

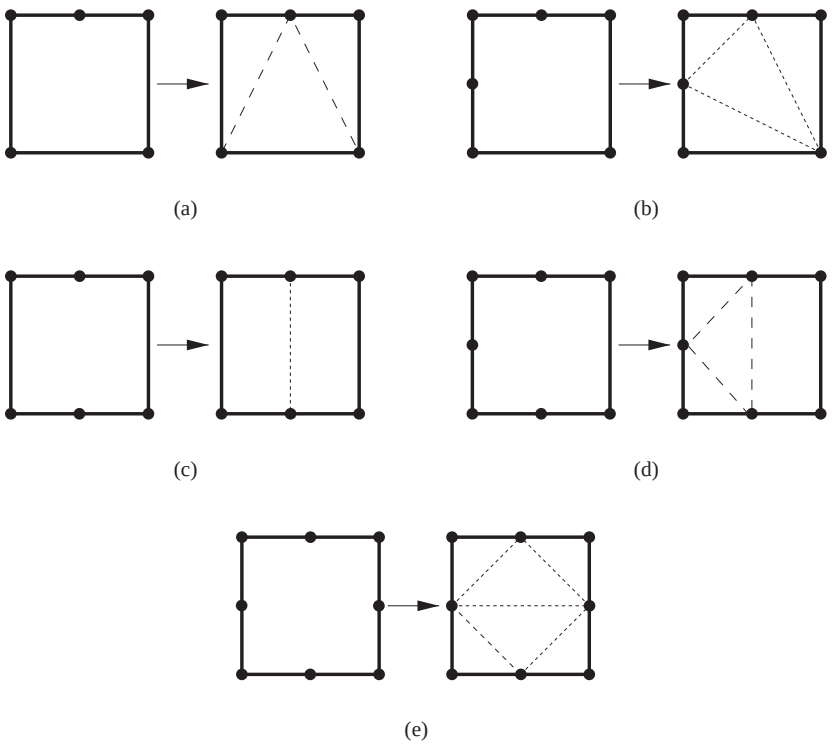

Fig. 2. Adding temporary elements to make a compatible mesh.

The outline of this paper follows. The construction of the Laplace interpolant on polygonal meshes is described in Section 2. In Section 3, the quadtree data structure is reviewed. The application of a recently developed polygonal interpolant [9] to solve the problem of hanging nodes is presented in Section 4. The adaptivity strategy is described in Section 5, and the merits of the new technique for *h*-adaptive numerical computations is demonstrated in Section 6. We close with some final remarks in Section 7.

2. Conforming interpolants on polygons

Barycentric coordinates on simplex elements (triangles in 2D and tetrahedra in 3D) are well-known. The generalization of barycentric coordinates to *n*-sided polygons is a topic of current research. In 1975, Wachspress [13] proposed a rational basis function for convex polygons, which has recently generated renewed interest. Meyer and co-workers [14] derived a simplified expression for Wachspress basis functions, whereas Floater [15] proposed barycentric coordinates in which a vertex in a planar triangulation is expressed as a convex combination of its neighboring elements. Recently, a new polygonal interpolant based on *natural neighbors* [9] was proposed [16]. This technique will be reviewed here and later we will use it to construct conforming approximations on quadtree meshes.

Consider a domain Ω that is discretized by N nodes. The Voronoi cell of a point p that lies within the convex hull is the locus of all points that are closer to that point than to any other node in the plane [17].

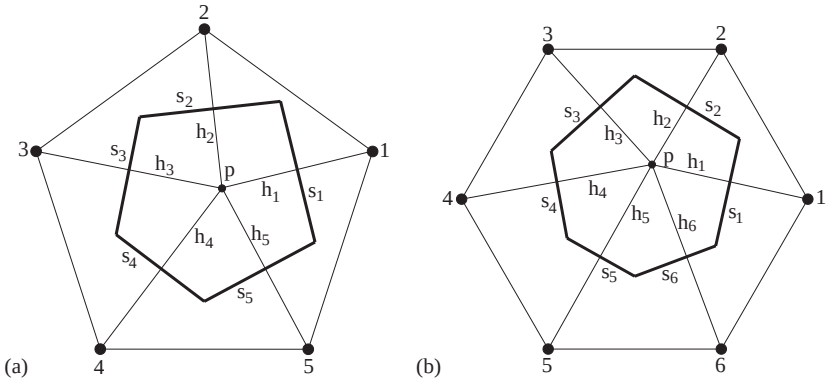

Fig. 3. Voronoi cell of point p and construction of Laplace shape function. (a) Pentagonal reference element and (b) hexagonal reference element.

In Fig. 3, the Voronoi cell of point p is shown. The Delaunay triangulation can be constructed by connecting nodes that have a common Voronoi edge. If a point p lies inside a Delaunay triangulation t, those nodes that define triangle t are natural neighbors of point p [16]. The Laplace interpolant of point p is defined as

$$\phi_i(\mathbf{x}) = \frac{\alpha_i(\mathbf{x})}{\sum_{j=1}^{n} \alpha_j(\mathbf{x})}, \quad \alpha_j(\mathbf{x}) = \frac{s_j(\mathbf{x})}{h_j(\mathbf{x})}, \quad \mathbf{x} \in \mathbf{R}^2, \tag{1}$$

where s_i is the common edge between Voronoi cell of point p and node i and h_i is the distance between point p and node i (Fig. 3). The Laplace interpolant satisfies the following properties [10]:

(1) Non-negative, interpolate, and form a partition of unity:

$$0 \leqslant \phi_i(\mathbf{x}) \leqslant 1, \quad \phi_i(\mathbf{x}_j) = \delta_{ij}, \quad \sum_{i=1}^{n} \phi_i(\mathbf{x}) = 1, \tag{2}$$

where δ_{ij} is the Kronecker-delta.

(2) Linear precision or completeness [18]:

$$\sum_{i=1}^{n} \phi_i(\boldsymbol{\xi})\mathbf{x}_i = \mathbf{x}, \tag{3}$$

which indicates that the shape functions can exactly reproduce a linear function. On the boundary of the domain Ω the interpolant is linear. This property in conjunction with the Kronecker-delta property in Eq. (2) ensures that linear essential boundary conditions can be imposed as in finite elements. For further details on the Laplace interpolant and its use within Galerkin methods, the interested reader can refer to [19,20].

To construct conforming approximations over quadtree meshes, first we define the interpolant over reference elements. In Fig. 4, the reference elements for a pentagon, hexagon, heptagon, and octagon are shown. Note that all of the vertices (nodes) of the reference elements lie on the same circumcircle and hence all the nodes are natural neighbors for any point that lies within the circumcircle. For a triangle and

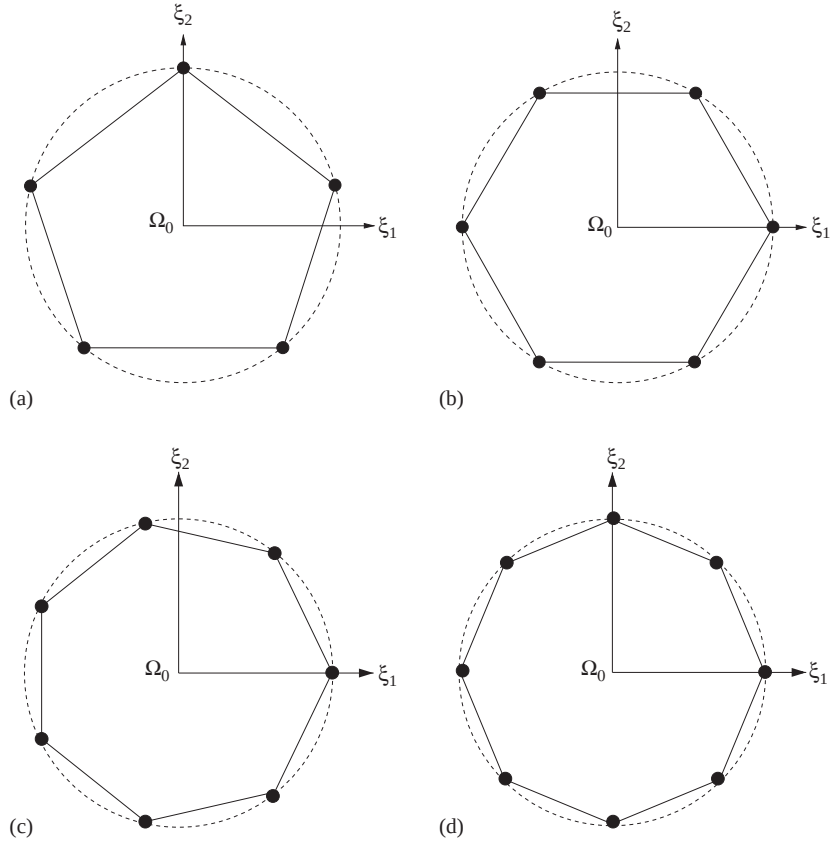

Fig. 4. Reference (canonical) elements. (a) Pentagon; (b) hexagon; (c) heptagon and (d) octagon.

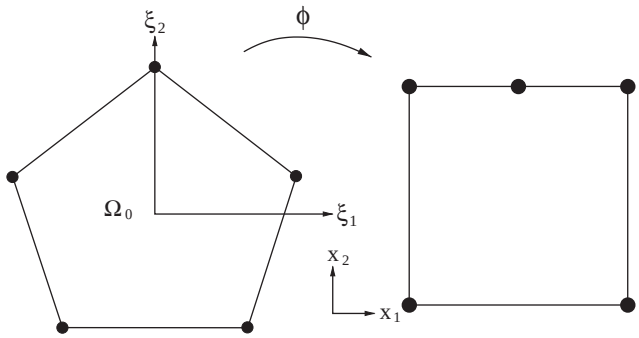

Fig. 5. Mapping from a regular pentagon to a quadtree element with one hanging node.

a bi-unit square with nodes on the circumcircle, Laplace and finite element interpolation are equivalent [19]. The interpolation function over the physical element is constructed by using the isoparametric map given in Eq. (3). In Fig. 5, the mapping from a regular pentagonal reference element to a quadtree element

with one hanging node is shown. Since the mapping is affine, the interpolant remains linear on the edges of the physical element.

3. Quadtree data structure

In this section, we briefly review the quadtree data structure and define some of the related terminology. For more information, the interested reader can see [1,21]. Quadtree is a hierarchical data structure, which was first used in computer graphics [22–24] and image processing [25]. The use of quadtree for mesh generation in finite element studies was introduced by Yerry and Shephard [2]. It has also been used in computational fluid dynamics [26,27], *h*- and *p*-adaptive schemes [6,8,28], solution of Euler equation [29], etc.

In a quadtree data structure, the domain is a unit square. This unit square, which is known as the *root* of the data structure can be decomposed into four new equal elements, and each of these new elements is split recursively until a stopping criterion is met. The new elements are called *children* of the decomposed (*father*) element. The children can be recognized by their relative position in the father element {SW, SE, NW, NE}. An element is called a *leaf* cell or a *terminal* cell if it does not have any children. Two elements are called *neighbor* or *adjacent* if they have a common edge. The *level* of an element is the number of decompositions needed to obtain that element. The level of the root is zero.

The information about a quadtree can be stored in a tree structure. In Fig. 6, a quadtree data structure and its representative tree is shown, with a pointer connecting each element to its father. On using the tree structure, information such as level of refinement, ancestors, neighbors, and children of an element can be easily traced. For each element, the connectivity data, refinement level, a pointer to its father, and a pointer to its children are stored. With this information, other required data such as the neighbors of an element can be found [21]. Different storage methods for quadtree can be found in [1,27].

After each refinement, if the new elements and their adjacent elements are in different levels, hanging nodes are generated. Hanging nodes are the vertices of the smaller element that lie on the edge of the larger element, but not on any of its vertices (e.g., nodes *a* and *b* in Fig. 1a). In Fig. 7a, a quadtree mesh with two hanging nodes is shown. Note that the classical shape function of node *a* along edge 1–2 of element *A* is parabolic, whereas it is piecewise-linear on edges 1-a-2 of elements *B* and *C*. Therefore,

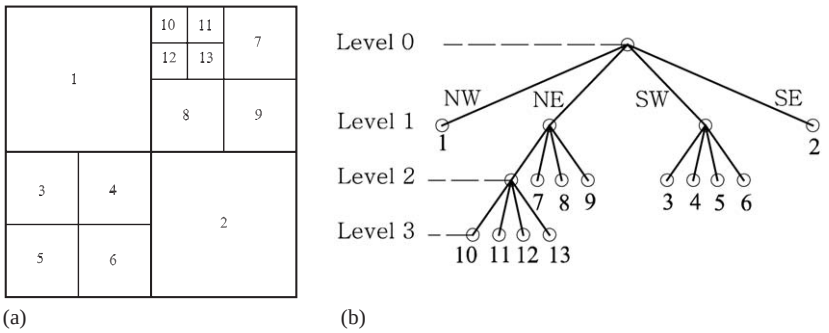

Fig. 6. Quadtree and its representative tree.

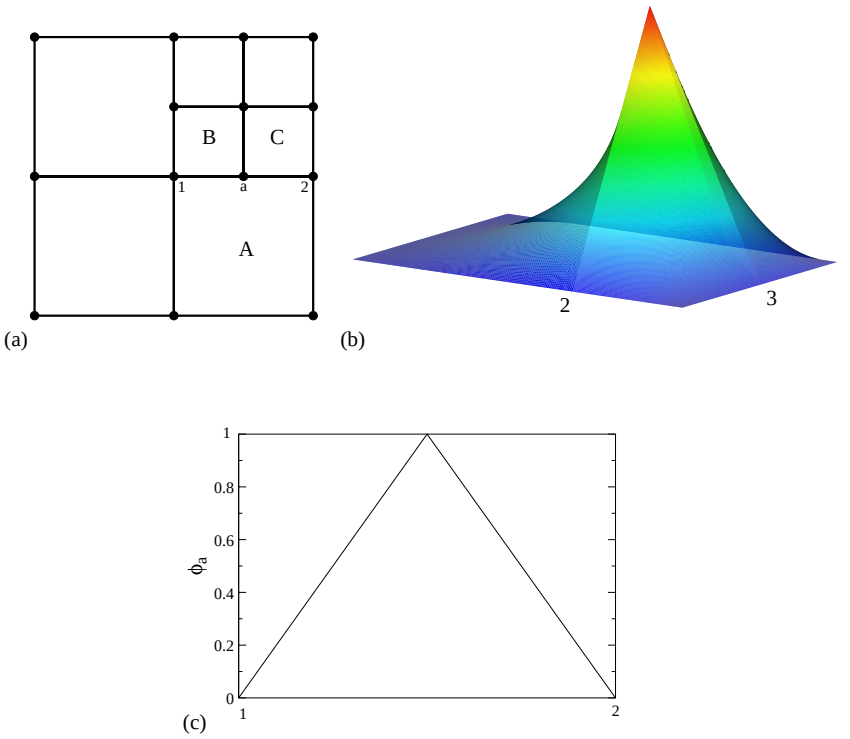

Fig. 7. A quadtree mesh and the shape function of a hanging node. (a) Quadtree mesh; (b) shape function of node *a* and (c) shape function of node *a* along edge 1–2.

finite elements are non-conforming along the edge 1–2, which include a hanging node. In order to satisfy the conformity of the primary variable between these elements, careful treatment is required. In this paper, we use the Laplace interpolants to construct conforming shape functions over a quadtree mesh. On using this technique, there is no need to decompose existing elements or to use special nodal shape functions.

4. Conforming shape functions on quadtree meshes

As described in the previous section, classical finite elements are non-conforming along edges that include a hanging node. In the present study, we use the polygonal interpolant introduced in Section 2 to construct conforming $C^0(\Omega)$ approximation on quadtree meshes. Consider element A in Fig. 7a, which has one hanging node along the edge 1–2. Referring to Fig. 5, the Laplace shape functions are defined on a regular pentagon and an affine (isoparametric) map is used to obtain the shape functions for element A. In Fig. 7b, the shape function plot for node *a* is shown. As one can observe, the shape function of node *a* is piecewise-linear along edges 1-*a* and 2-*a* (Fig. 7c). Note that this behavior on the boundary is distinct from higher-order FEM, as mentioned earlier in Section 3. For elements with more hanging nodes, the shape functions can be constructed by mapping from the corresponding polygonal reference element (hexagon, heptagon, octagon, etc.). On using Laplace interpolation, there is no distinction between hanging nodes and corner nodes and in fact all the nodes are considered as corner nodes and the shape function of all the

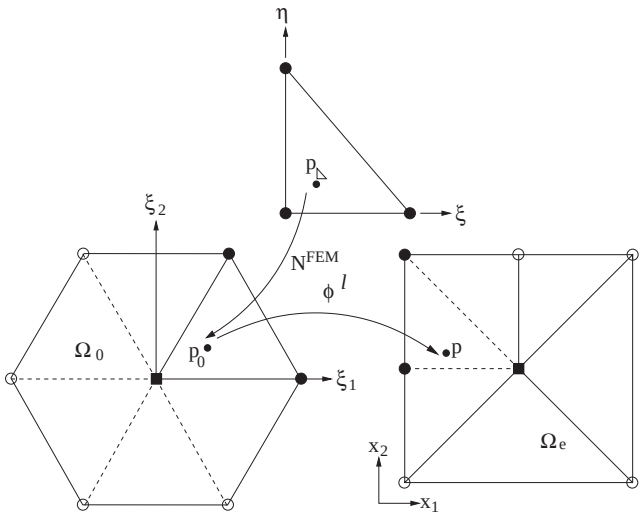

Fig. 8. Numerical integration scheme based on the partition of the reference element.

nodes are obtained similarly. Because of this, there is no need to restrict the number of hanging nodes and the 2:1 rule can be ignored. For the purpose of numerical integration, we first sub-divide the reference element into triangular elements and integrate the function over these triangles by using well-known quadrature rules. The integral of a scalar function f on the physical element can be obtained as

$$\int_{\Omega_e} f \,\mathrm{d}\Omega = \int_{\Omega_0} f|\mathbf{J}_2|\,\mathrm{d}\Omega = \sum_{j=1}^{n} \int_{\substack{\Delta_j \\ \Omega_0}} f|\mathbf{J}_2|\,\mathrm{d}\Omega = \sum_{j=1}^{n} \int_0^1 \int_0^{1-\xi} f|\mathbf{J}_1^j||\mathbf{J}_2|\,\mathrm{d}\xi\,\mathrm{d}\eta. \tag{4}$$

The sequence of mappings used in the integration scheme are shown in Fig. 8.

5. Adaptive strategy

The model problem which we consider is the following elliptic boundary-value problem:

$$-\nabla^2 u(\mathbf{x}) = f(\mathbf{x}), \quad \text{in } \Omega, \tag{5a}$$

subject to the boundary conditions:

$$\frac{\partial u}{\partial n} = g, \quad \text{on } \Gamma_N, \tag{5b}$$

$$u = 0, \quad \text{on } \Gamma_D. \tag{5c}$$

In the above equation, Ω is the problem domain, Γ_D and Γ_N are disjoint boundary segments of the domain with $\Gamma_D \cup \Gamma_N = \partial\Omega$. The weak or variational form of these equations can be written as

$$B(u, v) = L(v), \quad \forall v \in H_0^1(\Omega), \tag{6a}$$

$$B(u, v) = \int_{\Omega} \nabla u \cdot \nabla v \, \mathrm{d}\Omega, \tag{6b}$$

$$L(v) = \int_{\Omega} f v \, \mathrm{d}\Omega + \int_{\Gamma_N} g v \, \mathrm{d}s, \tag{6c}$$

where $H_0^1(\Omega)$ is the Sobolev space of functions with square-integrable derivatives and vanishing values on $\partial\Omega$.

The error in any numerical solution can be defined as

$$e = u - u^h, \tag{7}$$

where u is the exact solution field and u^h is the numerical approximation. Since this local definition is not computationally convenient, mathematical norms are introduced to measure the error. One of the attractive mathematical norms, which can be easily related to the weak form in Eq. (6) is the energy of the error:

$$\|e\|^2 = B(e, e) = \int_{\Omega} \nabla e \cdot \nabla e \, \mathrm{d}\Omega. \tag{8}$$

Since the exact solution field is generally unknown, the approximate solution is post-processed to obtain a more accurate measure for the gradient of u^h. This can be done by using recovery based methods such as L^2 or Z^2 [30,31] projection method. Therefore, in general, if u is unknown, e in Eq. (8) is replaced by $e^\star = u^\star - u^h$, where $u^\star$ is the recovered solution field and e^* is an estimate for the true error e. The relative percentage error in the energy norm is

$$\eta = \frac{\|e\|}{\|u\|}, \tag{9}$$

where $\|u\|$ is the exact energy norm.

A simple criterion to achieve a solution with acceptable level of error is [32]:

$$\eta \leqslant \eta_{\max}, \tag{10}$$

where $\eta_{\max}$ is the maximum permissible error percentage in the whole domain, and η is given by

$$\eta = \frac{\|e\|}{(\|u^h\|^2 + \|e\|^2)^{1/2}}. \tag{11}$$

To obtain an economical mesh with high convergence rate, the equi-distribution of the error among all of the elements is used, i.e.,

$$\|e\| = \sqrt{m}\|e\|_i, \tag{12}$$

where m is the number of elements in the mesh. On the basis of the previous equations, we can infer that the error in each element is such that [32]

$$\|e\|_i \leqslant \eta_{\max} \left(\frac{(\|u^h\|^2 + \|e\|^2)}{m} \right)^{1/2} \equiv \bar{e}_m. \tag{13}$$

Hence, those elements that do not satisfy the above equation are refined. If

$$\xi_i = \frac{\|e\|_i}{\bar{e}_m} > 1, \tag{14}$$

the element is refined; otherwise it can be merged. In this study, first a reasonable rectangular mesh is defined over the domain. Each of these elements can be considered as the root of a different tree. If based on Eq. (14), refinement is needed on an element, the particular element is split into four new elements. Each new element has the same aspect ratio as its father and therefore there is no need to check the aspect ratio of the new elements. The above steps are repeated until all the elements satisfy Eq. (13).

6. Numerical results

To investigate the performance of the adaptive technique, a few numerical simulations are carried out. First, the patch test result for the Laplace equation is studied on quadtree meshes with different number of hanging nodes. Then, three Poisson problems that involve steep gradients and singularities within their domain are solved. For the purpose of numerical integration, we use 2×2 Gauss–Legendre quadrature on four-node bilinear quadrilateral elements, whereas on elements with hanging nodes, 25 Gauss points are used in each subtriangle. We interfaced the direct solver, SuperLU [33], to our adaptive code for the solution of the linear system of equations. In the numerical examples, the maximum permissible error in energy norm is set to five percent ($\eta_{\max} = 0.05$).

6.1. Patch test

The patch test for the Laplacian is $\nabla c^2 u = 0$ in $\Omega = (-1, 1)^2$, with essential boundary conditions $u = g(\mathbf{x}) = x_1 + x_2$ imposed on the boundary of the biunit square. The exact solution is $u(\mathbf{x}) = x_1 + x_2$. Consider the three regularized (2:1 rule) meshes shown in Fig. 9. The patch test results are presented in Table 1. The relative error in L^2 and energy norms are $\mathcal{O}(10^{-10})$ and $\mathcal{O}(10^{-10})$, respectively. The second patch test is carried out over non-regularized meshes. The meshes are shown in Fig. 10. The size of adjacent elements are very different, and some elements have more than one hanging node along an edge. This large change in element size does lead to a slight decrease in accuracy—errors in L^2 norm are $\mathcal{O}(10^{-7})$ and $\mathcal{O}(10^{-6})$, respectively. The shape function of node a of mesh Fig. 10b is shown in Fig. 11. The linear behavior of the shape function along the edges can be seen, which demonstrates the conformity

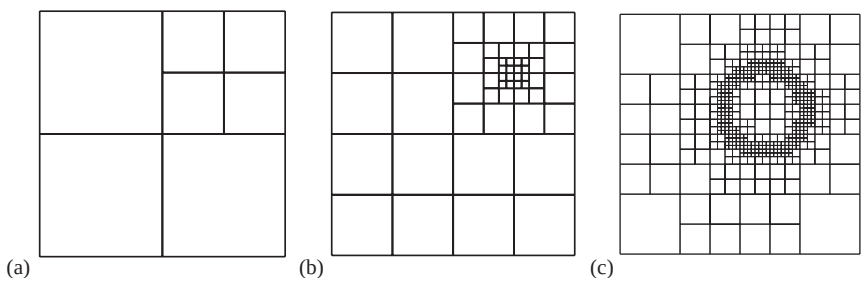

Fig. 9. Patch test on regularized meshes. (a) Mesh a (14 nodes); (b) mesh b (73 nodes) and mesh c (591 nodes).

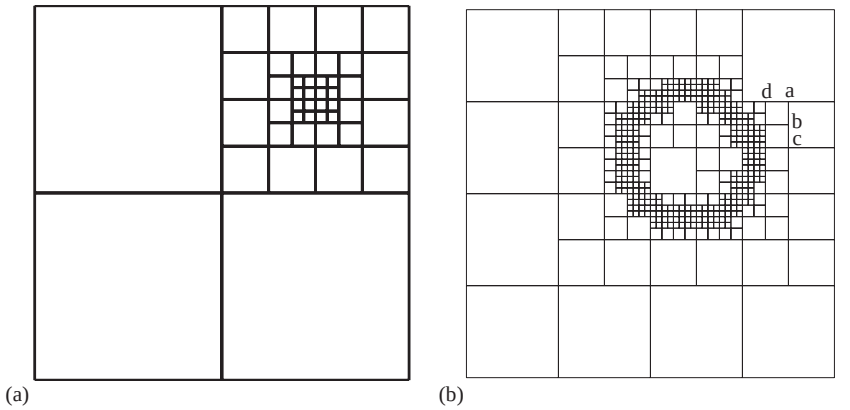

Fig. 10. Patch test on non-regularized elements. (a) Mesh *a* (62 nodes) and (b) mesh *b* (503 nodes).

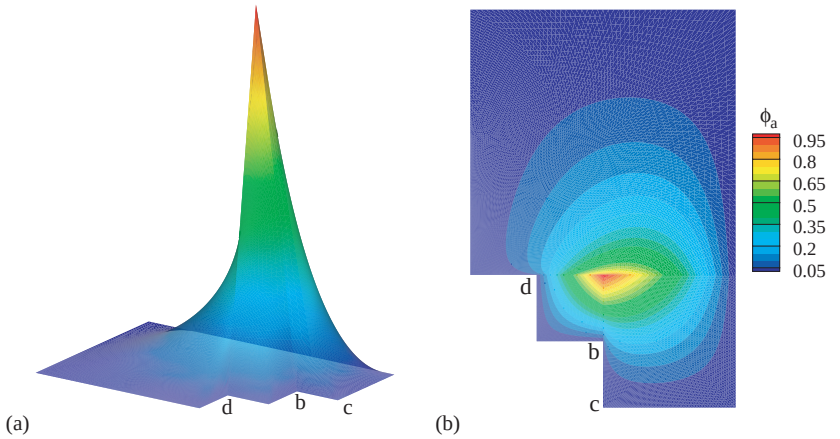

Fig. 11. Shape function of node *a* in Fig. 10b. (a) 3D plot and (b) contour plot.

Table 1
Relative error in the L^2 and energy norm for the patch test

Meshes	Number of elements	Number of nodes	Relative error in L^2 norm	Relative error in energy norm
a	7	14	9.9×10^{-11}	3.3×10^{-10}
b	52	73	3.5×10^{-11}	2.9×10^{-10}
c	496	591	6.8×10^{-10}	8.2×10^{-9}

of the shape function. The above numerical results indicate that sufficient accuracy is obtained in the patch test, and we attribute this to the conformity of the interpolant. On-going research work is addressing the issue of further improvements in the numerical integration.

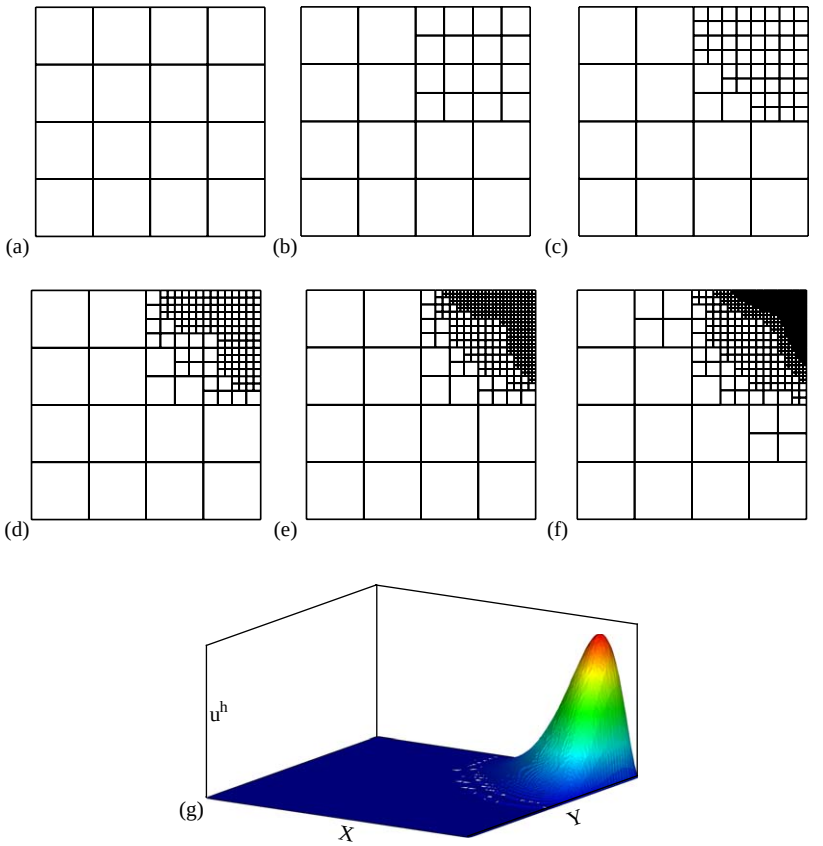

Fig. 12. Successive adaptive refinement and numerical solution for Poisson Problem I. (a) Level 0 (25 nodes); (b) level 1 (41 nodes); (c) level 2 (88 nodes); (d) level 3 (195 nodes); (e) level 4 (464 nodes); (f) level 5 (991 nodes) and (g) u^h.

6.2. *Poisson Problem I*

We solve the Poisson equation over a unit square with Dirichlet boundary conditions:

$$-\nabla^2 u = f, \quad \text{in } \Omega = (0,1)^2, \tag{15a}$$

$$u = 0, \quad \text{on } \partial\Omega. \tag{15b}$$

The source term f is chosen such that the exact solution of the problem is [8]:

$$u(\mathbf{x}) = x_1^{10} x_2^{10} (1 - x_1)(1 - x_2). \tag{16}$$

In Fig. 12, the initial mesh and its refinements are shown. We observe from Fig. 12g that the numerical solution u^h computed on mesh 12f (991 nodes) captures the sharp gradient near (1,1).

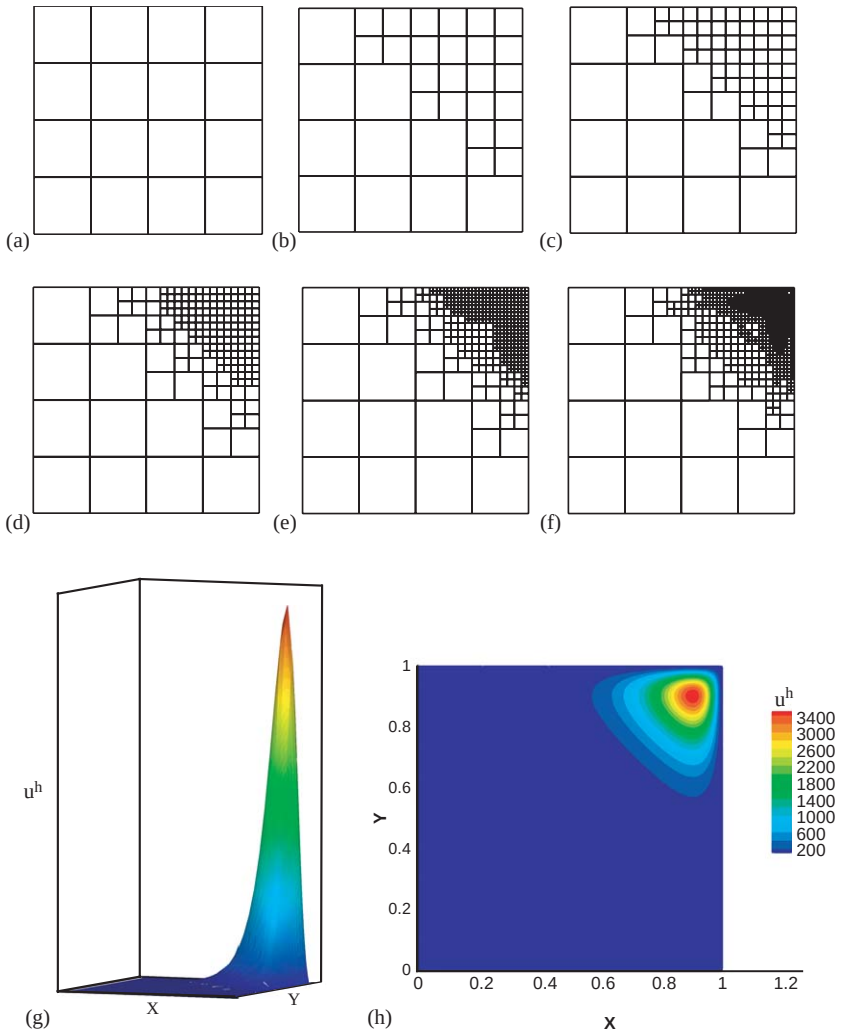

Fig. 13. Successive adaptive refinement and numerical solution for Poisson Problem II. (a) Level 0 (25 nodes); (b) level 1 (49 nodes); (c) level 2 (104 nodes); (d) level 3 (220 nodes); (e) level 4 (538 nodes); (f) level 5 (1347 nodes); (g) 3D plot of u^h and (h) contour plot of u^h.

6.3. *Poisson Problem II*

The second Poisson problem with Dirichlet boundary conditions corresponding to an exact solution [34]

$$u(\mathbf{x}) = 5x^2(1-x)^2(\mathrm{e}^{10x^2}-1)y^2(1-y)^2(\mathrm{e}^{10y^2}-1) \tag{17}$$

is solved on the unit square. The initial and refined meshes are shown in Fig. 13. As one might expect, we obtain mesh refinement in the neighborhood of the region with steep gradients. A 3D plot of the numerical

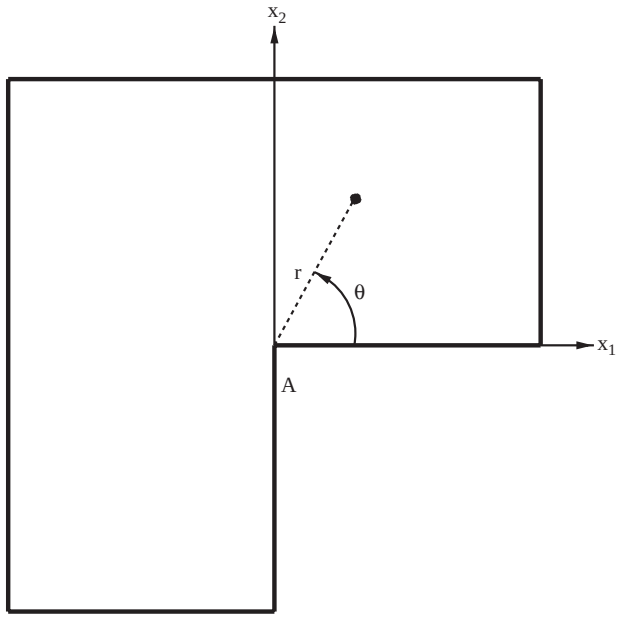

Fig. 14. L-shaped domain.

solution is shown in Fig. 13g, whereas a contour plot of u^h is presented in Fig. 13h. In this example also, the numerical solution is able to resolve the presence of the high gradient in the problem.

6.4. L-shaped domain

Finally, we consider the Laplace equation in the L-shaped region, Ω, shown in Fig. 14. For the Dirichlet problem, the exact solution is [35,36]:

$$u(r, \theta) = r^{2/3} \sin\left(\frac{2\theta}{3}\right), \tag{18}$$

where r and θ are measured with respect to vertex A. The exact solution is used to impose the essential boundary conditions on $\partial\Omega$. Since the derivatives of u are singular at the origin, we expect to obtain refinement near the re-entrant corner. In Fig. 15, the initial mesh and the refined meshes are shown. As expected, the quadtree mesh provides high resolution in an area surrounding the re-entrant corner. A contour plot of the numerical solution is depicted in Fig. 15g.

7. Concluding remarks

In this paper, we have presented a new method for h-adaptive mesh refinement. We used the natural neighbor interpolant to construct conforming approximation over non-compatible quadtree meshes. The process of constructing shape functions over physical elements is similar to classical finite elements. First, we defined Laplace shape functions over reference elements [9], and then on using an isoparametric

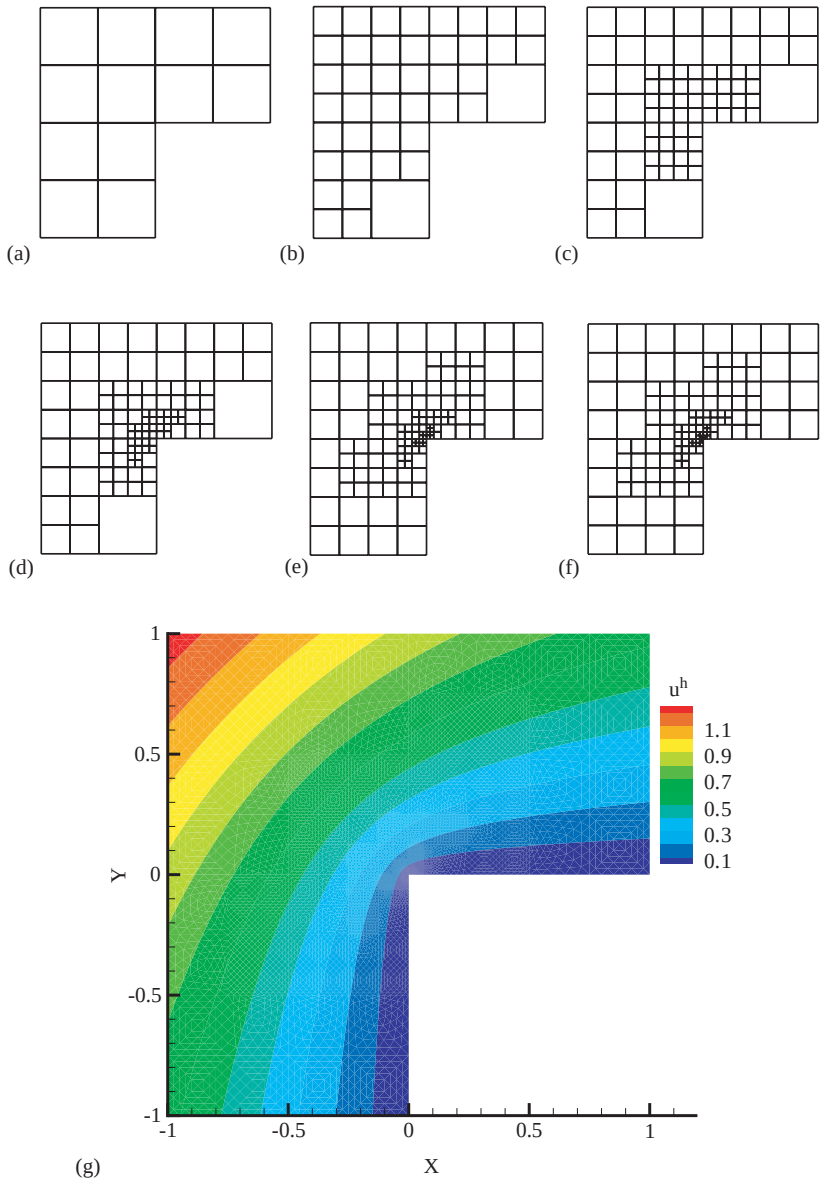

Fig. 15. Successive adaptive refinement near a corner singularity. (a) Level 0 (21 nodes); (b) level 1 (59 nodes); (c) level 2 (103 nodes); (d) level 3 (138 nodes); (e) level 4 (179 nodes); (f) level 5 (192 nodes) and (g) contour plot of u^h.

map, the shape functions over physical elements were obtained. The striking advantages of this technique are its simplicity and generality. The technique preserves continuity on edges with multiple-nodes, and avoids the need to use Lagrange multipliers or multi-point constraints, as is usually required. Numerical results for the patch test and Poisson problems were presented. The L^2 and energy error norm results for the patch test were sufficiently accurate, and the method was able to capture the sharp gradients and

singularity in the Poisson problems. The efficiency of the quadtree data structure in combination with the attractive properties of the Laplace interpolant results in a simple and robust mesh refinement technique. Further verification studies and implementation of this *h*-adaptive method for solving elasticity problems involving crack discontinuities and singularities are currently under investigation.

Acknowledgements

The financial support of this work by the National Science Foundation through research award CMS-0352654 to the University of California, Davis, is gratefully acknowledged.

References

[1] H. Samet, Application of Spatial Data Structure, Addison-Wesley, New York, 1990.
[2] M. Yerry, M. Shephard, A modified-quadtree approach to finite element mesh generation, IEEE Comput. Graph. Appl. 3 (1) (1983) 39–46.
[3] D.M. Greaves, A.G.L. Brothwick, Hierarchical tree-based finite element mesh generation, Int. J. Numer. Methods Eng. 45 (1999) 447–471.
[4] N. Palle, J.A. Dantzig, An adaptive mesh refinement scheme for solidification problems, Metall. Mater. Trans. 27A (1996) 707–718.
[5] N. Provatas, N. Goldenfeld, J. Dantzig, Adaptive mesh refinement computation of solidification microstructures using dynamic data structures, J. Comput. Phys. 148 (1) (1999) 265–290.
[6] P. Krysl, P. Grinspun, E. Schröder, Natural hierarchical refinement for finite element methods, Int. J. Numer. Methods Eng. 56 (8) (2003) 1109–1124.
[7] P. Krysl, A. Trivedi, B. Zhu, Object-oriented hierarchical mesh refinement with CHARMS, Int. J. Numer. Methods Eng. 60 (8) (2004) 1401–1424.
[8] P. Kegan, A. Fischer, P.Z. Bar-Yoseph, Mechanically based models: adaptive refinement for B-spline finite element, Int. J. Numer. Methods Eng. 57 (2003) 1145–1175.
[9] N. Sukumar, A. Tabarraei, Conforming polygonal finite elements, Int. J. Numer. Methods Eng. 61 (12) (2004) 2045–2066.
[10] N.H. Christ, R. Friedberg, T.D. Lee, Weights of links and plaquettes in a random lattice, Nucl. Phys. B 210 (3) (1982) 337–346.
[11] V.V. Belikov, V.D. Ivanov, V.K. Kontorovich, S.A. Korytnik, A.Y. Semenov, The non-Sibsonian interpolation: a new method of interpolation of the values of a function on an arbitrary set of points, Comput. Math. Math. Phys. 37 (1) (1997) 9–15.
[12] H. Hiyoshi, K. Sugihara, Two generalizations of an interpolant based on Voronoi diagrams, Int. J. Shape Model. 5 (2) (1999) 219–231.
[13] E.L. Wachspress, A Rational Finite Element Basis, Academic Press, New York, 1975.
[14] M. Meyer, H. Lee, A.H. Barr, M. Desbrun, Generalized barycentric coordinates for irregular polygons, J. Graph. Tools 7 (1) (2002) 13–22.
[15] M.S. Floater, Mean value coordinates, Comput. Aided Geom. Des. 20 (1) (2003) 19–27.
[16] R. Sibson, A vector identity for the Dirichlet tesselation, Math. Proc. Cambridge Philos. Soc. 87 (1980) 151–155.
[17] A. Okabe, B. Boots, K. Sugihara, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, Wiley, Chichester, England, 1992.
[18] T.J.R. Hughes, The Finite Element Method, Prentice-Hall, Englewood Cliffs, NJ, 1987.
[19] N. Sukumar, B. Moran, A.Y. Semenov, V.V. Belikov, Natural neighbor Galerkin methods, Int. J. Numer. Methods Eng. 50 (1) (2001) 1–27.
[20] E. Cueto, N. Sukumar, B. Calvo, M.A. Martínez, J. Cegoñino, M. Doblaré, Overview and recent advances in natural neighbor Galerkin methods, Arch. Comput. Methods Eng. 10 (4) (2003) 307–384.
[21] H. Samet, The quadtree and related hierarchical data structures, ACM Comput. Surv. 16 (2) (1984) 187–260.
[22] H. Samet, R.E. Webber, Hierarchical data structures and algorithms for computer graphics, I. Fundamentals, IEEE Comput. Graph. Appl. 8 (3) (1988) 48–68.

[23] H. Samet, R.E. Webber, Hierarchical data structures and algorithms for computer graphics, II. Applications, IEEE Comput. Graph. Appl. 8 (4) (1988) 59–75.
[24] G.M. Hunter, K. Steiglitz, Operations on images using quadtree, IEEE Trans. Pattern Anal. Mach. Intell. 1 (1979) 145–153.
[25] H. Samet, Neighbour finding techniques for images represented by quadtrees, Comput. Graph. Image Process. 18 (1982) 37–57.
[26] P.L. Baehmann, S.L. Wittchen, M.S. Shephard, K.R. Grice, M.A. Yerry, Robust, geometrically based, automatic two-dimensional grid generation, Int. J. Numer. Methods Eng. 24 (1987) 1043–1087.
[27] K.F.C. Yiu, D.M. Greaves, S. Cruz, A. Saalehi, A.G.L. Borthwick, Quadtree grid generation: information handling, boundary fitting and CFD application, Comput. Fluids 25 (8) (1996) 759–769.
[28] T. Belytschko, M. Tabbara, H-adaptive finite element methods for dynamic problems, with emphasis on localization, Int. J. Numer. Methods Eng. 36 (1993) 4245–4265.
[29] D. De Zeeuw, K.G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations, J. Comput. Phys. 104 (1993) 56–68.
[30] O.C. Zienkiewicz, J.Z. Zhu, The superconvergent patch recovery and a posteriori error estimates, Part 1: The recovery technique, Int. J. Numer. Methods Eng. 33 (1992) 1331–1364.
[31] O.C. Zienkiewicz, J.Z. Zhu, The superconvergent patch recovery and a posteriori error estimates, Part 2: Error estimates and adaptivity, Int. J. Numer. Methods Eng. 33 (1992) 1365–1382.
[32] O.C. Zienkiewicz, R.L. Taylor, The Finite Element Method, vol. 1, fifth ed., Butterworth-Heinemann, New York, 2000.
[33] J.W. Demmel, J.R. Gilbert, X.S. Li, SuperLU User's Guide, Technical Report LBNL-44289, Lawrence Berkeley National Laboratory, Berkeley, CA, 2003, URL http://crd.lbl.gov/~xiaoye/SuperLU/.
[34] M. Ainsworth, J.T. Oden, A Posteriori Error Estimation in Finite Element Analysis, Wiley, New York, 2000.
[35] G.F. Carey, Computational Grids, Generation, Adaptation, and Solution Strategy, Taylor and Francis, Washington DC, 1997.
[36] I. Babuška, W.C. Rheinboldt, Reliable error estimation and mesh adaptation for the finite element method, in: J. Oden (Ed.), Computational Methods in Nonlinear Mechanics, North-Holland, Amsterdam, The Netherlands, 1980, pp. 67–108.