Implementation of a General Mesh Refinement Technique

Paul Hammon[1] and Petr Krysl[2]

PK: Dedicated to Zdenek Bittnar, mentor and friend.

Abstract In this paper, we describe a pilot implementation of a hierarchical adaptive mesh refinement technique. While the focus is on the implementation in one-dimensional problems, the methodology is generic in nature in that it is equally applicable to the construction of adaptive approximations in multi-dimensional domains, as is also being demonstrated on some examples of steady-state heat conduction and other potential-like equations (EEG).

Keywords: adaptive mesh refinement, refinement equation.

1 Introduction

The finite element method is an approach to finding approximate solutions to differential equations. This approach works by breaking up the domain of the problem into a set of elements, which comprise a region between nodes and the trial solutions defined over those regions. Next, a technique such as the Galerkin weighted residual method is used to construct a trial function as a linear combination of the basis functions that approximates the solution to the differential equation. This paper is concerned with the construction of a set of basis functions on a collection of hierarchically arranged finite elements (a hierarchical mesh). For the sake of simplicity, this paper will only treat the one-dimensional case in detail, but the technique described in this paper is applicable to multi-dimensional problems [KGS (2002) and GKS (2002)]. Let us also note that application to high-order basis functions is equally straightforward.

In this paper we describe a hierarchical adaptive refinement technique for use in the finite element method and discuss one particular implementation of this technique in an object-oriented computer language (Java). The strengths of the theoretical approach are its simplicity, especially compared to current mesh refinement techniques, and its ability to create conforming approximations in two, three, and higher-dimensional cases, for a wide range of elements types and approximation orders without any special tricks [KGS (2002)].

2 Overview of Mesh Refinement

In recent years there has been much interest in hierarchical and multigrid methods for refining finite element meshes. The primary benefit of these techniques is their increased efficiency that results from adaptive meshing, i.e. from the improvement of the existing mesh which more efficiently represents the pertinent parts of the problem.

Adaptive mesh refinement as discussed, for instance, in Reference [BKS (1998)] is geometric in nature, and involves refinement by splitting elements in areas of low accuracy and merging elements in areas of high accuracy. In order to prevent the creation of an inconsistent or poor-quality mesh, techniques such as this require special algorithms to improve and control the quality of the mesh. The advantage of this method is that it can effectively create a highly non-uniform mesh. Approaches such as this have the drawback that the algorithms used to refine the mesh are rather complicated. The main reason is the necessity to maintain compatibility. Hierarchical approaches to mesh

[1] Undergraduate student, University of California, San Diego, CA, U.S.A.

[2] Professor, University of California, San Diego, U.S.A

refinement have been shown to yield stiffness matrices with a condition number which does not increase as quickly with refinement as it otherwise would [YSE (1986)]. Creating nested meshes is attractive also from the point of view of availability of highly efficient multilevel solvers for systems of linear equations resulting from the refined approximations.

The hierarchical refinement approach described in this paper is particularly intuitive and simple to understand. It leads naturally to what Yserentant describes as a *self-similar structure* in [YSE (1992)]. The basic idea of the present approach is that the refinement is applied to basis functions, not to isolated pieces of basis functions as defined over each element. Each refined mesh exists on its own refinement level. The one requirement our technique imposes on the meshing of new hierarchical levels is that nodes from lower levels are retained in higher refinement levels. It is this requirement which ensures compatibility and removes the need for complex mesh integrity checks and repair operations.

3 Conforming Hierarchical Mesh Refinement

For the benefit of the reader we shall very concisely summarize the setting in which we wish to introduce our implementation. It should be emphasized that our discussion may indeed be entirely limited to the one-dimensional setting, without any loss of generality. As we shall show, our use of the refinement (multi-resolution) equation guarantees that our results carry over to higher dimensions without change.

3.1 The Galerkin Approximation Method

We are going to introduce the basic concepts for two-point boundary-value problems of the form $u'' + f = 0$, with boundary conditions $u(1) = g$ and $-u'(0) = h$. We will be using the Galerkin approximation method [HUG (1987)], which involves the adoption of a set of basis functions N_A, where $N_A(1) = 0$ for A = 1, 2, …, n and $N_{n+1}(1) = 1$. Using the matrix notation, we write $\boldsymbol{Kd} = \boldsymbol{F}$, where $\boldsymbol{K}$ is an n by n matrix, $\boldsymbol{d}$ is an n by 1 vector, and $\boldsymbol{F}$ is an n by 1 vector:

$$K_{AB} = \int_0^1 N_{A,x} N_{B,x}\, dx$$

and

$$F_A = \int_0^1 N_A\, f dx + N_A(0)h - g\int_0^1 N_{A,x} N_{n+1,x}\, dx$$

3.2 Conceptual Overview of Hierarchical Refinement

In our hierarchical mesh refinement technique, we will be constructing a hierarchy of approximation spaces denoted $V^{(j)}$, where $V^{(0)}$ is the coarsest space and each successive $V^{(i)}$ represents a finer-scale space. $V^{(j)}$ is a collection of all the basis functions defined on the mesh $M^{(j)}$ such that:

$$V^{(j)} = \{N_i^{(j)}(x) \mid N_i^{(j)} \text{is supportedon } M^{(j)}\}$$

In short, the set of functions $N^{(j)}(x)$ form a basis for $V^{(j)}$. We construct the spaces $V^{(j)}$ so that each new $V^{(j)}$ is a subset of each of the coarser spaces in the refinement tree. It is this condition which ensures that the entire set of refinement levels are consistent. This requirement can be written as:

$$V^{(0)} \subset V^{(1)} \subset V^{(2)} \subset \ldots \subset V^{(j)}$$

This nested aspect of the refinement spaces indicates that any basis function $N_i^{(j)}(x) \in V^{(j)}$ can be written as a linear combination of basis functions in a finer space:

$$N_i^{(a)}(x) = \sum_k b_{ik}^{(b)} N_k^{(b)}(x), \quad b > a \qquad (1) \qquad (6)$$

Equation (1) motivates our hierarchical refinement technique. Using this relation, we can enhance resolution by deactivating one or more basis functions in $V^{(a)}$ and activating functions on level $V^{(b)}$, where b > a, to replace the deleted functions (see [KGS (2002)]).

4 Hierarchical Refinement Algorithm

The hierarchical technique developed in this paper can be easily understood with reference to a diagram. The node one wishes to refine about on the initial coarse level is deactivated, and a new refinement level is created. This new level adds several new nodes but also maintains all nodes present in levels lower down in the refinement tree. Carrying over nodes from lower refinement levels onto new levels of mesh refinement ensures compatibility between refinement levels without necessitating any complex checks of mesh continuity. It is not necessary to use only three basis functions in a refinement level; three are used in the above example

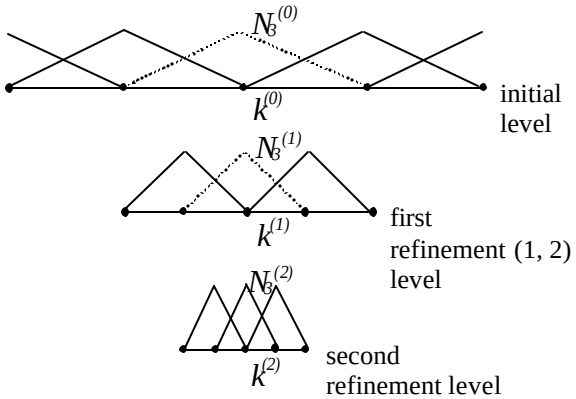

Figure 1: Two levels of hierarchical mesh refinement about node k. Dashed lines indicate omitted basis functions.

only for sake of the argument. Any number of new basis functions can be introduced in each refinement level. As is evident from Figure 1, multiple refinement levels can be created to increase accuracy in regions of interest.

There are two different ways to represent a family of refinement levels such as the one shown in Figure 1. The refinements can be represented as a set of distinct levels, each of which reference their immediate parent and child refinement levels. This is shown in Figure 1. Alternatively, since each node to be refined about is deactivated, all refinement levels can be "flattened" into a single level. Figure 2 shows a "flattened" representation of the family of refinement levels shown in Figure 1.

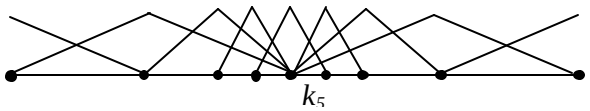

Figure 2: Flattened representation of refinement.

A derefinement technique does just the opposite of the refinement technique. A set of basis functions about a node can be deactivated and replaced by a single node in order to decrease mesh resolution. This process is shown in Figure 3.

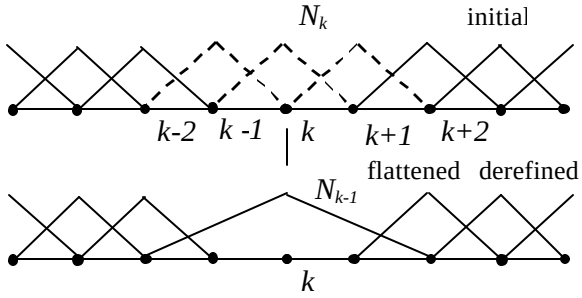

Figure 3: Basic derefinement about node k.

The basis function at the site of the refinement (node k in this example), along with the basis functions immediately before and after it, are deactivated and replaced with a single basis function which spans from $k - 2$ to $k + 2$ and peaks at k. Note that this most basic form of derefinement leads to meshes which are slightly different from those shown in Figs. 1 and 2 because there are more nodes than basis functions. This is undesirable in our implementation, so we have devised a modified derefinement algorithm that adjusts the basis functions on either side of the newly derefined basis function so that they both terminate in the node below the derefined basis function (node k in our example). The modified derefinement algorithm applied to a flattened mesh is shown in Figure 4.

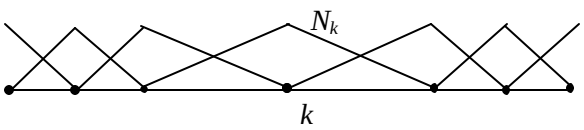

Figure 4: Modified flattened derefinement about node k.

After a series of refinements about nodes of interests and derefinements in mostly slow-variation areas, this method creates a highly irregular yet consistent and robust meshing of the domain. While refinement and derefinement are presented in this paper in the context of one-dimensional domains, the basic method is extensible in an analogous way to higher-dimensional function spaces [KGS (2002)].

5 Implementation

The implementation of our hierarchical refinement technique was coded in Java and uses polynomial and matrix algebra [BES (2001)] and Ptplot plotting [LH (2001)]. Objects to represent each of the different components used in this technique had been defined, at different levels of abstraction. Each object contains several data objects and methods to modify and interpret the data stored on those objects. At the lowest level of abstraction is the **Cell**, which contains data about the two nodes between which it is define.d. Next, we have **BFunSegment**, which contains the **Cell** over which it is defined, and the segment of the nodal basis function over which it is defined. **BFunSegment** also keeps track of its parent nodal basis function. The class **Node** represents a nodal basis function. It consists of a

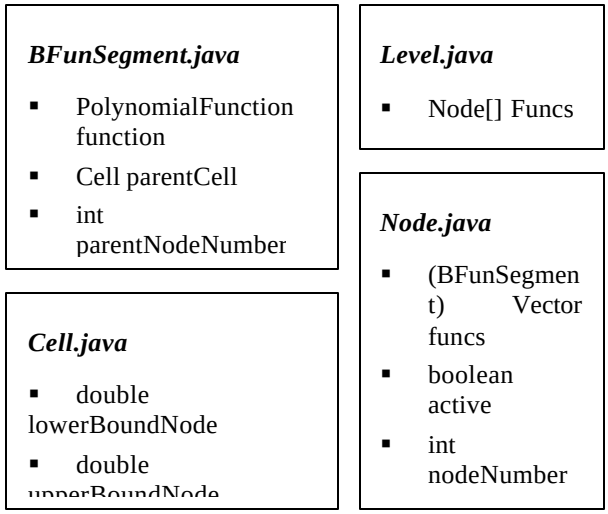

Figure 5: Main classes in Java implementation.

collection of **BfunSegments**, its number in the refinement level, and the position of its peak. Finally, the class **Level** consists of the set of nodal basis functions for the entire level. This information is summarized in Figure 5.

The algorithms used for refinement, derefinement, and flattening the mesh are all relatively straightforward. Graphical representations of these processes can be seen in Section 3 of this paper. The basic operation of our implementation is to refine or derefine at a single nodal basis function then flatten the initial mesh and refined mesh.

Pseudo-code versions of these algorithms follow:

Refine Nodal Basis Function N_r:

- deactivate **Node** N_r
- create new **Level** with **Cell** size half that of the previous **Level**. Increment *refinement* of nodes on new **Level**. Activate **Nodes** k_{r-1} to k_{r+1} on new **Level**.

Derefine Nodal Basis Function N_r:

- deactivate N_r, N_{r-1}, and N_{r+1}
- create a new **Level** of only two **Cells**, from k_{r-2} to k and from k to k_{r+2}
- remove **Nodes** k_{r-1} and k_{r+1} on the initial **Level**
- move termination of N_{r-2} and N_{r+2} to **Node** k_r

Flatten Mesh and Reorder Basis Functions

- collect set of **Nodes** from initial and refinement **Level**, sort them by position of **Node** maximum, and create the new flattened **Level** with this set of **Nodes**

6 Refinement Example

The following example demonstrates how our refinement technique and computer implementation work. Our implementation used the Galerkin weighted residual method to compute the approximate solution to the differential equation $u'' + f = 0$ with the boundary conditions $u(1) = g$ and $-u'(0) = h$ over the interval [0,1] using the refined mesh of nodal basis functions. A polynomial f is used in this example. The known u is being used to derive g, h, and f. For this example,

$$u = 100x^5 - 300x^4 + 325x^3 - 150x^2 + 25x,$$

and working backwards, we find

$$u(1) = g = 0 \quad \text{and} \quad -u'(0) = h = -25, \quad \text{and}$$
$$f = -2000x^3 + 3600x^2 - 1950x + 300.$$

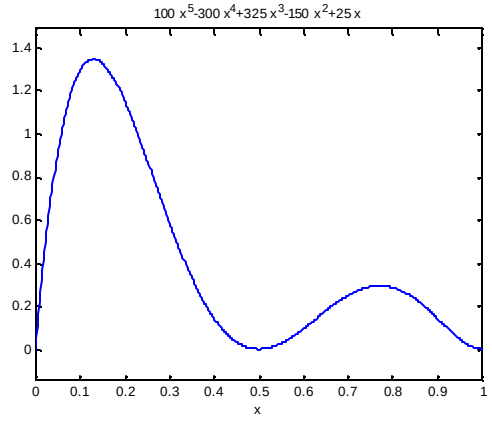

Figure 6: Function to be interpolated using finite elements.

Graphing u over the interval [0,1] yields Figure 6.

Applying the Galerkin method to this problem with 10 uniform cells and the g, h, and f from above yields the following graphs: Figure 7a shows the initial uniform mesh of nodal basis functions. Applying the Galerkin method results in a set of weighting factors for each of the basis functions, and figure 7b shows the weightings applied to the basis functions. Finally, the contributions of all the basis functions are added together over each cell, and the resulting piece-wise function is the Galerkin solution to the differential equation (Figure 7c). To improve this approximation, refine about nodes that correspond to areas of rapid change. The refined mesh and solution are shown in Figure 8.c

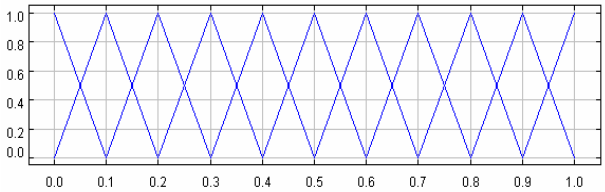

Figure 7a: Unweighted uniform mesh.

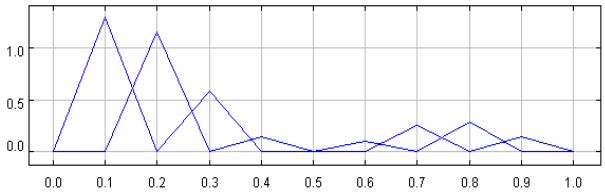

Figure 7b: Weighted basis functions.

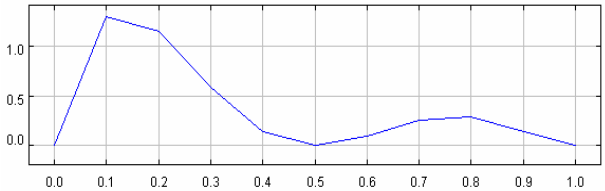

Figure 7c: Galerkin solution.

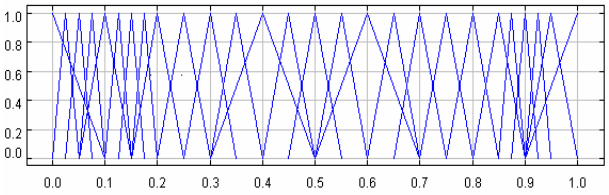

Figure 8a: Unweighted flattened two-level mesh.

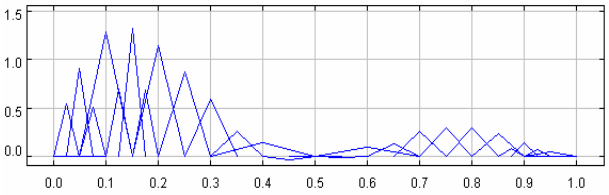

Figure 8b: Refined mesh with weighted basis functions.

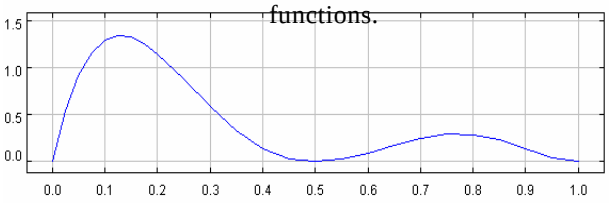

Figure 8c: Refined Galerkin solution.

Note that this refined mesh in Figure 8 is non-uniform, yet with conforming. This results from the fact that refined meshes constructed using our hierarchical

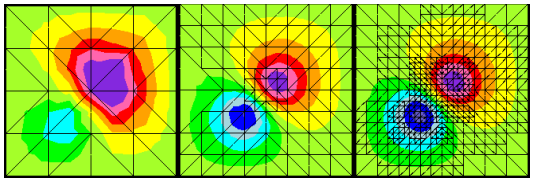

Figure 9: Galerkin solution of a dipole equation in a square domain. Mesh of 6-node quadratic triangles. Left to right: initial, and two refined meshes.

refinement procedure are consistent by design, and yet can produce highly non-uniform meshes. The second refinement level of the mesh (Figure 8b) contains elements that are one-fourth the size of the initial mesh.

7 Examples in two and three dimensions

with

It is very important to realize that the only assumption as to the properties of the basis was the refinability (refinement) equation (1). Otherwise the basis is arbitrary (i.e. in addition to continuity as dictated by the Galerkin form). In particular, the basis may be constructed over the real line, over two-, three- or even higher dimensional domains. Also the order is arbitrary: linear, qudratic, cubic, Lagrange or Hermite, polynomial or non-polynomial. This *genericity* is very powerful as will be also shown on some examples in this section. See also discussion in References [KGS and GKS (2002)], which includes simulations on subdivision surfaces.

The first example demonstrates the refinement of the approximation spaces constructed on mesh of six-node (quadratic) triangles in a square domain. The dipole equation function is shown in the contours displayed on the computational meshes in Figure 9. Note that there are *no hanging nodes*, i.e the basis is naturally conforming, even though the picture might suggest otherwise. The triangles shown in Figure 9 are the *integration cells*, and the integration cells support basis functions of different resolutions. That is why it appears as if the refinement was non-conforming.

The last example deals with a three-dimensional biomechanics problem. Figure 10 shows the solution to a point source Poisson equation solved on the tetrahedral mesh of the human brain, which is the setup commonly encountered in inverse EEG. Four mesh refinements have been applied. The refinement of approximations on tetrahedral meshes is described in detail in Reference [EK (2002)]. The natural hierarchical refinement described in this paper is applicable without change. However, for this particular finite element shape the issue is to ensure that the tetrahedral cells resulting from purely geometrical division of the original mesh cells be of guaranteed bounded shape quality measure. The approach described in [EK (2002)] attains this property through the use of the so-called Kuhn octasection with careful alternation of the division stencil between refinement levels.

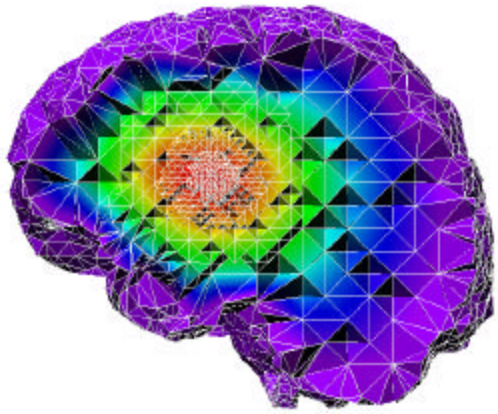

Figure 10: Galerkin solution for a point-source forcing for the Poisson equation in the human brain. Mesh of 4-node tetrahedra.

8 Conclusions

The hierarchical refinement method described in this paper has several major advantages. The refined meshes created using this method are compatible by construction and therefore do not require any complicated algorithms designed to ensure compatibility of the refined mesh. Although this paper presents the details of a one-dimensional case, this process is easily extensible to higher-dimensional (two, three, and more) domains [KGS (2002)]. Note that the theoretical outline given in Section 3 at no place invokes the dimension of the domain supporting the function space V—this is because the refinement technique is not dimension-specific. The refinement equation applies equally well to higher-dimensional spaces as it does to the one-dimensional case treated in this paper. Therefore, one of the major advantages of out technique is its easy extensibility to higher dimensions without imposing added complications to the refinement algorithm. Additionally, because this refinement scheme is based on a fixed-subdivision stencil, elements on higher refinement levels do not become ill-shaped (this applies also to higher-dimensional spaces, and to approximations on tetrahedral meshes, which are notoriously difficult to refine) [EK (2002)]. The proposed approach is exceedingly simple, and does not invoke any special tricks to produce conforming approximations. The implementations may be correspondingly simple and robust, an appealing contrast to existing techniques.

Acknowledgements: Support for this research by NSF/Darpa (DMS-9874082) and the Hellman Foundation Fellowship (2001) is gratefully acknowledged.

References:

[BDY (1998)] Bank, R.E., Dupont, T., Yserentant, H. (1998): *The Hierarchical Basis Multigrid Method*. Numer. Math. 52, 427-458.

[BES (2001)] Besset, D.H. (2001): *Object-Oriented Implementation of Numerical Methods: an Introduction with Java and Smalltalk*. New York: Academic Press.

[BKS (1998)] Bottasso, C. L., Klass, O.; Shepard, M.S. (1998): *Data Structures and Mesh Modification Tools for Unstructured Adaptive Techniques*. Engineering with Computers. 14, 235-247.

[EK (2002)] Endres, L; Krysl, P. (2002): *Refinement of Tetrahedral Meshes with Guaranteed Quality*, submitted.

[HUG (1987)] Hughes, T.J.R. (1987): *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Englewood Cliffs, NJ: Prentice Hall.

[GKS (2002)] Grinspun, E, P. Krysl, and P. Schröder (2002): CHARMS: A Simple Framework for Adaptive Simulation. *Proceedings of SIGGRAPH 2002, and ACM Transactions on Graphics*, to appear.

[KGS (2002)] Krysl, P., Grinspun, E., Schröder, P. (2002): *Natural Hierarchical Refinement of Finite Element Meshes*, to appear, Int. J. Numerical Methods of Engineering.

[LH (2001)] Lee, E. A.; Hylands, C. (2001): Ptolemy II. http://ptolemy.eecs.berkeley.edu/java/ptplot5.1p1/ptolemy/plot/doc/index.htm

[YSE (1992)] Yserentant, H. (1992): *Hierarchical Bases*. "ICIAM 91" (O'Malley, R.E., ed.), SIAM, Philadelphia.

[YSE (1986)] Yserentant, H. (1986): *On the Multi-Level Splitting of Finite Element Spaces*. Numer. Math. 49,379-412