Natural Hierarchical Refinement for Finite Element Methods

Petr Krysl[1,*], Eitan Grinspun[2], Peter Schröder[2]

[1] *University of California, San Diego, Structural Engineering, Mail code 0085, 9500 Gilman Dr, La Jolla, CA 92093-0085,*
[2] *California Institute of Technology, Computer Science, Mail code 256-80 Pasadena, CA 91125.*
pkrysl@ucsd.edu, eitan@cs.caltech.edu, ps@cs.caltech.edu

KEY WORDS: Finite element, mesh refinement, hierarchical, adaptive approximation, subdivision surface, subdivision element method

SUMMARY

Current formulations of adaptive finite element mesh refinement seem simple enough, but their implementations prove to be a formidable task. We offer an alternative point of departure which yields equivalent adapted approximation spaces wherever the traditional mesh refinement is applicable, but our method proves to be significantly simpler to implement. At the same time it is much more powerful in that it is general (no special tricks are required for different types of finite elements), and applicable for some newer approximations where traditional mesh refinement concepts are not of much help, for instance on subdivision surfaces.

Introduction

Adaptive finite element computations rely on adjustments of the spatial resolution of the domain discretization to deliver higher accuracy where it is needed. When the domain is discretized into a finite element mesh, a possible option, albeit somewhat expensive and in some cases complex, is to create a new mesh with the desired resolution, i.e., *remeshing*. Another alternative is to adjust the density of the mesh by performing local *refinement* (resp. unrefinement) of the existing mesh so that in some regions finite elements are split to decrease their "size", in other regions they are merged to reduce the resolution. Both choices, remeshing and refinement, have their advantages and disadvantages. We are not going to argue for one or the other option. Rather, we assume that refinement had been adopted as the method of choice.

What are the desirable properties of a mesh refinement algorithm? It should certainly be *efficient* in that it should not become a bottleneck of the adaptive computations; it should generate *good quality* geometric meshes, i.e., elements must remain well-shaped on refinement and unrefinement; and, finally, it must be *robust*, which is usually expressed as the requirement to terminate with a valid result in finite time. An additional plus is if it generates *nested* meshes in the refinement hierarchy, which simplifies the incorporation of multigrid solvers [2, 20].

State of the art mesh refinement algorithms adopt the viewpoint that the centerpiece of refinement is the geometric division of finite elements. However, the result of local refinement based on element-by-element splitting is that it does not in general ensure global *compatibility* of the modified (refined) mesh. A number of approaches are being used to resolve this issue (see Reference [4] for a good introduction): (i) the unknowns of the incompatibly placed nodes are constrained with respect to other nodes so that compatibility of the resulting approximation is ensured even though the mesh remains incompatible; (ii) incompatibility is treated with Lagrangian multipliers or penalty methods; (iii) compatibility of the mesh is achieved by splitting additional elements until the mesh becomes globally compatible by construction. A number of specialized mesh refinement schemes have been proposed for a variety of practically important cases, for triangular and quadrilateral meshes in two dimensions [17, 3, 15, 16], tetrahedral [19, 9, 10, 14, 13, 1], or hexahedral meshes in three dimensions [8]. A critical review of the existing adaptive algorithms based on mesh refinement leads to the conclusion that they tend to be quite complex (constraint methods, splitting of neighboring elements), or lead to undesirable algorithmic features (Lagrange multipliers, penalty methods). A general, flexible, and robust mesh refinement algorithm that would be at the same time *simple* to implement is very desirable. However, there is at least one other motivating factor for research in this area: some recent approximation spaces used in the mechanics of thin shells (a set of fourth-order partial differential equations) rely on the notion of a subdivision basis [5]. This basis is constructed by the repeated application of a subdivision stencil to a control polygon of arbitrary connectivity, which in the limit leads to a C^1 basis function associated with any given control vertex (node), supported on *two rings* of triangles around the given node. This enlarged support of the basis functions means that suddenly the traditional concept of finite element refinement is no longer applicable. In other words, the basis functions do not consist of isolated pieces over each element incident to the node, and hence it is not possible to split the triangles in isolation. Therefore, a more general approach is needed to encompass these newer discretization methods.

To summarize, the question suggests itself whether the mesh refinement issue is viewed from the best angle if tackled from a traditional perspective, or whether there is possibly an alternative viewpoint that would lead to the right kind of questions, and hence to better answers.

As we show in this paper there *is* an approach which is at the same time much *simpler* and much *more general* than current techniques. This alternative approach exploits *refinement of basis functions* rather than refinement of elements. It is in spirit much closer to some recent developments in the design of meshless methods. Our approach meets the above desiderata of mesh refinement ab initio, in any number of spatial dimensions, and for a much wider variety of finite element types than any standard mesh refinement algorithm.

1. Adaptive Basis Function Refinement

Consider a set of linearly independent scalar basis functions $\{\phi_i(x)\}$ with span $\mathcal{X}$,

$$\mathcal{X} = \left\{ v(x) : v(x) = \sum_i \phi_i(x) v_i \right\} , \tag{1}$$

with $x \in R^n$, $v(x) \in R^m$, $v_i \in R^m$ ($n \geq 1$, $m \geq 1$), and $\phi_i(x) \in R$.

For the moment, take a one-dimensional ($n = 1$) finite element mesh M, with linear nodal basis functions $\phi_i(x)$, where i is the node index. The finite element approach dictates that the function $\phi_i(x)$ is constructed piecewise over individual finite elements that share node i. In this setting, the concept of uniform refinement is well understood. For example, a uniform bisection refinement of the mesh M produces another mesh, M'. The set of functions $\phi'_i(x)$ constructed on mesh M' constitutes a basis for the *finer* space $\mathcal{X}'$, which is related to the *coarser* space $\mathcal{X}$ by inclusion:

$$\mathcal{X} \subset \mathcal{X}' . \tag{2}$$

Now let us progress to general finite element meshes, in any number of dimensions ($n > 0$), and of any approximation order. Let us *assume* that given a coarser mesh M with the associated basis

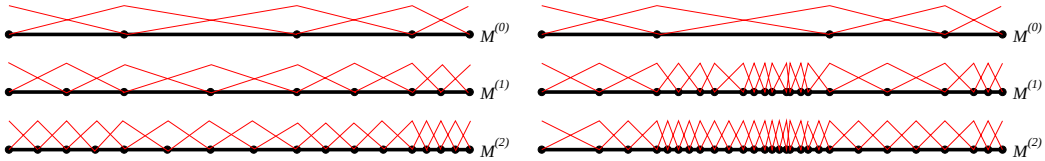

Figure 1. Hierarchy with (left) uniform/stationary or (right) non-uniform/non-stationary refinement.

functions $\phi_i(x)$ it is possible to construct a finer mesh M' and associated basis functions $\phi'_i(x)$ such that the nesting relation (2) holds. Recursive application of this one level uniform refinement results in a *hierarchy* of approximation spaces $\mathcal{X}^{(j)}$. The coarsest space in the hierarchy is $\mathcal{X}^{(0)}$, progressively finer spaces have increasing indices $\mathcal{X}^{(1)}$, $\mathcal{X}^{(2)}\ldots$, and this sequence may be either infinite or finite. In this setting, the analogy to (1) is

$$\mathcal{X}^{(j)} = \left\{ v(x) : v(x) = \sum_i \phi_i^{(j)}(x) v_i^{(j)} \right\} . \tag{3}$$

By repeated composition of (2) we arrive at the hierarchical nestedness relation

$$\mathcal{X}^{(0)} \subset \mathcal{X}^{(1)} \subset \mathcal{X}^{(2)} \subset \ldots \subset \mathcal{X}^{(m)} ; \tag{4}$$

in other words, the spaces $\mathcal{X}^{(j)}$ are part of a nesting hierarchy. Is the assumption that it is possible to construct the hierarchy (4) realistic? In many practically important cases (4) follows trivially, e.g., uniform bisection of line segments and quadrisection of triangles. If the basis function pieces over finite elements are constructed through the master element in the parametric space which is then mapped to the physical space, the natural choice for refinement is uniform division in the parametric space. For instance, consider the quadratic 8-noded quadrilateral element in Figure 2 (isoparametric mappings are indicated by (I), and refinement is indicated by (R)).

Notice that the *interiors* of the finite elements on level j may be triangulated arbitrarily on level $j+1$, but the $(n-1)$-dimensional *faces* (edges in two dimensions, faces in three dimensions, etc.) need to be triangulated compatibly. In other words, if two n-dimensional finite elements on level j share an $(n-1)$-dimensional face, its triangulation on level $j+1$ has to be shared by the n-dimensional finite elements on level $j+1$.

The one-dimensional example with linear basis functions discussed above is illustrated in Figure 1. On the left side of the figure, the refinement is uniform (each element is divided), and stationary (each level is divided in the same way). However, that is not the only way the hierarchy could be constructed. As shown on the right side, the refinement may be neither uniform nor stationary. In general, uniform and stationary division will lead to the simplest algorithms, but it is not required in general. There may be advantages to applying non-stationary division, for instance when performing several levels of anisotropic refinement in a limited subregion (say, to resolve a boundary layer), followed by isotropic refinement. Different applications will require different division strategies.

1.1. Refinement equation

The hierarchical nestedness relation (4) opens the door to our refinement strategy: since $\mathcal{X}^{(j)} \subset \mathcal{X}^{(j+1)}$, any basis function $\phi_i^{(j)}(x)$ on level j may be exactly resolved in the finer basis $\{\phi_i^{(j+1)}(x)\}$:

$$\phi_i^{(j)}(x) = \sum_k a_{ik}^{(j+1)} \phi_k^{(j+1)}(x) , \tag{5}$$

where $a_{ik}^{(j+1)}$ are the coefficients of the linear combination. We shall *assume* that the sum in (5) has a finite number of terms with non-zero coefficients $a_{ik}^{(j+1)}$. It is possible to construct functions such that the sum is infinite, but we do not consider these rather exotic functions here.

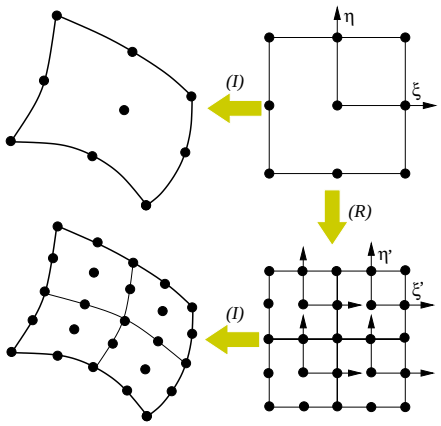

Figure 2. Refinement of quadratic quadrilaterals in the parametric space.

In many practical applications, a small set of finer basis functions on level $j' > j$ is sufficient to reconstruct the coarser basis function; this ***refinement set*** of the coarser function $\phi_i^{(j)}$ on level j' is denoted by $\mathcal{C}^{(j')}[\phi_i^{(j)}]$, and consists of those basis functions that contribute to the right-hand side of the refinement equation with a non-zero coefficient:

$$\mathcal{C}^{(j')}[\phi_i^{(j)}] = \left\{ \phi_k^{(j')} \mid a_{ik}^{(j')} \neq 0 \right\} \ . \tag{6}$$

Note that recursive application of (5) yields a more general formulation: any function $\phi_i^{(j)}(x)$ may be written in terms of a finite number of functions from various finer levels.

We now introduce some terminology that will aid us in the next section. If $\phi_k^{(j')}$ belongs to $\mathcal{C}^{(j')}[\phi_i^{(j)}]$, we say that "$\phi_i^{(j)}$ *is a* ***parent*** *of* $\phi_k^{(j')}$," and "$\phi_k^{(j')}$ *is a* ***child*** *of* $\phi_i^{(j)}$." Below we will see that refinement of our approximation space locally around a parent may create a new space in which the parent has been replaced by its children. Note that a function may have multiple parents as well as multiple children.

Equation (5) is an instance of the well-known ***refinement equation*** (also known under the name of dilation or multiresolution equation) [18]. The refinement equation is the key to our adaptivity approach: Given an approximation space $\mathcal{X}^{(j)}$ with basis $\mathcal{B}^{(j)}$, one can derive a custom space $\mathcal{X}$, with basis $\mathcal{B}$ whose resolution is locally finer, by replacing one or more parent basis functions from $\mathcal{B}^{(j)}$ by children functions from $\mathcal{B}^{(j')}$ on levels $j' > j$. In this sense, the refinement equation may be viewed as two statements: Firstly, the *approximation properties* of the basis function set $\mathcal{B}$ preserve and enhance those of $\mathcal{B}^{(j)}$ if the children basis functions are substituted for the parent basis function. Secondly, the *continuity* of the children functions is greater than or equal to the continuity of the parent function (since any finite linear combination of C^k functions is C^k). To put this into context, recall the discussion in the Introduction: existing mesh refinement techniques have to constantly wrestle with continuity because the refinement operations are applied to isolated *pieces* of basis functions.

1.2. Construction of adaptive spaces

1.2.1. Two-level refinement How should we adapt an approximation space? We choose to interpret the refinement equation (5) as "the left-hand side (the coarser parent function) is equivalent to the right-hand side (combination of functions from its refinement set)." The result will have at least the approximation properties of the coarser function, and since the children functions have narrower supports, the spatial resolution will be enhanced.

For the sake of the argument, let us consider first the case where a single, arbitrary function $\phi_i^{(j)}$ on level j is replaced by a linear combination of functions on level $j' = j + 1$ as given in Eq. (5). The adapted basis of space $\mathcal{X}$ may be written as

$$\mathcal{B} = \left(\mathcal{B}^{(j)} \backslash \phi_i^{(j)}\right) \bigcup \mathcal{C}^{(j+1)}[\phi_i^{(j)}] \ . \tag{7}$$

The basis set $\mathcal{B}$ consists of functions from two sets, $\mathcal{B}^{(j)}$ and $\mathcal{B}^{(j+1)}$. Since we have replaced function $\phi_i^{(j)}$ exactly, we obtain $\mathcal{X}^{(j)} \subset \mathcal{X} \subset \mathcal{X}^{(j+1)}$. Briefly, we have produced a "richer" set from $\mathcal{B}^{(j)}$ by replacing a given function by an equivalent set of functions with finer spatial resolution. We could also view this construction as activation (or deactivation) of selected functions from $\mathcal{B}^{(j+1)}$ (or $\mathcal{B}^{(j)}$).

We shall use the term ***active function*** for functions selected from a given basis set, and the symbol $\widehat{\mathcal{B}}^{(k)}$ will be used for the set of active functions from basis set $\mathcal{B}^{(k)}$. With this notation, Eq. (7) may be rewritten as

$$\mathcal{B} = \widehat{\mathcal{B}}^{(j)} \bigcup \widehat{\mathcal{B}}^{(j+1)} \ , \tag{8}$$

where

$$\widehat{\mathcal{B}}^{(j)} = \mathcal{B}^{(j)} \backslash \phi_i^{(j)} \qquad \text{and} \qquad \widehat{\mathcal{B}}^{(j+1)} = \mathcal{C}^{(j+1)}[\phi_i^{(j)}] \ .$$

Of course, the above argument is readily extended for the replacement of multiple functions from $\mathcal{B}^{(j)}$. Note that in many applications we require that $\mathcal{B}$ consists of *linearly independent* functions. In these applications we must pay special attention every time we activate a function; a detailed discussion of this issue follows in Section 2.

We choose to name the basis function set constructed in the above fashion the ***quasi-hierarchical*** basis, because it assumes a hierarchical character in those parts of the domain where functions from two or more levels are active at the same point. This feature is absent where only a single level of functions is activated, in which case an ordinary finite element approximation is recovered; hence the prefix "quasi."

1.2.2. Multiple-level refinement At this point we are ready to generalize the algorithm to an arbitrary number of refinement levels and arbitrarily complex selections from the levels. The basis $\mathcal{B}$ of a refined space $\mathcal{X}$ may be defined as

$$\mathcal{B} = \bigcup_{j=0}^{\infty} \widehat{\mathcal{B}}^{(j)} \ . \tag{9}$$

Recall that the sets $\widehat{\mathcal{B}}^{(j)}$ consist of functions activated on level j. If no functions are activated on level j, $\widehat{\mathcal{B}}^{(j)}$ is an empty set. For the moment we assume that the selections $\widehat{\mathcal{B}}^{(j)}$ are such that $\mathcal{B}$ consists of linearly independent functions; we shall discuss an algorithm that ensures this below.

1.2.3. Construction of a hierarchical basis Now let us consider an alternative construction. Instead of completely replacing the coarse function $\phi_i^{(j)}$ by the linear combination of the finer functions $\sum_k a_{ik}^{(j+1)} \phi_k^{(j+1)}$, we combine the coarse function with a *subset* of the finer functions. For definiteness, assume function $\phi_p^{(j+1)}$, with $a_{ip}^{(j+1)} \neq 0$, is omitted from the sum of the finer functions, and the basis function set obtained in this way is

$$\mathcal{B} = \mathcal{B}^{(j)} \bigcup \left(\mathcal{C}^{(j+1)}[\phi_i^{(j)}] \backslash \phi_p^{(j+1)}\right) \ . \tag{10}$$

The result is a *hierarchical* adapted basis set $\mathcal{B}$. Again, the conditions for $\mathcal{B}$ to consist of linearly independent functions are discussed below.

Note that the difference between the sets of (7) and (10) is only in which functions are active on levels j and $j + 1$. Therefore, we may write Eq. (10) as

$$\mathcal{B} = \widehat{\mathcal{B}}^{(j)} \bigcup \widehat{\mathcal{B}}^{(j+1)} \ , \tag{11}$$

with sets $\widehat{\mathcal{B}}^{(j)}$ and $\widehat{\mathcal{B}}^{(j+1)}$ defined as

$$\widehat{\mathcal{B}}^{(j)} = \mathcal{B}^{(j)} \qquad \text{and} \qquad \widehat{\mathcal{B}}^{(j+1)} = \left(\mathcal{C}^{(j+1)}[\phi_i^{(j)}] \backslash \phi_p^{(j+1)} \right) \ .$$

We choose to use the term ***hierarchical*** for the adapted basis constructed in (11) because the coarse level functions are not deleted (replaced), and the added functions on finer levels therefore represent finer *details* added onto coarser approximation scales.

As an illustration, consider Figure 3. Basis function $\phi_j^{(0)}$ on level zero is being refined by the two functions $\phi_{j-1}^{(1)}$ and $\phi_{j+1}^{(1)}$ on level one. Note that this is the classical "dyadic" hierarchical refinement [21]. It is certainly not the only choice: Out of the three functions $\phi_{j-1}^{(1)}$, $\phi_j^{(1)}$, and $\phi_{j+1}^{(1)}$ that may be used to *replace* function $\phi_j^{(0)}$ we could have chosen any *single* function, or any *pair* of functions to *refine* it. The two functions selected in Figure 3 are just one possible choice. For comparison, Figure 4 also shows how the refinement would proceed using our quasi-hierarchical method. The function $\phi_j^{(0)}$ would be replaced by all three functions $\phi_{j-1}^{(1)}$, $\phi_j^{(1)}$, and $\phi_{j+1}^{(1)}$.

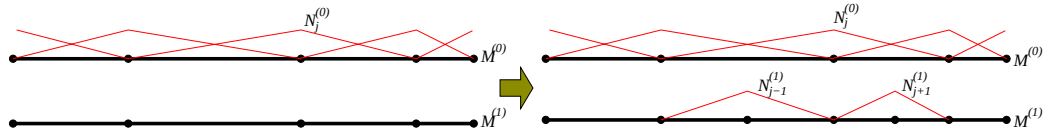

Figure 3. Example of true hierarchical refinement.

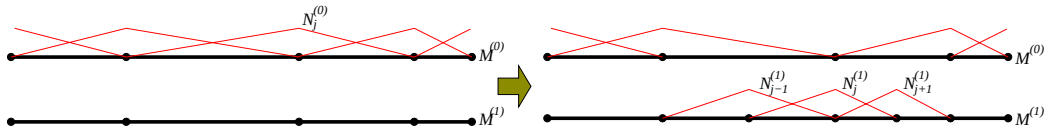

Figure 4. Example of quasi-hierarchical refinement.

To summarize: The representation (9) of the quasi-hierarchical basis may also be used for the hierarchical basis function set. The differences are limited only to the construction of the active sets $\widehat{\mathcal{B}}^{(j)}$: using the quasi-hierarchical route we replace coarse-level functions, while following the true hierarchical route we add fine-level details on top of coarser functions. This argument is illustrated further in Figure 5 where the quasi-hierarchical and the true hierarchical refined bases constructed from the mesh hierarchy of Figure 1 (on the left) are compared. Note that the spans of the two basis functions sets are identical. The hat functions represent the active functions; the inactive functions are not shown.

2. Refinement and unrefinement

For many applications, it is imperative that the active functions constitute a *basis*, i.e. they have to be linearly independent. We call this requirement the ***linear independence requirement***. In this section we describe algorithms which activate and deactivate functions from the nesting hierarchy and guarantee the linear independence of the active function set. Many alternative strategies seem possible; here we outline two: *quasi-hierarchical refinement* and *hierarchical refinement.*

Let us delimit, by a set of assumptions, the types of finite element approximations that we wish to consider. Our focus in this section is not general (e.g., non-interpolating spline approximations

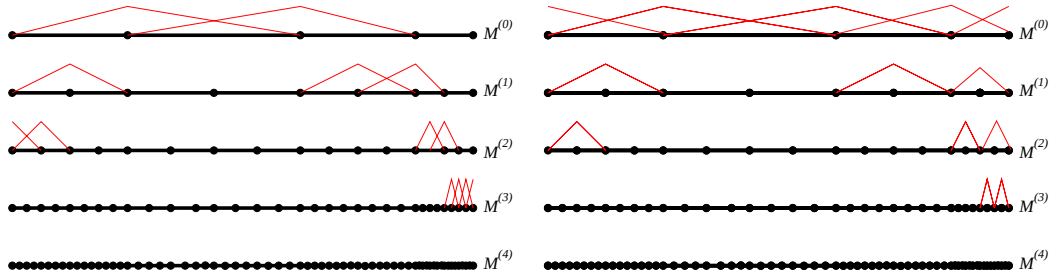

Figure 5. The two refinement strategies, quasi-hierarchical and hierarchical, compared side by side.

are excluded from consideration), instead we choose an important, practical set of approximations, which are amenable to simple refinement/unrefinement algorithms. In particular, we consider only approximations which satisfy these assumptions:

1. On each level j of the nesting hierarchy, the basis functions $\phi_i^{(j)}$ verify the Kronecker delta property, i.e.,
$$\phi_i^{(j)}(x_k) = \delta_{ik} \ , \tag{12}$$
where x_k is the location of the node associated with function $\phi_k^{(j)}$. In other words, we assume that an approximation built of basis functions from any single level *interpolates* the nodal values. This assumption is not very restrictive for traditional finite element approximations. Indeed, they are as a rule constructed in this way: consider classical FE bases constructed on meshes of triangular, quadrilateral, tetrahedral, or hexahedral cells. This assumption also accommodates approximations using interpolating subdivision basis functions; however, for an important class of approximations, including non-interpolating splines and also non-interpolating subdivision surfaces, this assumption must be lifted [11, 22]. The application of the present refinement methodology to the modeling of thin-shells by the subdivision element method has been discussed in Reference [6].
2. Vertices, edges, (all topological entities with smaller dimension than the domain) on level j are covered with corresponding entities on level $j+1$. Consequently, a cell on level j is a disjoint union of cells on level $j+1$.

The eager reader may wish to explore a corollary to these assumptions: no basis function support is entirely enclosed by the support of another basis function, i.e.,

$$\text{supp}\left[\phi_k^{(j)}\right] \not\subseteq \text{supp}\left[\phi_i^{(j)}\right] \quad \text{for} \quad k \neq i \text{ and } \forall j \geq 0 \ . \tag{13}$$

Consider two functions on the same level; consequent to the corollary, the refinement set of either must contain a function not present in the refinement set of the other.

Now that we have described the set of approximations on which we will focus, we are nearly ready to describe algorithms for refining/unrefining these approximations. Since there are many possible algorithms for building adapted bases, we simplify our algorithmic design by adhering to three **rules**:

1. The refining/unrefining of a function on level j may activate or deactivate that function or any of its children on level $j+1$; no other function may be affected.
2. A function on level $j+1$ ($j+1 \geq 1$) may be refined only when all its parents on level j have been refined.
3. A function on level j may be unrefined only if (i) it was previously refined and (ii) all its children on level $j+1$ are not refined.

The reader may note that if the basis functions are completely supported by one ring of elements around a node, then rules 2 and 3 enforce the common rule of one-level-difference refinement of neighbors; commonly referred to as the "restriction criterion," this rule has been applied to finite element meshes, as well as to quadtrees and octrees in various contexts (graphics, mesh generation, spatial searches, etc.).

We now describe two approaches to atomic operations that produce adaptive basis function sets. We show that if our (un)refinement operation is applied in an atomic fashion (i.e., either it is entirely executed or not at all) then the linear independence requirement is preserved. Furthermore, our algorithms ensure that the refinement step is *lossless*; the span of the resulting set includes the span of the original set. Lossless refinement means that the refined basis allows for any function in the original basis to be reproduced exactly. In contrast, unrefinement cannot be lossless: some information is always going to be lost, since the goal of unrefinement is to decrease the span of the approximation space.

2.1. Quasi-hierarchical basis

Let us begin by exploring the quasi-hierarchical refinement strategy. We first describe the refinement operation, and show that it preserves the linear independence requirement and is lossless; we then describe the unrefinement operation, and show that it too preserves the linear independence requirement.

2.1.1. Refinement Given an initial basis function set, $\mathcal{B}$, which contains $\phi_i^{(j)}$, and satisfies the linear independence requirement, we choose to produce another function set $\mathcal{B}$' by deactivating $\phi_i^{(j)}$ and activating all of its children $\mathcal{C}^{(j+1)}[\phi_i^{(j)}]$. We refer to this algorithm, which maps $\mathcal{B}$ to $\mathcal{B}$', as *quasi-hierarchical refinement.*

Proposition: We claim that quasi-hierarchical refinement (i) preserves the linear independence requirement and (ii) is lossless.

Proof (i): Assumption 2 guarantees that there is some x_i^{j+1} covering x_i^j. Further, assumption 1 guarantees that on level j only $\phi_i^{(j)}$ is non-zero at x_i^j, and on level $j+1$ only $\phi_i^{(j+1)}$ is non-zero at x_i^{j+1}; consequently $\phi_i^{(j+1)}$ belongs *only* to $\mathcal{C}^{(j+1)}[\phi_i^{(j)}]$. $\phi_i^{(j+1)}$, a "private child" of $\phi_i^{(j)}$, may be introduced into the adapted basis function set *only* through the refinement of $\phi_i^{(j)}$. Therefore, activating functions in $\mathcal{C}^{(j+1)}[\phi_i^{(j)}]$ cannot produce a complete refinement set of another function on level j, for the other function too has a private child not present in $\mathcal{C}^{(j+1)}[\phi_i^{(j)}]$. It also cannot produce a redundant representation of $\phi_i^{(j)}$, since this coarser function is deactivated. Furthermore, by force of rule 2, none of the functions $\phi_k^{(j+1)}$ from the refinement set $\mathcal{C}^{(j+1)}[\phi_i^{(j)}]$ is currently present through its complete refinement set, $\mathcal{C}^{(j+2)}[\phi_k^{(j+1)}]$; therefore, activation of any function from $\mathcal{C}^{(j+1)}[\phi_i^{(j)}]$ cannot introduce linear dependencies with functions on levels finer than $j+1$.

Proof (ii): since we have activated all members of $\mathcal{C}^{(j+1)}[\phi_i^{(j)}]$, then (5) and (6) guarantee that $\mathcal{B}$' still spans $\phi_i^{(j)}$, the only function removed from $\mathcal{B}$.

2.1.2. Unrefinement Given an initial basis function set, $\mathcal{B}$, that satisfies the linear independence requirement, and given a previously refined $\phi_i^{(j)}$, not in $\mathcal{B}$ and eligible (under rule 3) for unrefinement, we choose to produce another function set $\mathcal{B}$' by activating $\phi_i^{(j)}$ and deactivating those children of $\phi_i^{(j)}$ which have no other currently refined (i.e., inactive) parent. Expressed mathematically, the members

of the following set are deactivated:

$$\left\{ \phi_m^{(j+1)} : \overbrace{\phi_m^{(j+1)} \in \mathcal{C}^{(j+1)}[\phi_i^{(j)}]}^{\substack{\phi_m^{(j+1)} \text{ is in} \\ \text{refinement set of } \phi_i^{(j)}}} \wedge \overbrace{\forall\, r \neq i \;\; \phi_m^{(j+1)} \in \mathcal{C}^{(j+1)}[\phi_r^{(j)}] \rightarrow \phi_r^{(j)} \in \widehat{\mathcal{B}}^{(j)}}^{\substack{\text{it may } \textit{only} \text{ be in refinement sets} \\ \text{of active (non-refined) nodes on level } j}} \right\} . \tag{14}$$

We refer to this algorithm, which maps $\mathcal{B}$ to $\mathcal{B}$', as *quasi-hierarchical unrefinement.*
Proposition: We claim that quasi-hierarchical unrefinement preserves the linear independence requirement.
Proof: By rule 3, function $\phi_i^{(j+1)}$ is not present in $\mathcal{B}$ through its refinement set, $\mathcal{C}^{(j+2)}[\phi_i^{(j+1)}]$. It follows from assumption 1 that $\phi_i^{(j+1)}$ may be introduced only through the refinement of $\phi_i^{(j)}$. Therefore, $\phi_i^{(j+1)}$ is certainly member of set (14), and since at least $\phi_i^{(j+1)}$ is deactivated, the refinement set of $\phi_i^{(j)}$ is guaranteed not to be complete at the end of the unrefinement step. Hence, activating $\phi_i^{(j)}$ cannot introduce a linear dependency: the linear independence requirement is preserved.

2.2. Hierarchical basis

We have completed our overview of quasi-hierarchical refinement, which treats refinement as the *replacement* of coarse-level functions by finer-level functions (Figure 4). Let us turn to an alternative strategy for constructing adapted bases: hierarchical refinement treats refinement as the *addition* of finer-level "detail functions" to an unchanged set of coarse-level functions (Figure 3). Before we continue, let us formalize the concepts of a detail function and a detail (function) set.
Definition: Given a function $\phi_i^{(j)}$, construct the set of all functions $\phi_k^{(j+1)} \in \mathcal{C}^{(j+1)}[\phi_i^{(j)}]$ such that they vanish at the location x_i of node i

$$\mathcal{G}^{(j+1)}[\phi_i^{(j)}] = \left\{ \phi_k^{(j+1)} \mid \phi_k^{(j+1)} \in \mathcal{C}^{(j+1)}[\phi_i^{(j)}] \text{ and } \phi_k^{(j+1)}(x_i) = 0 \right\} . \tag{15}$$

The set $\mathcal{G}^{(j+1)}[\phi_i^{(j)}]$ is the ***detail set*** of $\phi_i^{(j)}$. Functions that belong to at least one detail set are called ***detail functions***.

2.2.1. Refinement Given an initial basis function set, $\mathcal{B}$, which contains $\phi_i^{(j)}$, and satisfies the linear independence requirement, we choose to produce a refined set $\mathcal{B}$' by activating $\mathcal{G}^{(j+1)}[\phi_i^{(j)}]$. We refer to this algorithm, which maps $\mathcal{B}$ to $\mathcal{B}$', as *hierarchical refinement.*
Proposition: We claim that hierarchical refinement (i) preserves the linear independence requirement and (ii) is lossless.
Proof (i): The detail set of function $\phi_i^{(j)}$ vanishes at x_i by definition. Furthermore, by assumption 1, the detail set of any other function $\phi_m^{(j)}$, $m \neq i$, vanishes at x_i. Consequently, activating an arbitrary number of detail functions on level $j+1$ cannot produce the complete refinement set of $\phi_i^{(j)}$ (there has to be at least one function that is non-zero at x_i, and that function is not a member of any detail set). Therefore it may be concluded that activating detail functions preserves the invariant.
Proof (ii): we do not remove functions from $\mathcal{B}$ during refinement; its span cannot shrink.

2.2.2. Unrefinement Given an initial basis function set, $\mathcal{B}$, that satisfies the linear independence requirement, and given a previously refined $\phi_i^{(j)}$, also in $\mathcal{B}$ and eligible for unrefinement (rule 3), we choose to produce another function set $\mathcal{B}$' by deactivating functions $\phi_m^{(j+1)} \in \mathcal{G}^{(j+1)}[\phi_i^{(j)}]$ that are absent from all the refinement sets of currently refined functions on level j. We refer to this algorithm, which maps $\mathcal{B}$ to $\mathcal{B}$', as *hierarchical unrefinement.*

Proposition: We claim that hierarchical unrefinement preserves the linear independence requirement.
Proof: Unrefinement does not involve the activation of any function, hence a linear dependence cannot be introduced.

3. Examples

For the sake of brevity, we will refer to our approach as CHARMS in this section. The acronym stands for "Conforming Hierarchical Adaptive Refinement Methods."

The grids in this section have been used in Galerkin finite element solutions of the partial differential equation of linear diffusion. Unless stated otherwise, the grids are displayed as a collection of "integration cells". Those are the "smallest" finite elements that support an active function at a given location in the domain. The integration cells are being used to evaluate the weak form terms (note that they tile the domain).

CHARMS has been applied to the implementation of finite element refinement in an experimental computer code. The code has been designed and debugged for one-dimensional adaptive approximations. The power of CHARMS became apparent during the next step, two-dimensional mesh refinement for quadrilateral finite elements (the geometric refinement is performed by quadrisection in the bi-unit master coordinates). The implementation took the first author little less than three hours! Figure 6 shows the adapted grid; Figure 7 illustrates the hierarchies of finite elements: the quasi-hierarchical basis on the left, the true hierarchical basis on the right. The balls at certain nodes indicate the presence of an active function associated with that node. Notice that the active functions vanish on the interior boundary of the patches of finite elements that support them. That is a consequence of our use of the refinement equation (and a visual explanation of why the compatibility is a non-issue with CHARMS).

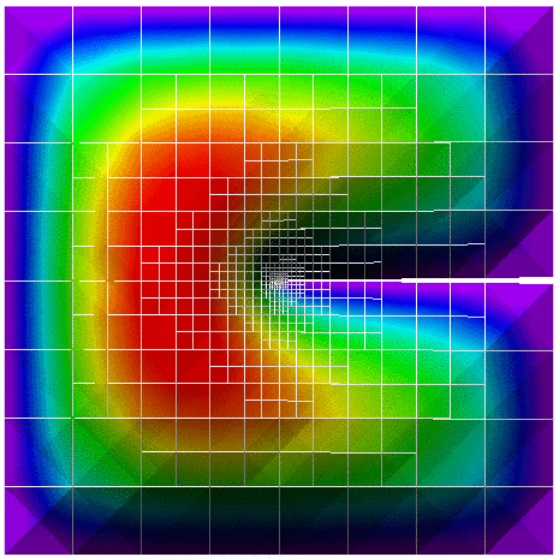

Figure 6. Adapted grid on a square domain with "crack", composed of quadrilateral finite elements. Integration cells are shown by white edges.

CHARMS has been subsequently exercised in the implementation of adaptive refinement on 8-node hexahedral meshes (octa-section in the master coordinates). Because the hexahedron element

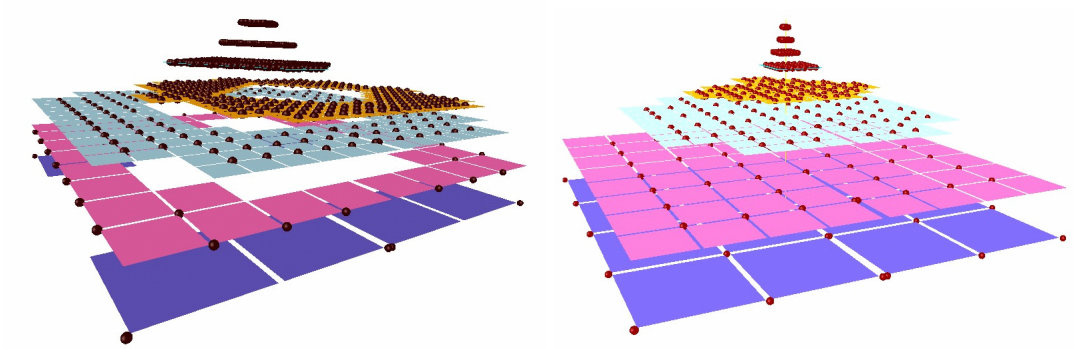

Figure 7. Finite element hierarchy for the problem of Figure 6.

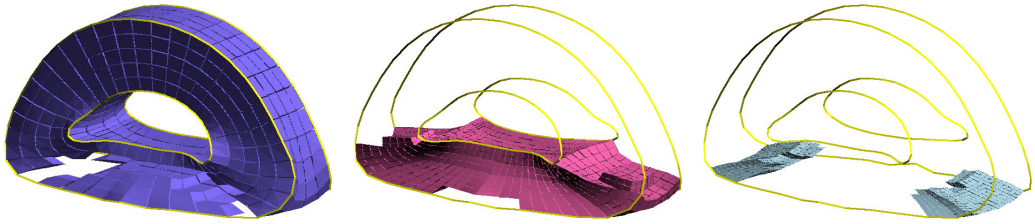

Figure 8. Hierarchy of finite element grids for a quasi-hierarchical basis.

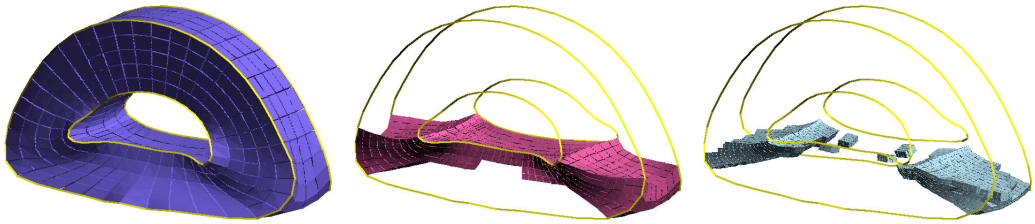

Figure 9. Hierarchy of finite element grids for a true hierarchical basis.

represents a direct extension of the bilinear quadrilateral to three dimensions, the implementation was now even easier, and was completed in a couple of hours. Importantly, no special tricks were required when going from two to three dimensions. Figures 8 (quasi-hierarchical basis) and 9 (true hierarchical basis) display the hierarchy of the finite elements that support active functions at one of three refinement levels.

Figure 10 shows an adapted grid of the human brain tissue composed of linear-precision tetrahedra with four refinement levels. The tetrahedron is a more complex element type to refine because of its space-tiling properties. We have used the Kuhn triangulation of the cube which guarantees constant tetrahedron shape quality upon octa-section for any number of refinement levels [12]. As expected, the implementation of the refinement proper presented no difficulties whatsoever.

It is not surprising that CHARMS is also easily applied to meshes composed of quadratic-precision

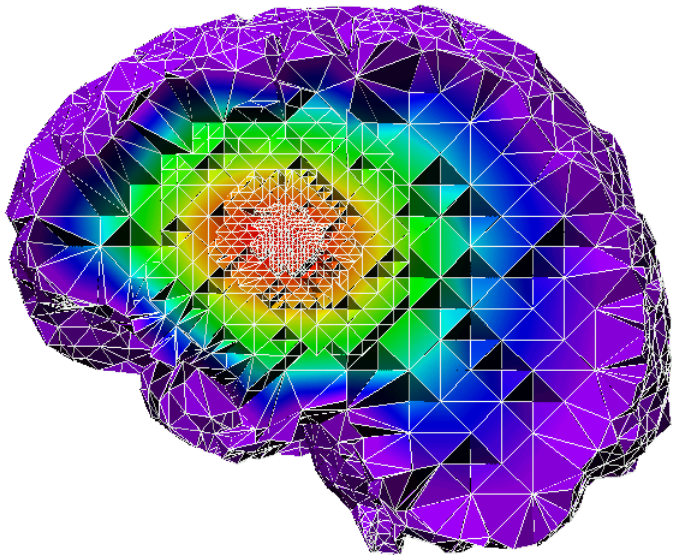

Figure 10. Cut through the grid of human brain tissue adapted to a point source.

finite elements. As an example, Figure 11 shows an adapted solution to a source/sink problem on the square obtained with 6-node triangles. The triangles are isoparametric, and the geometric division (quadrisection) is applied in the parametric space ξ, η. Because of the higher number of interacting nodes, the refinement connectivity is more tedious to program than for the linear-precision elements, but otherwise no special treatment is required.

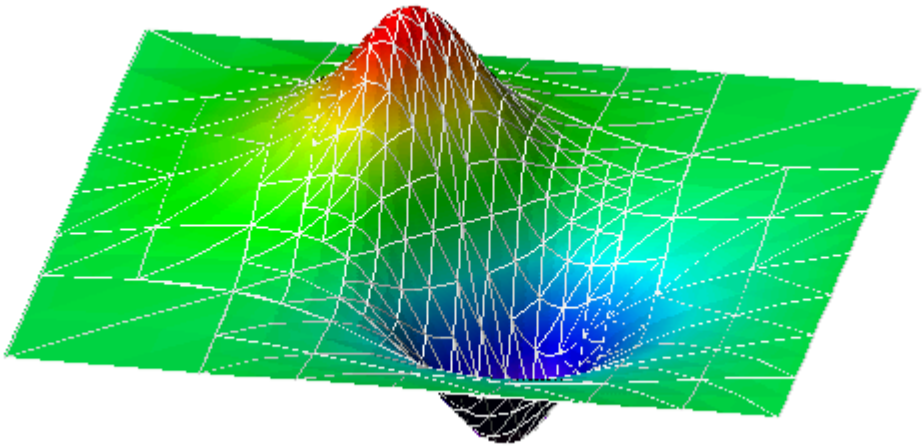

Figure 11. Solution to a two-dimensional diffusion equation with a source/sink pair obtained with quadratic triangles.

Conclusions

We have presented a general mesh refinement approach. Its beauty is its extreme simplicity. Our only assumption is that it is possible to construct an infinite hierarchy of meshes of increasing spatial resolution such that a given function at an arbitrary level may be replaced exactly by a linear combination of functions on some finer level. Said differently, we propose to interpret the well-known refinement equation as a tool for refinement and unrefinement. We show that we can construct a refined set of basis functions by selectively deactivating coarse functions, replacing them with finer functions which become active in the process. We choose to label this refined set "quasi-hierarchical", because it assumes a hierarchical character in those regions of the mesh where two or more levels of functions are active at the same time. Elsewhere in the domain there is no difference between the character of a refined basis function set, and a single-level finite element basis. We also present an alternative refinement strategy: Not all the finer functions are activated, and the coarse function is kept instead of being replaced by the finer functions. The resulting approximation spaces are hierarchical in the classical sense: the finer functions are associated with "details," while the coarser functions define the "global" variation. The constructions of the two refined basis function sets, quasi-hierarchical and true hierarchical, are completely equivalent in the sense that both proceed by activating and deactivating certain functions from the virtual hierarchy of nested approximation spaces. Therefore, it is also perfectly feasible to mix these two strategies in the construction of the approximation basis: parts of the domain may be covered by a true hierarchical basis, parts by the quasi-hierarchical basis.

It is well known that state-of-the-art mesh refinement based on isolated element splitting is trivial in one dimension, and becomes much harder in multi-dimensional settings. However, note that nowhere we had to worry about preserving the compatibility of the refined basis. As a consequence of our use of the refinement equation, the resulting refined basis is conforming by construction. This removes one of the major headaches that accompanies traditional mesh refinement approaches, and makes it much easier (or trivial) to preserve or enhance the shape quality of the finite elements across all refinement levels.

Moreover, the refinement equation holds without any mention of the number of dimensions of the ambient space, or of the order of the basis functions. Therefore, it is equally easy to apply to piecewise linear approximation on the line as to trilinear approximation on hexahedral meshes, or piecewise cubic tensor-product approximation in three dimensions. In fact, we show in Reference [6] that the present technique renders adaptivity for subdivision surfaces not only feasible, but easy.

Note that our formulation may under certain conditions yield approximation basis function sets equivalent in terms of their span to those generated with other approaches, for instance when degrees of freedom associated with hanging nodes are eliminated via constraints. Indeed, we do not claim to have a method for constructing better approximations in those cases. Rather, our claim is that we formulate an alternative algorithm for constructing adaptive approximation spaces, which in some cases may be equivalent to those resulting from other methods. One of the most appealing characteristics of our method is that it makes it so much easier to formulate methods whose implementations have traditionally been very tedious and error-prone.

At the same time, our method *is* more general than existing approaches, and provides a consistent and robust path towards formulations of adaptive approximations where none have been available so far: a practically important example are Loop subdivision surfaces [11].

The finite elements at finer levels are nested within finite elements of coarser levels. Therefore, the road towards the exploration of multilevel solvers (multigrid) is open. For some subdivision schemes, such as the $\sqrt{3}$-subdivision [7], nestedness is not available, but CHARMS are still applicable, and hierarchical preconditioning and multigrid apply as well.

Finally, our approach is linked to wavelets through the refinement equation, which is one of the pillars of wavelet theory; this is a clear avenue to explore.

Acknowledgments

The research reported here was supported in part by NSF (DMS-9874082, ACI-9721349, DMS-9872890, ACI-9982273) with additional support from the Packard Foundation, Microsoft, Alias|Wavefront, Pixar, and Intel.

REFERENCES

1. D. N. Arnold, A. Mukherjee, and L. Pouly. Locally adapted tetrahedra meshes using bisection. *SIAM Journal on Scientific Computing*, 22(2):431–448, 2001.
2. R.E. Bank, T.F. Dupont, and H. Yserentant. The hierarchical basis multigrid method. *Numerische Mathematik*, 52(4):427–458, 1988.
3. R.E. Bank and J.C. Xu. An algorithm for coarsening unstructured meshes. *Numerische Mathematik*, 73(1):1–36, 1996.
4. Graham F. Carey. *Computational Grids: Generation, Adaptation and Solution Strategies*. Taylor & Francis, 1997.
5. F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072, 2000.
6. E. Grinspun, P. Krysl, and P. Schröder. CHARMS: A simple framework for adaptive simulation. *Proceedings of ACM SIGGRAPH 2002*, 2002.
7. Leif Kobbelt. $\sqrt{3}$ subdivision. *Proceedings of SIGGRAPH*, pages 103–112, 2000.
8. H. P. Langtangen. *Computational Partial Differential Equations*. Springer Verlag, 1999.
9. A.W. Liu and B. Joe. Quality local refinement of tetrahedral meshes based on bisection. *SIAM Journal on Scientific Computing*, 16(6):1269–1291, 1995.
10. A.W. Liu and B. Joe. Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. *MATHEMATICS OF COMPUTATION*, 65(215):1183–1200, 1996.
11. C. T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, 1987.
12. M.E.G. Ong. Uniform refinement of a tetrahedron. *SIAM Journal on Scientific Computing*, 15(5):1134–1144, 1994.
13. A. Plaza and G.F. Carey. Local refinement of simplicial grids based on the skeleton. *Applied Numerical Mathematics*, 32(2):195–218, 2000.
14. A. Plaza, M.A. Padron, and G.F. Carey. A 3d refinement/derefinement algorithm for solving evolution problems. *Applied Numerical Mathematics*, 32(4):401–418, 2000.
15. M.C. Rivara. New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations. *International Journal for Numerical Methods in Engineering*, 40(18):3313–3324, 1997.
16. M.C. Rivara and P. Inostroza. Using longest-side bisection techniques for the automatic refinement of delaunay triangulations. *International Journal for Numerical Methods in Engineering*, 40(4):581–597, 1997.
17. M.C. Rivara and G. Iribarren. The 4-triangles longest-side partition of triangles and linear refinement algorithms. *Mathematics of Computation*, 65(216):1485–1502, 1996.
18. G. Strang. Wavelets and dilation equations: A brief introduction. *SIAM Review*, 31(4):614–627, 1989.
19. S.O. Wille. A structured tri-tree search method for generation of optimal unstructured finite-element grids in 2 and 3 dimensions. *International Journal for Numerical Methods in Fluids*, 14(7):861–881, 1992.
20. S.O. Wille. A tri-tree multigrid recoarsement algorithm for the finite element formulation of the navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 135(1-2):129–142, 1996.
21. H. Yserentant. Hierarchical bases. In *ICIAM91*. SIAM, 1991.
22. Denis Zorin and Peter Schröder, editors. *Subdivision for Modeling and Animation*. Course Notes. ACM SIGGRAPH, 1998.