CHARMS: A Simple Framework for Adaptive Simulation

Eitan Grinspun
Caltech

Petr Krysl
UCSD

Peter Schröder
Caltech

Abstract

Finite element solvers are a basic component of simulation applications; they are common in computer graphics, engineering, and medical simulations. Although *adaptive* solvers can be of great value in reducing the often high computational cost of simulations they are not employed broadly. Indeed, building adaptive solvers can be a daunting task especially for 3D finite elements. In this paper we are introducing a new approach to produce *conforming, hierarchical, adaptive refinement methods* (CHARMS). The basic principle of our approach is to refine basis functions, not elements. This removes a number of implementation headaches associated with other approaches and is a general technique independent of domain dimension (here 2D and 3D), element type (e.g., triangle, quad, tetrahedron, hexahedron), and basis function order (piecewise linear, higher order B-splines, Loop subdivision, etc.). The (un-)refinement algorithms are simple and require little in terms of data structure support. We demonstrate the versatility of our new approach through 2D and 3D examples, including medical applications and thin-shell animations.

CR Categories: G.1.8 [**Partial Differential Equations**]: Finite element methods, Multigrid and multilevel methods; G.1.2 [**Approximation**]: Wavelets, Spline and piecewise polynomial approximation, Linear approximation; I.3.5 [**Computational Geometry and Object Modeling**]: Physically based modeling, Splines; I.3.7 [**Three-Dimensional Graphics and Realism**]: Animation; J.2 [**PHYSICAL SCIENCES AND ENGINEERING**]: Engineering

General Terms: Algorithms, Design, Experimentation, Performance

Additional Keywords: Adaptive Computation, Refinement Relation, Basis Function, Subdivision, Multiresolution

1 Introduction

Many applications of computer graphics require the modeling of physical phenomena with high visual or numerical accuracy. Examples include the simulation of cloth [House and Breen 2000], water [Foster and Fedkiw 2001], human tissue [Wu et al. 2001] and engineering artifacts [Kagan and Fischer 2000], among many others. Typically the underlying formulations require the solution of partial differential equations (PDEs). Such equations are also at the base of many geometric modeling [Celniker and Gossard 1991] and optimization problems [Lee et al. 1997]. Most often the continuous equations are discretized with the finite element (FE) or finite difference (FD) method before a (non-)linear solver can be used to compute an approximate solution to the original problem. For example, Terzopoulos and coworkers described methods to model many physical effects for purposes of realistic animation [1987]. Their discretization was mostly based on simple, uniform FD approximations. Later Metaxas and Terzopoulos employed FE methods since they are more robust, accurate, and come with more mathematical machinery [1992]. For this reason, human tissue simulations have long employed FE methods (e.g., [Gourret et al. 1989; Azar et al. 2001] and references therein).

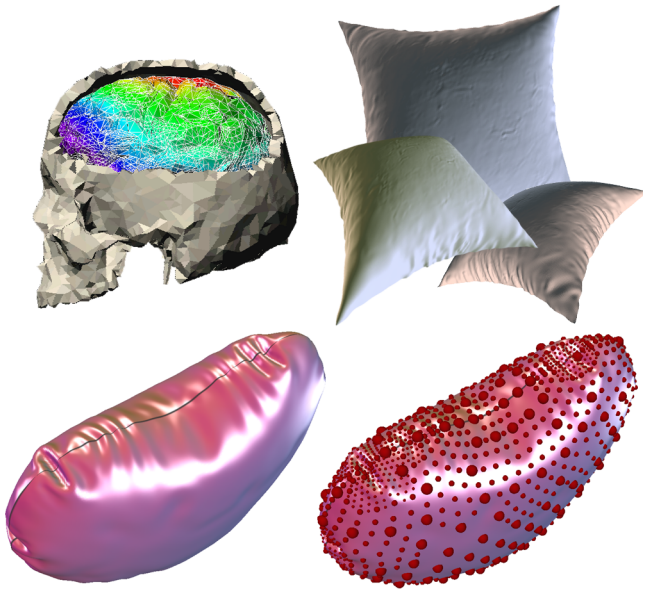

Figure 1: *Examples run within the CHARMS framework demonstrating a variety of simulations which benefit from our general adaptive solver framework: surgery simulation, modelling cloth pillows, inflating a metal-foil balloon. For details see Section 4.*

For better performance it is highly desirable to construct *adaptive* discretizations, allocating resources where they can be most profitably used. Building such adaptive discretizations robustly is generally very difficult for FD methods and very little theoretical guidance exists. For FE methods many different approaches exist. They all rely on the basic principle that the resolution of the domain discretization ('mesh') should be adjusted based on local error estimators. For example, Debunne et al. superimposed tetrahedral meshes at different resolutions and used heuristic interpolation operators to transfer quantities between the disparate meshes as required by an error criterion [2001]. Empirically this worked well for real-time soft-body deformation, though there exists no mathematical analysis of the method. A strategy based on precomputed progressive meshes (PM) [Hoppe 1996] was used by Wu et al. [2001] for surface based FE simulations. Since the PM is constructed in a pre-process it is unclear how well it can help adapt to the online simulation. O'Brien and Hodgins followed a more traditional approach by splitting tetrahedra in their simulation of brittle fracture (mostly to accommodate propagating cracks) [1999]. Such *refinement* algorithms and their associated unrefinement operators have the advantage that they come with well established theory [Cohen et al. 2001] and result in nested meshes, and by implication nested approximation spaces. Since the latter is very useful for many multi-resolution techniques we have adopted refinement as our basic strategy.

Typical mesh refinement algorithms approach the problem of refinement as one of splitting *elements* in isolation. Unfortunately this leads to a lack of *compatibility* ("cracks"); to deal with this issue one may: (1) "snap" T-vertices to the neighboring edge; (2) use Lagrange multipliers or penalty methods to numerically enforce compatibility; or (3) split additional elements through insertion of conforming edges as in "red/green" triangulations or bisection algorithms (the technique used by O'Brien and Hodgins [1999], for example). Each one of these approaches works, but none is ideal. For example, penalty methods lead to stiff equations with their associated numerical problems, while red/green triangulations are very cumbersome to implement in 3D because of the many cases involved [Bey 2000]. As a result various different, specialized algorithms exist for different element types such as triangles [Bank and Xu 1996; Rivara and Inostroza 1997], tetrahedra [Arnold et al. 2001] and hexahedra [Langtangen 1999].

This lack of a general approach at times coupled with daunting implementation complexity (especially in 3D) has no doubt contributed to the fact that sophisticated adaptive solvers are not broadly used in computer graphics applications or general engineering design. The situation is further complicated by the need of many computer graphics applications for higher order ("smooth") elements. For example, [Celniker and Gossard 1991] used higher order FEs for surface modeling with physical forces and geometric constraints (see also [Halstead et al. 1993] and [Mandal et al. 1997] who used Catmull-Clark subdivision surfaces and [Terzopoulos and Qin 1994] who used NURBS). None of these employed adaptivity in their solvers. In fact, for basis functions such as B-splines or those induced by subdivision, elements cannot be refined individually without losing nestedness of approximation spaces. For example, Welch and Witkin, who used tensor product cubic B-splines as their constrained geometric modeling primitive, encountered this difficulty [Welch and Witkin 1992]. To enlarge their FE solution space they added finer level basis functions, reminiscent of hierarchical splines [Forsey and Bartels 1988], instead of refining individual elements. Later, Gortler and Cohen used cubic B-spline wavelets to selectively increase the solution space for their constrained variational sculpting environment [Gortler and Cohen 1995].

Contributions The use of hierarchical splines and wavelets in a FE solver framework are specialized instances of a general class of *conforming, hierarchical, adaptive refinement methods*, for short CHARMS, which we introduce here. Instead of refining elements, CHARMS are based on the refinement of basis functions. From an approximation theory point of view, this is a simple statement, but it has a number of very important and highly practical consequences. Our adaptive solver framework *requires only* that the basis functions are refinable. It makes *no assumptions* as to (1) the dimension of the domain; (2) the actual element types, be they triangles, quadrilaterals, tetrahedra, hexahedra, or more general domains; (3) the approximation order; and (4) the connectivity of the support of the basis functions. The approach is *always globally compatible* without requiring any particular enforcement of this fact. Consequently, all the usual implementation headaches associated with maintaining compatibility are entirely eliminated. What does need to be managed are tesselations of the overlap between basis functions, possibly living at very different levels of the refinement hierarchy. However, we will show that very short and simple algorithms, based on simple invariants, keep track of these interactions in a guaranteed fashion. While we were originally motivated by the need to find a refinement strategy for higher order basis functions, CHARMS has significant advantages even when only piecewise linear basis functions are used.

A single programmer implemented and debugged our basic CHARMS algorithm for 1D problems within a day (using an existing non-adaptive FE solver). Extending the implementation to 2D and 3D problems took another day. This forcefully attests to the simplicity and generality of the underlying framework. We demonstrate the versatility of our approach by applying it to several different FE simulations, involving both surface and volume settings (Fig. 1).

2 Motivation

In this section we consider a very simple example to elucidate the difference between finite element and basis function refinement, before we describe the general (un-)refinement algorithms in Section 3. To serve our exposition we will examine a *boundary value* problem. *Initial value* problems, which unfold in time from given initial conditions, are very common in animation applications, and are also accommodated by our framework.

2.1 Piecewise Linear Approximation in 1D

As a canonical example of a second order boundary value problem consider Laplace's equation with prescribed displacements at the boundary

$$-\Delta u(x) = 0\,, \quad u|_{\partial\Omega} = 1\,, \quad x \in \Omega \subset \mathbb{R}^d\,. \tag{1}$$

A FE method typically solves the weak form of this equation, selecting from the *trial space* the solution U which satisfies

$$a(U, v) = \int_\Omega \nabla U \cdot \nabla v \, dx = 0$$

for all v in some *test space*. We write the solution $U = g + u$ as a sum of the function g that satisfies the inhomogeneous essential boundary condition, and of the trial function u that satisfies the homogeneous boundary condition $u|_{\partial\Omega} = 0$. In the Galerkin method, which we adopt in this discussion, these test and trial spaces coincide.

Since the bilinear form $a(\cdot, \cdot)$ contains only first derivatives, we may approximate the solution using piecewise linear basis functions for both spaces. The domain is discretized into a disjoint union of elements of finite extent. Each such element has an associated linear function (e.g., Fig. 2 for $d = 1$). This results in a linear system

$$\mathbf{K}\mathbf{u} = \mathbf{b}$$

where the *stiffness matrix* entries k_{ij} describe the interaction of degrees of freedom (DOFs) at vertex i and j under the action of $a(\cdot, \cdot)$; the right hand side $\mathbf{b}$ incorporates the inhomogeneous boundary conditions; and $\mathbf{u}$ is the unknown vector of DOFs.

Up to now we have made no reference to the number of spatial dimensions. We turn our attention to the 1D case (d=1), and consider afterwards $d > 1$. We shall discuss the discretization from two perspectives, which we will refer to as the (finite) *element* point of view and the *basis* (function) point of view respectively (Fig. 2).

Finite Elements In the element point of view, the approximation function is described by its restriction onto each element.

Basis Functions In the basis point of view, the approximation function is chosen from the space spanned by the basis functions.

We now consider adaptive refinement and observe that the strategies suggested by the alternative points of view are quite different.

Element Refinement In the most simple scheme, we bisect an element to refine, and merge a pair of elements to unrefine. In bisecting an element, the linear function over the element is replaced by a piecewise linear function comprised of linear segments over the left and right subelements. The solution remains unchanged if the introduced node is the mean of its neighbors. This style of refinement is very attractive since it is entirely local: each element can be processed independently of its neighbors (Fig. 3, left).

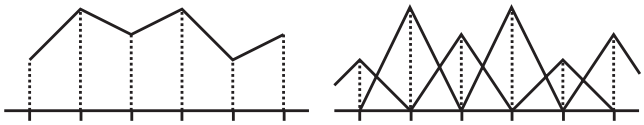

Figure 2: *Illustration of the finite-* element *(left) and* basis-*function (right) points of view using linear B-splines. In the element point of view, the solution is described over each element as a linear function interpolating the function values at the endpoints of the element. In the basis point of view, the solution is written as a linear combination of the linear B-spline functions associated with the mesh nodes.*

Basis Refinement Alternatively, we may reduce the error by enlarging the approximation space with additional basis functions. To refine, we augment the approximation space with "finer" (more localized spatially) functions; conversely to unrefine we eliminate the introduced functions. One possibility is to add a dilated basis function in the middle of an element to affect the exact same change as in the element bisection approach (Fig. 3, middle). The solution remains unchanged if the coefficient of the introduced function is zero. We refer to such *detail* or *odd* coefficients in deliberate analogy with the use of these terms in the subdivision literature [Zorin and Schröder 2000]. Bases constructed in this fashion are exactly the classical *hierarchical bases* of the FE literature [Yserentant 1986]. Note that in this setup there may be entries in the stiffness matrix corresponding to basis functions with quite different refinement levels j.

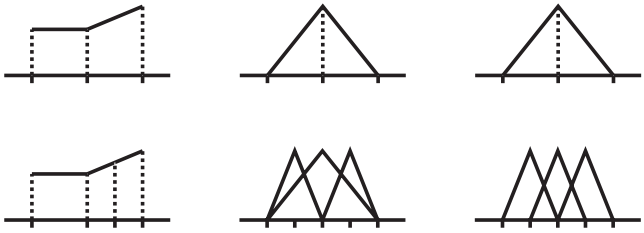

Figure 3: *Comparison of element refinement (left), and basis refinement in hierarchical (middle) or quasi-hierarchical fashion (right). For linear B-splines, each hierarchical introduction of a finer odd basis function (middle) affects the same change as element bisection (left). Quasi-hierarchical refinement (right) uses the refinement relation between the coarse basis function and its dilates.*

Alternatively we may take advantage of the fact that the hat function observes a *refinement relation*: it can be written as the sum of three dilated hats (Fig. 3, right). We may replace one of the basis functions by three dilated versions. Once again with appropriately chosen coefficients the solution is unaltered. To distinguish this approach we will later refer to it as *quasi-hierarchical* (Fig. 3, right). Here too we will have entries in the stiffness matrix which correspond to basis functions from different levels. In practice the disparity between levels will not be high since coarser functions are entirely replaced by finer functions, not just augmented, as in the hierarchical basis refinement method. To further illustrate this, consider uniformly refining the original coarse-level basis: the resulting hierarchical basis consists of both coarse- and finer -level functions, whereas the corresponding quasi-hierarchical basis consists entirely of finer -level functions, i.e., it has a *flat* , not hierarchical structure.

2.2 Higher Order Approximation in 1D

Because piecewise linear functions were sufficient for the discretization of the weak form of Laplace's equations, we have seen very few differences between the element and basis points of view, except when considering adaptive refinement strategies. This will now change as we consider a fourth order elliptic problem, the bilaplacian with prescribed displacements and normal derivatives at the boundary,

$$\Delta^2 u(x) = 0\,, \quad u|_{\partial\Omega} = 1\,, \quad \frac{\partial u}{\partial n}|_{\partial\Omega} = 0\,, \quad x \in \Omega \subset \mathbb{R}^d\,. \quad (2)$$

This kind of functional is often used in geometric modeling applications (e.g., [Gortler and Cohen 1995] and references therein). Its weak form involves second derivatives, favoring the use of basis functions which are C^1 (more precisely, they must be in H^2 [Strang and Fix 1973]).

One of the advantages of the element point of view was that each element could be considered in isolation from its neighbors. To maintain this property and satisfy the C^1 condition a natural approach is to raise the order of the local polynomial over each element. The natural choice that maintains symmetry is the Hermite cubic interpolant (Fig. 4). Two DOFs, displacement and derivative, are now associated with each vertex. Note that in using Hermite interpolants the dimension of our solution space has doubled and non-displacement DOFs were introduced—these are quite unnatural in applications which care about displacements, not derivatives.

As an alternative basis we can use quadratic B-splines (Fig. 4). They satisfy the C^1 requirement, require only displacement DOFs, and lead to smaller linear systems. Perhaps most importantly, we will see in Section 2.4 that in the bivariate, arbitrary topology setting, Hermite interpolation becomes considerably more cumbersome, while generalizations of B-splines such as subdivision methods continue to work with no difficulties .

We again compare the two perspectives for adaptive refinem ent, and learn that basis refinement applies where element refinement does not.

Element Refinement Using Hermite cubic splines it is easy to refine a given element through bisection. A new vertex with associated value and derivative coefficients is introduced in the middle of the element and the single cubic over the parent element becomes a pair of C^1 cubics over the two child elements. This refinement can be performed without regard to neighboring elements.

For quadratic (and higher order) B-splines refinement of an element *in isolation*, i.e., without regard to its neighbors, is not possible. The reason for this is that a given B-spline of degree two or higher overlaps more than two elements.

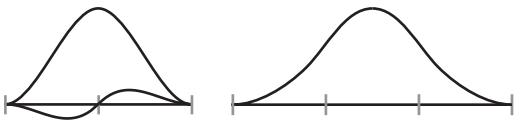

Figure 4: *Basis functions of cubic Hermites (left) and quadratic B-splines (right) give rise to C^1 approximations. The Hermite basis functions are centered at nodes and supported over adjacent elements hence allow either element or basis refinement, but they require non-displacement DOFs and do not easily generalize to higher dimensions. The B-spline basis functions have larger support hence allow only basis refinement.*

Basis Refinement Hermite basis functions admit basis refinement and the picture is substantially the same as in the hat function case. Quadratic (and higher order) B-splines, which do not admit isolated element refinement, do admit basis refinement since they all observe a refinement relation.

2.3 Piecewise Linear Approximation in 2D

In the 2D setting, we find new differences between the element and basis perspectives that were not apparent in 1D. Again we may approximate the solution to Laplace's equation (1) using a piecewise linear, i.e., C^0 function, but this time over a triangulation of the domain. The DOFs live at the vertices and define a linear interpolant over each triangle. As before, we view the discretization alternating between the *element* and *basis* points of view. The element point of view defines $u(x)$ by its *restriction* over each element, whereas the basis function point of view defines $u(x)$ as a *linear combination* of basis functions, each of which spans several elements.

Once more we compare the two perspectives for adaptive refinement, and shed new light on the simplicity of basis refinement:

Element Refinement One possibility is to quadrisect only those triangles that are 'too big." A new problem appears that did not reveal itself in the 1D setting: this approach produces a mesh with T-vertices, i.e., incompatibly placed nodes (Fig. 5). Such nodes are problematic since they introduce discontinuities. Introduction of conforming edges ('red/green" triangulations) can fix these incompatibilities [Bey 2000]. Alternatively one may use bisection of the longest edge instead of quadrisection [Rivara and Inostroza 1997]. This approach is limited to simplices only and becomes cumbersome in higher dimensions [Arnold et al. 2001].

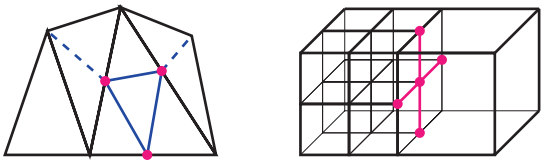

Figure 5: *Refinement of an element* in isolation *produces T-vertices, or incompatibilities with adjacent elements. In the case of 2D triangulations (left) incompatibilities may be addressed by introducing conforming edges. In other settings, e.g., quadrilateral meshes, 3D tetrahedral meshes or hexahedral meshes (right), the analogy to insertion of conforming edges is more involved. Basis refinement never leads to such incompatibilities.*

Basis Refinement Alternatively, we may augment the approximation space with finer, more compactly supported functions. Consider refining the original mesh globally via triangle quadrisection, which preserves all the existing vertices and introduces new vertices on the edge midpoints. Every node in this finer mesh associates to a (finer) nodal basis function supported by its (finer) incident triangles. We may now augment our original approximation space (induced by the coarser triangulation) with any of the nodal basis functions of the finer mesh. As such, the result is simply an expanded linear combination with additional functions. With this approach compatibility is automatic; we don't deal with problematic T-vertices.

If we augment the current basis with only *odd* finer basis functions, we generate a *hierarchical basis*. If instead we replace a coarser function with all finer (even *and* odd) functions of its refinement relation, we generate a *quasi-hierarchical* basis.

2.4 Higher Order Approximation in 2D

Consider the bilaplacian (2) in 2D; this kind of functional appears in thin-plate and thin-shell problems. As in Section 2.2 its discretization is built of C^1 basis functions; here the element point of view has a serious handicap. Building polynomials over each element and requiring that they match up globally with C^1 continuity leads to high order and cumbersome Hermite interpolation problems. On the other hand, constructing basis functions over arbitrary triangulations using, for example, Loop's [1987] subdivision scheme is quite easy and well understood. Such basis functions are supported on more than a 1-ring of triangles. Consequently, locally refining the triangulation induces a new function space which does not in general span (a superset of) the original space. In the basis point of view, the original space is augmented, thus the original span is preserved.

Summary and Preview Element refinement becomes more cumbersome or even impossible as the number of dimensions or approximation order is increased. In contrast, basis refinement applies uniformly to any refinable function space.

What are needed in the basis refinement strategy are efficient data structures and algorithms to (1) keep track of non-zero entries in the stiffness matrices and (2) manage a tesselation of the domain suitable for evaluation of the associated integrals.

In traditional, piecewise-linear elements, non-zero entries in the stiffness matrix are trivially identified with the edges of the FE mesh. When using higher order B-splines or subdivision basis functions their enlarged support implies that there are further interactions, which must be identified and managed. Additionally, interactions induced between active members of the refinement hierarchy lead to inter-level interactions. Similarly, for numerical integration, the cells of the FE mesh are a suitable tesselation when using piecewise linear elements, while for the basis refinemen t methods suitable tesselations must be explicitly constructed.

Some of these issues were confronted by earlier researchers who wished to enrich cubic B-spline tensor product surfaces with finer functions in selected regions. This was done by enforcing 'b uffer regions" of control points which were not allowed to move [Forsey and Bartels 1988; Welch and Witkin 1992] or through explicit wavelets which were resolved into B-splines based on the refinement relation [Gortler and Cohen 1995].

These earlier approaches are specialized instances of CHARMS and we now proceed to the general algorithms which rely solely on refinability .

3 Algorithms

To establish a context for our discussion, we put down a framework that might be used in an animation application to adaptively solve a nonlinear initial value problem using basis refinement:

IntegratePDE

1 **While** $t < t_{end}$
2 *predict:* measure error and construct sets $\mathcal{B}^+$ and $\mathcal{B}^-$
3 *adapt:*
4 $\mathcal{B} := \mathcal{B} \cup \mathcal{B}^+ \backslash \mathcal{B}^-$
5 maintain basis: remove redundant functions from $\mathcal{B}$
6 *solve:* $\mathbf{R}_t(\mathbf{u}_t) = \mathbf{0}$
7 $t := t + \Delta t$

Each simulation step has three stages: predict, adapt and solve. First, an oracle predicts which regions of the domain require more (resp. less) resolution, and constructs a set of basis functions to be introduced to (resp. removed from) the approximation space (line 2). Next, the approximation space is adapted: the set of active basis functions is updated (line 4), functions redundant to the basis are removed (line 5). The removal of redundant functions ensures that the set $\mathcal{B}$ is linearly independent; in certain settings this is important for numerical stability, in others this step may be skipped. The solution at time t is found by solving a system of linear or nonlinear equations (line 6). For a nonlinear system $\mathbf{R}_t(\cdot)$ we linearize and solve with Newton's method; therefore, the Jacobian matrix $\mathbf{K}_t$ and

the "load" term $\mathbf{b}_t$ need to be assembled. Note that the structure of $\mathbf{K}_t$ depends on which basis functions are active.

The framework above is one of many that could be adopted; all will have an adaptation stage, and our discussion focuses on laying out definitions and then algorithms for managing the data structures which represent the approximation space $\mathcal{B}$ and quantities depending on $\mathcal{B}$, e.g., $\mathbf{K}$.

3.1 Refinable Functions

In order to formally describe our algorithms we need to fix a number of ideas, foremost amongst them the notion of a refinable function. Traditionally, the theory of such functions (e.g., Strang and Nguyen [1996]) is pursued in the *regular* Euclidian setting, i.e., as functions from $\mathbb{R}^d$ to $\mathbb{R}$, coupled with a regular tesselation. In this case functions are linear combinations of their own dilates. Examples from 1D include B-splines and Deslauriers-Dubuc interpolating functions [1989]. In contrast, we are interested in a more general formulation: consider arbitrary topology surfaces and subsets of $\mathbb{R}^3$; in both settings the domain will in general not admit regular tesselations. We need a broader context: the theory and algorithms of *subdivision* provide such a framework [Lounsbery et al. 1997; Zorin 2000; Zorin and Schröder 2000]. In this case the finer level functions are not all strict dilates of a coarser level function, but the subdivision *stencils* still supply the basic ingredients for the *refinement relation* we need. Another setting of importance to us is that of elements with locally supported polynomials up to some desired order. The elements may be split into finer elements each carrying dilations of the coarser polynomials. The associated theory is that of *multi-scaling function* refinement [Strela et al. 1999; Alpert 1993]. Such bases have been used with great success in wavelet radiosity simulations [Gortler et al. 1993].

Our algorithms will cover both cases. To simplify the exposition we use surface subdivision as the canonical example, but will ensure that more general approaches such as volume subdivision, and multi-scaling functions defined relative to elements, are captured as well.

3.2 Building Blocks

For our purposes a *mesh* consists of sets of topological entities together with the usual incidence relations: *vertices*, $V = \{v_i\}$; *edges*, $E = \{e_j\}$; *faces*, $F = \{f_k\}$; and (in 3D) *cells*, $C = \{c_l\}$. We assume that the incidence relations define a manifold (with boundary). Typical examples include triangle, quad, tetrahedra, and hexahedra meshes. The term *element* refers to faces in the bivariate and cells in the trivariate setting.

The mesh carries *coefficients* associated with basis functions. These coefficients may describe the geometric shape, e.g., $(x, y) \in \mathbb{R}^2$ or $(x, y, z) \in \mathbb{R}^3$ or functions defined over the shape such as displacement, density, force, etc. Coefficients may "live" at any of the topological quantities. Most of the time coefficients will be associated with vertices; some schemes have coefficients associated with elements. Similarly, polynomials over individual elements will often result in coefficients associated with elements.

A *topological refinement* operator describes how topological entities are split and a finer mesh constructed with them. In developing our theory, we consider *global* refinement (all entities are split); in practice we implement adaptive refinement as *lazy evaluation* of a conceptually global and infinite refinement hierarchy. Most topological refinem ent operators split elements or vertices (Fig. 6). Less typical (but accommodated here) are 4-8 [Velho and Zorin 2001] and $\sqrt{3}$ [Kobbelt 2000] schemes.

A *coefficient refinement* operator associated with a given topological refinement operator describes how the coefficients from the coarser mesh are used to compute coefficients of the finer mesh.

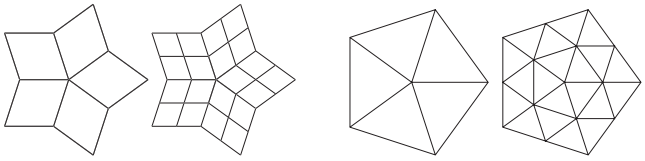

Figure 6: *Examples of topological refinement operators: quadrisection for quadrilaterals and triangles.*

We assume that these operators are linear, finitely supported, of local definition, and depend only on connectivity.

A *subdivision scheme* is a pairing of topological- and coefficient-refinement operators. Examples of common subdivision schemes include linear splines over triangles or tetrahedra; bilinear or trilinear tensor product splines over quadrilaterals and hexahedra; Doo-Sabin [1978], Catmull-Clark [1978] and their higher order [Zorin and Schröder 2001; Stam 2001] and 3D [Bajaj et al. 2002] generalizations; Loop [1987], Butterfly [Dyn et al. 1990; Zorin et al. 1996], and $\sqrt{3}$ [Kobbelt 2000] schemes for triangles. In the case of *primal* subdivision schemes, i.e., those with coefficients at vertices and splitting of faces/cells as their topological refinement operator, we distinguish between *even* and *odd* coefficients. The former correspond to vertices that the finer mesh inherits from the coarser mesh, while the latter correspond to newly created vertices.

A *basis function* is the limit of repeated subdivision beginning with a single coefficient set to unity and all others set to zero. In this way a basis function is associated in a natural way with each entity carrying a coefficient, such as vertices in the case of linear splines (both triangles and tetrahedra) or Loop's scheme, and faces in schemes such as Doo-Sabin[1] or Alpert's multi-scaling functions [1993].

A *refinement relation* is observed by all functions defined through subdivision. It states that a basis function from a coarser level can be written as a linear combination of basis functions from the next finer level

$$\phi_i^{(j)}(x) = \sum_k a_{ik}^{(j+1)} \phi_k^{(j+1)}(x) \tag{3}$$

where j indicates the level of refinement ($j = 0$ corresponding to the original, coarsest mesh), and i, respectively k index the basis functions at a given level. The coefficients $a_{ik}^{(j+1)}$ can be found by starting with a single 1 at position i on level j, applying a single subdivision step and reading off all non-zero coefficients. Note that the $a_{ik}^{(j+1)}$ generally depend on i, but for stationary schemes they do not depend on j. Since we assume that the subdivision scheme is finitely supported only a finite number of $a_{ik}^{(j+1)}$ will be non-zero. In the case of multi-scaling functions we will have matrix valued $a_{ik}^{(j+1)}$. The *children of a basis function* are given by

$$\mathcal{C}(\phi_i^{(j)}) = \{\phi_k^{(j+1)} | a_{ik}^{(j+1)} \neq 0\},$$

while the *parents* follow from the adjoint relation

$$\mathcal{C}^\star(\phi_i^{(j)}) = \{\phi_k^{(j-1)} | \phi_i^{(j)} \in \mathcal{C}(\phi_k^{(j-1)})\}.$$

The *natural support set*, $\mathcal{S}(\phi_i^{(j)})$, of a basis function is the minimal set of elements at level j, which contain the parametric support of the basis function. For example, linear splines, $\phi_i^{(j)}$ are supported on the triangles (tetrahedra) incident to v_i at mesh refinement level j; a Loop basis function has the 2-ring of triangles surrounding

[1] This is not the usual way Doo-Sabin (or other dual schemes) are described, but our description can be mapped to the standard view by dualizing the mesh [Zorin and Schröder 2001]. From a FE point of view this turns out to be more natural as it ensures that elements from finer levels are strict subsets of elements from coarser levels.

the given vertex as its natural support set (Fig. 7); and a Doo-Sabin basis function, which is centered at an element in our dualized view, has a natural support set containing the element and all elements that share an edge or vertex with it. The adjoint, $\mathcal{S}^\star(\varepsilon_l^j)$, returns the set of basis functions whose natural support contains the element ε_l^j. The *descendants of an element*, $\mathcal{D}(\varepsilon_i^j)$, are all elements at levels $> j$ which have non-zero intersection (in the parametric domain) with the given element. The *ancestor* relation is defined through the adjoint, $\mathcal{D}^\star(\varepsilon_i^j)$.

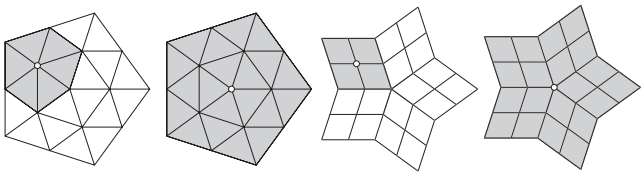

Figure 7: *Examples of natural support sets. Left to right: linear splines, Loop basis, bilinear spline, and Catmull-Clark basis.*

3.3 Putting It All Together

Synopsis of Theory Given a coarsest-level mesh, consider the infinite sequence of meshes generated by some subdivision scheme. Coefficient i on mesh level j associates to the basis function $\phi_i^{(j)}(x)$. To describe the current approximation space, we choose a finite subset $\mathcal{B} \subset \cup_{i,j}\phi_i^{(j)}(x)$ of all available basis functions; $\mathcal{B}$ is the *set of active basis functions*. The span of $\mathcal{B}$ must always include the span of the coarsest-level basis $\cup_i \phi_i^{(0)}(x)$. Refinement enlarges the space, span $\{\mathcal{B}^{old}\} \subset$ span $\{\mathcal{B}^{new}\}$; unrefinement reduces it. See also our companion paper [Krysl et al. 2002].

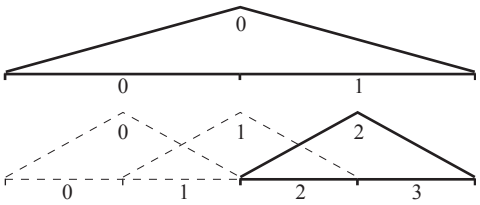

Figure 8: *Illustrative example of the data structures. Shown in bold are a pair of active basis functions on mesh levels 0 and 1. The associated data structures are:* $\mathcal{B} = \{\phi_0^{(0)}, \phi_2^{(1)}\}$, $\mathcal{E} = \{\varepsilon_0^0, \varepsilon_2^1, \varepsilon_3^1\}$, $\mathcal{S}(\phi_0^{(0)}) = \{\varepsilon_0^0, \varepsilon_1^0\}$, $\mathcal{S}(\phi_2^{(1)}) = \{\varepsilon_2^1, \varepsilon_3^1\}$, $B^s(\varepsilon_0^0) = \{\phi_0^{(0)}\}$, $B^a(\varepsilon_0^0) = \emptyset$, $B^s(\varepsilon_2^1) = \{\phi_2^{(1)}\}$, $B^a(\varepsilon_2^1) = \{\phi_0^{(0)}\}$, $B^s(\varepsilon_3^1) = \{\phi_2^{(1)}\}$, $B^a(\varepsilon_3^1) = \{\phi_0^{(0)}\}$.

Data Structures Our algorithm maintains $\mathcal{B}$, as well as an associated set of active integration elements $\mathcal{E}$; recall that *element* refers to a face in 2D and a cell in 3D. The set of active basis functions is useful for error indicators, for example, which iterate over currently active basis functions to see which must be refined and which can be unrefined (deactivated) (line 2 of **IntegratePDE**). The set of active integration cells is required to evaluate the weak-form integrals, e.g., to compute stiffness matrix entries. These sets are initialized as $\mathcal{B} = \{\phi_i^{(0)}\}$ and $\mathcal{E} = \{\varepsilon_i^0\}$, i.e., all basis functions (resp. integration cells) at the coarsest level. For each integration cell $\varepsilon \in \mathcal{E}$ we need to keep track of the active functions whose natural support sets overlap it: those from the same level $B^s(\varepsilon)$ and those from ancestor levels $B^a(\varepsilon)$ (Fig. 8). Initially $B^a(\varepsilon) = \emptyset$ and $B^s(\varepsilon) = \mathcal{S}^\star(\varepsilon)$ for $\varepsilon \in \mathcal{E}$.

Algorithms To evaluate the stiffness matrix, we need to be able to compute the action of the operator on pairs of basis functions. Traditionally this is done by iterating over all active elements, computing local interactions and accumulating these into the global stiffness matrix $\mathbf{K}$. With the data structures described above, we have all necessary tools at hand to effect this computation:

```
ComputeStiffness(E)
1   ForEach ε ∈ E do
2     ForEach φ ∈ B^s(ε) do
3       k_φφ += Integrate(φ,φ,ε)
4       ForEach ψ ∈ B^s(ε) \ {φ} do
5         k_φψ += Integrate(φ,ψ,ε)
6         k_ψφ += Integrate(ψ,φ,ε)
7       ForEach ψ ∈ B^a(ε) do
8         k_φψ += Integrate(φ,ψ,ε)
9         k_ψφ += Integrate(ψ,φ,ε)
```

Here we used $+=$ (and later $\cup=$ and $\setminus=$) in C-language fashion to indicate a binary operation with the result assigned to the left hand side. **ComputeStiffness** considers interactions between every pair of overlapping basis functions *at the coarsest level that captures the interaction:* if coarser function ϕ_c overlaps finer function ϕ_f, we evaluate the bilinear form over cells in the natural support set of ϕ_f which also support ϕ_c: $\{\varepsilon \,|\, \varepsilon \in \mathcal{S}(\phi_f) \wedge \mathcal{D}^\star(\varepsilon) \cap \mathcal{S}(\phi_c) \neq \emptyset\}$. With this approach every interaction is considered exactly once, at a sufficiently fine resolution. To implement this approach, we iterate over each active cell (line 1), and consider only interactions between every same-level active function (line 2) and every active function either on the same level (lines 3-6) or ancestral level (lines 7-9). For symmetric $\mathbf{K}$ appropriate calls to **Integrate** can be omitted. In practice, we do not call **ComputeStiffness** every time the basis $\mathcal{B}$ is adapted, rather we make incremental modifications to $\mathbf{K}$.

In some settings it may not be possible or desirable to express the integrand as a bilinear form, e.g., explicit dynamics simulators may directly compute non-linear forces. In this case, the use of a stiffness matrix is abandoned and integration is carried out over a set of *tile* cells that has the following properties: (a) the set tiles the integration domain and (b) each tile cell is at least as fine as every active cell that overlaps it. Given the set of active cells one can construct this tile set from scratch but it is preferable to maintain it incrementally.

During the course of the solution process, basis functions are (de)activated (lines 4-5 of **IntegratePDE**) and the data structures described above must be updated. When a basis function ϕ is activated $\mathcal{B}$ and $\mathcal{E}$ as well as B^s and B^a must be updated:

```
Activate(φ)
1   B ∪= {φ}
2   ForEach ε ∈ S(φ) do
3     B^s(ε) ∪= {φ}
4     // upon activation initialize ancestor list
5     If ε ∉ E then B^a(ε) ∪= Ancestor(ε) ; E ∪= {ε} fI
6     // add to ancestor lists of active descendants
7     ForEach γ ∈ (D(ε) ∩ E) do B^a(γ) ∪= {φ}

Ancestor(ε)
1   ρ := ∅
2   ForEach γ ∈ D*(ε) ∩ E do
3     ρ ∪= B^s(γ) ∪ B^a(γ)
4   return ρ
```

Activate first augments the set of active functions (line 1), and then iterates over each cell in the natural support set of ϕ (lines 2-7). Since ϕ is active, it belongs in the table of same-level active functions of every supporting cell (line 3). Furthermore since ϕ is active its supporting cells are active (line 5): they are activated (if inactive)

by adding them to the set of active cells and initializing their table of ancestral active-functions. Note here the call to **Ancestor**(ε), which returns all active coarser-level basis-functions whose natural support set overlaps ε. Finally, all active descendants of the supporting cell also support ϕ, hence we update their tables of ancestral active-functions (line 7).

Conversely, to deactivate a basis function ϕ we proceed as follows:

```
Deactivate(φ)
1   B \= {φ}
2   ForEach ε ∈ S(φ) do
3     B^s(ε) \= {φ}
4     // deactivate element?
5     If B^s(ε) = ∅ then E \= {ε}
6     // update ancestor lists of active descendants
7     ForEach γ ∈ D(ε) ∩ E do B^a(γ) \= {φ}
```

As before, we first update the set of active functions (line 1) and then iterate over the supporting cells (lines 2-7). Since ϕ has become inactive, it is removed from the table of same-level active-functions of every supporting cell ε (line 3) and from the table of ancestral active-functions of every active descendant of ε (line 7). Furthermore if the supporting cell is left with an empty active-function table then it is deactivated (line 5).

Assuming that an appropriate error estimator is at hand we can consider a wide variety of adaptive solver strategies built on top of **Activate**. Here we present two refinement strategies, hierarchical and quasi-hierarchical, as applications of **Activate** and **Deactivate** (there are other attractive strategies, e.g., selectively activating individual functions). The refinement algorithms take some active basis function $\phi \in \mathcal{B}$, modify the basis and update the vector of DOFs $\mathbf{u}$. Similarly, the unrefinement algorithms take some previously-refined basis function $\phi \notin \mathcal{B}$.

```
HierarchicalRefine (φ)
1   ForEach ψ ∈ C(φ) do
2     If ψ ∉ B ∧ Odd(ψ) then Activate(ψ) ; u_ψ := 0 fi

HierarchicalUnrefine (φ)
1   ForEach ψ ∈ C(φ) do
2     If ψ ∉ B ∧ Odd(ψ) then Deactivate(ψ)

QuasiHierarchicalRefine (φ)
1   Deactivate(φ)
2   ForEach ψ ∈ C(φ) do
3     If ψ ∉ B then Activate(ψ) ; u_ψ := 0 fi
4     u_ψ += a_{φ,ψ} u_φ

QuasiHierarchicalUnrefine (φ)
1   Activate(φ) ; initialize u_φ
2   ForEach ψ ∈ C(φ) do
3     If ψ ∉ B then Deactivate(ψ)
```

Here u_ψ is the coefficient associated with ψ, and $a_{\phi,\psi}$ is the weight of ψ in the refinement relation of ϕ (Eqn. 3). Refinement is "lossless," whereas unrefinement must be "lossy" except in the special case that the current approximation lies inside the new unrefined approximation space (as noted in the literature, unrefinement error can be hidden using interpolation techniques). Line 1 of **QuasiHierarchicalUnrefine** initializes u_ϕ by projecting the current approximation into the unrefined space, e.g., by choosing the u_ϕ that (for some given norm $\|\cdot\|$) minimizes

$$\left\| u_\phi \phi(x) - \sum_{\psi \in \mathcal{C}(\phi)} u_\psi \psi(x) \right\| .$$

In certain settings, it is important that the active functions are linearly independent. This is the case, for example, in classical FE applications, as a linear dependency in the basis leads to a singular stiffness matrix. If only hierarchical refinement is applied then the active set is always a proper basis. If other refinement strategies are used (e.g., quasi-hierarchical, and selective (de)activation of individual functions) then maintaining a proper basis requires special care. Our companion paper [Krysl et al. 2002] treats the specific case of CHARMS applied to classical FEs, i.e., a setting in which basis functions are supported on a 1-ring. There we present efficient algorithms for maintaining linear independence of the active set $\mathcal{B}$ during (un)refinement. In more general settings, approaches such as those used by Kraft may be adopted [1997]. Finally, in some settings, such as our explicit time-integration of non-linear thin-shells (Section 4), we observe that the solution process remains well-behaved even without linear independence of the active set.

With this, all the basic elements are in place to build simulators. What remains to be added are standard solvers for the resulting (non-)linear algebraic systems, error estimators appropriate for the equation to be solved, and a quadrature routine.

4 Example Applications

Here we show that CHARMS can be profitably applied to many application domains including animation, modeling, engineering, and medical simulation/visualization. To that end, we present example applications covering different types of elements, in 2D and 3D settings, with different basis functions, using hierarchical as well as quasi-hierarchical refinement. Although we have implemented these examples, our aim here is to provide a survey of the applications; to that end we have omitted details including problem specific error estimators. These are best left to the original literature.

The 2D examples employ subdivision basis functions to simulate thin flexible structures including a balloon, a metallic cylinder, and a pillow. The 3D examples employ linear tetrahedra and trilinear hexahedra to address bio-medical problems: (1) brain volume deformation during surgery; (2) stress distribution in a human mandible; and (3) potential fields in the human thorax for electrocardiography (ECG) modeling.

4.1 Non-Linear Mechanics of Thin-Shells

The thin-shell equations describe the behavior of thin flexible structures. Examples include aluminium cans, cloth, Mylar, and paper among others. The underlying PDEs, based on the classic Kirchhoff Love theory [Timoshenko and Woinowsky-Krieger 1959], describe the mechanical response of the surface to external forces in terms of the first and second fundamental forms of the original and deformed surfaces. Thin-*shells* are closely related to thin-*plates*, which are useful for variational geometric modeling and intuitive direct manipulation of surfaces. Thin-plate equations assume that the undeformed geometry is flat: the resulting equations are easier to solve but cannot capture subtleties of the nonlinear dynamic behavior of more complex shapes (Figs. 1 and below). Thin-*shell* equations accommodate arbitrary initial configurations and capture nonlinearities important for accurate modeling of stability phenomena, e.g., complex wrinkling patterns, buckling and crushing (Figs. 9 and 11). Subdivision bases are ideal for discretizing thin-shell PDEs. For example, Loop basis functions (a) naturally satisfy the H^2 smoothness requirement of these fourth order PDEs; (b) are controlled by displacements (not derivative quantities); and (c) easily model arbitrary topology. Cirak introduced the discretization of thin-shells using Loop basis functions and presented non-adaptive simulations [2000; 2001]. Adaptivity is essential for efficiently modeling complex material phenomena such as wrinkling and buckling; such simulations were the original motivation behind

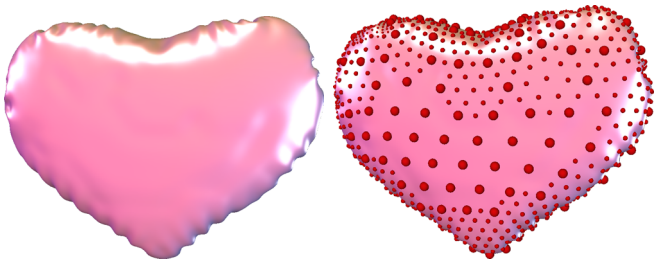

Figure 9: *Thin-shell simulation of inflating metal-foil balloon (left); red spheres represent active basis functions (right). Note the concentration of finer basis functions near wrinkles and folds.*

the development of CHARMS. Here we present one static and two dynamic simulations that demonstrate the application of CHARMS to thin-shells using Loop basis functions; the accompanying movie (filename: CHARMS.mov) is on the Conference Proceedings CD-ROM and DVD-ROM.

Inflating Balloon We simulated the dynamic behavior of a rapidly inflating metal-foil balloon (Fig. 9). The initial flat configuration has 50 nodes, and the fully-inflated configuration has 1000 active nodes. We applied internal pressure to the balloon and used quasi-hierarchical refinement over the course of the 5ms simulated inflation. Figure 10 shows the distribution of active nodes and cells near the end of the simulation; note the sparsity at the finest levels. Non-adaptive approaches require a very fine grid throughout this simulation, in contrast our adaptive approach begins with a coarse mesh and adds only necessary detail.

Poking Balloon We poked the inflated balloon with a "finger" and used quasi-hierarchical refinem ent as well as unrefinement to adapt the basis near the contact region.

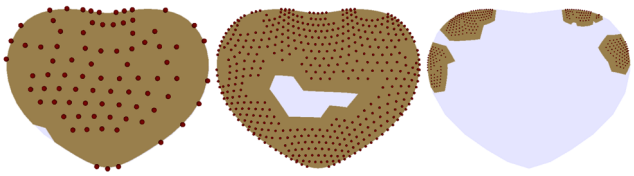

Figure 10: *Visualization of the active nodes and cells at the end of inflation. The second through fourth levels of the six-level hierarchy are shown (left to right); the fourth and finer levels are sparsely populated.*

Pillow Using the balloon-inflation technique we modeled a pillow (Fig. 1). Starting with two rectangular pieces of fabric, we applied internal pressure and solved for the equilibrium state. The adapted solution captures the fine wrinkles of the fabric. The pillow uses a thicker material (cloth) than the balloon (metal-foil), thus it forms characteristically different wrinkling patterns.

Crushing Cylinder We animated the dynamic behavior of an aluminium cylinder under compression (Fig. 11). The crushing was applied as follows: the bottom rim of the cylinder was fixed; the vertical velocity (only) of the top rim was prescribed using a linear ramp. The final animation shows the rapid buckling patterns in slow-motion.

4.2 Volume Deformation as Surgery Aid

Surgeons plan a brain operation based on landmarks from a time-consuming, pre-operative, high-resolution volume scan of the patient [Warfield et al. 2000]. After opening the skull, the surgeons may acquire additional low-resolution volume scans, which show

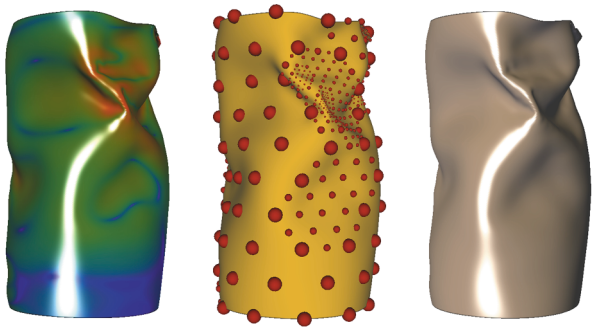

Figure 11: *Thin-shell simulation of a crushing cylinder. (left) regions with high bending force density are indicated in red; (middle) these regions have a high concentration of finer basis functions, (right) consequently the animation captures the buckling mode and its sharp folds.*

the deformation of the brain boundary surface, e.g., collapsing under gravity. However, these rapid scans do not encode landmarks. Warfield uses physical simulation with tetrahedral finite elements to infer the volume deformation from the position of the brain boundary [2000]. He maps the high-resolution scan via the computed volume deformation, and shows surgeons the shifted landmarks. CHARMS adapts the discretization to maintain high accuracy.

We modeled the volumetric deformation of the brain following the removal of cancerous tissue in the left hemisphere. Our material model is an isotropic elastic continuum [Zienkiewicz and Taylor 2000]; as the deformations are small we adopted linearized equations of equilibrium.

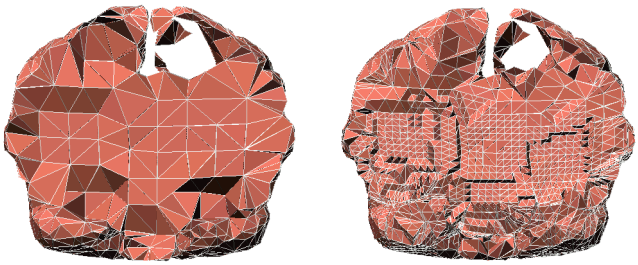

Figure 12: *The initial and refined models. Dorsal (cut-away) view at the location of the resection.*

The initial model has 2,898 nodes (5,526 DOFs) and 9,318 tetrahedral elements. We first solve for the coarse displacement field, and then refine quasi-hierarchically to 64,905 DOFs, aiming for error equidistribution. Our error metric is the strain energy density. Figure 12 shows the initial and refined meshes side by side. For comparison, a uniformly finer mesh with the same precision as the finest regions of the adapted grid would involve approximately 300,000 DOFs. Solving the volume deformation problem for the refined mesh takes 38s on a 600MHz PIII laptop with 256MB: with a two- or four- CPU PC our simulation is fast enough for actual surgical interventions.

Figure 13 shows the refined FE model viewed in the caudal direction (left). The cavity after resection is visible in this view. Note that very little refinement is introduced next to the cavity itself. The deformation of the brain due to sagging under gravity is visualized in Fig. 1 (color coded displacement amplitude with zero: red and maximum: purple), where the skull has been included as a visual aid.

4.3 Stress Distribution in Human Mandible

Numerical simulations are widely used in biomechanics to visualize response of skeletal structures to mechanical loads, as planning

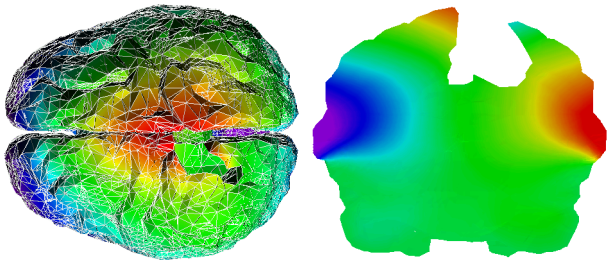

Figure 13: *Refined FE model of the brain in a caudal view with color coded displacement amplitude (zero: purple; maximal: red). Notice the cavity resulting from the surgical removal of tissue in the left hemisphere. On the right color coded lateral displacement amplitude displayed on a dorsal cutting plane.*

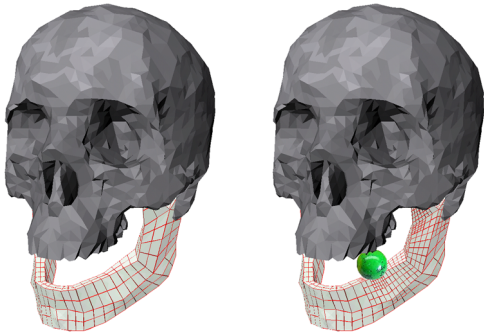

Figure 14: *Adaptive refinement in response to stress. Unstressed mandible composited with skull (left); chewing on hard candy exerts a force on the jaw (right). The model (1,700 DOFs) is refined (4,200 DOFs) in the vicinity of the applied force as well as near the corner and attachment points.*

aid for operative treatments, design of implants, and exploration of ostheosynthesis methods [Kober and Muller-Hannemann 2000].

Here we present an adaptive simulation of the response of the human mandible to the pressure involved in biting on a hard object. The internal structure of the bone is very complex, but for the purpose of this simulation we consider the bone to be homogeneous and isotropic. The initial model is a coarse approximation of the geometry of the human mandible. Figure 14 shows the original and refined FE model.

CHARMS refinemen t is achieved through octasection of each cube in the $[-1,1]^3$ reference configuration with odd vertices placed at edge, face, and cell barycenters. The initial mesh consists of 304 hexahedral, trilinear cells (1,700 DOFs; Figure 14, left). The hierarchically refined model, based on strain-energy error indication, captures the stress concentration immediately underneath the pressure point and in the thinner extremities of the mandible. It has approximately 4,200 DOFs (Fig. 14, right). We also ran this simulation with quasi-hierarchical refinement with practically identical results. Figure 15 shows a close-up of the refined hierarchical model. Active basis functions are shown as green dots, and are supported on the cells sharing that vertex. The refined basis consists of functions on three levels in the mesh hierarchy.

4.4 Potential Field in the Human Torso

Inverse problems, in which heart surface potentials are determined from measurements on the outer surfaces of the upper torso, are of particular importance in computer-assisted electrocardiography (ECG). An adaptive procedure for this problem has been outlined

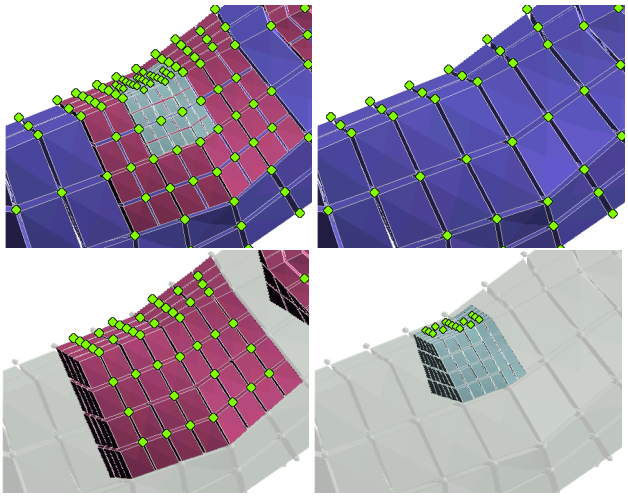

Figure 15: *Mandible FE model with a hierarchical basis. Green dots indicate nodes associated with active basis functions. Top row: quadrature cells from all levels with active nodes; cells supporting basis functions on level 1 are colored blue. Bottom row: cells supporting basis functions on level 2 and 3 are respectively colored purple and tan. Note that cells at different levels overlap, and the basis functions active on the finer levels vanish along the internal boundaries of their supporting cells, thereby guaranteeing compatibility.*

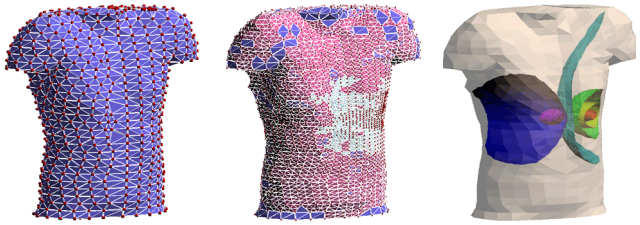

Figure 16: *ECG field analysis example. On the left the initial grid; on the right the quasi-hierarchically refined grid (four levels). Red balls indicate active basis functions, cells of different colors indicate the support of basis function on different levels. Initial surface grid courtesy of the Center for Scientific Computing and Imaging, University of Utah.*

by Johnson [2001]. Here we show how CHARMS can be applied to a subproblem of inverse ECG, the adaptive solution of the generalized Laplace equation.

Figure 16 (left) shows the initial grid with 900 nodes; (middle) a three-level quasi-hierarchically refined grid with 8,500 nodes; (right) the computed field of a dipole located on the epicardial surface is visualized through isopotential surfaces (assuming isotropic, homogeneous conductivity).

5 Conclusion and Future Work

CHARMS provide a simple framework for the construction of adaptive solvers for PDEs with applications in computer graphics, engineering, and bio-medical computing. The methods exploit refinability of basis functions and use it as a literal prescription for adaptive enrichment of approximation spaces used in Galerkin discretizations of PDEs.

On the theory side, CHARMS provide no surprises; the approach can reproduce many well studied adaptive approximation spaces of interest. The main advantage of CHARMS lies in the implementation ease. Traditional, element based, adaptive refinement strate-

gies are difficult to implement, especially in 3D. Some bases, such as higher order B-splines or subdivision surfaces do not even support elementwise refinement. In contrast, the algorithmic issues of compatibility are entirely circumvented in CHARMS, leading to rapid development and efficient management of adaptive solvers. For animations, the refinability property also avoids troublesome "popping" artifacts during refinement.

In future work we hope to explore hierarchical solvers. In principle all the machinery to apply multigrid and wavelet preconditioning techniques are in place, because of our use of refinement relations.

Acknowledgment This work was supported in part by NSF (DMS-9874082, ACI-9721349, DMS-9872890, ACI-9982273), the DOE (W-7405-ENG-48/B341492), Intel, Alias|Wavefront, Pixar, Microsoft, the Packard Foundation, and the Hellman Fellowship 2001 (PK). Special thanks to Mathieu Desbrun, Steven Schkolne, Sylvain Jaume, Christopher Malek, Mika Nystroem, Patrick Mullen, Jeff Boltz, Mark Meyer, Ilja Friedel, Joe Kiniry, Andrei Khodakovsky, Nathan Litke, and Zoë Wood.

References

ALPERT, B. K. 1993. A Class of Bases in L^2 for the Sparse Representation of Integral Operators. *SIAM Journal on Mathematical Analysis 24*, 246-262.

ARNOLD, D. N., MUKHERJEE, A., AND POULY, L. 2001. Locally Adapted Tetrahedra Meshes using Bisection. *SIAM Journal on Scientific Computing 22*, 2, 431-448.

AZAR, F. S., METAXAS, D. N., AND SCHNALL, M. D. 2001. A Deformable Finite Element Model of the Breast for Predicting Mechanical Deformations under External Perturbations. *Academic Radiology 8*, 10, 965-975.

BAJAJ, C., WARREN, J., AND XU, G. 2002. A Smooth Subdivision Scheme for Hexahedral Meshes. *The Visual Computer*. to appear.

BANK, R., AND XU, J. 1996. An Algorithm for Coarsening Unstructured Meshes. *Numerische Mathematik 73*, 1, 1-36.

BEY, J. 2000. Simiplicial Grid Refinement: On Freudenthal's Algorithm and the Optimal Number of Congruence Classes. *Numerische Mathematik 85*, 1-29.

CATMULL, E., AND CLARK, J. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *CAD 10*, 6, 350-355.

CELNIKER, G., AND GOSSARD, D. 1991. Deformable Curve and Surface Finite Elements for Free-Form Shape Design. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4, 257-266.

CIRAK, F., AND ORTIZ, M. 2001. Fully c^1-conforming subdivision elements for finite deformation thin-shell analysis. *IJNME 51*, 7, 813-833.

CIRAK, F., ORTIZ, M., AND SCHRÖDER, P. 2000. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *IJNME 47*, 12, 2039-2072.

COHEN, A., DEVORE, R., AND DAHMEN, W. 2001. Adaptive Wavelet Methods for Elliptic Operator Equations: Convergence Rates. *Mathematics of Computation 70*, 233, 27-75.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling. *Proceedings of SIGGRAPH 2001*, 31-36.

DESLAURIERS, G., AND DUBUC, S. 1989. Symmetric iterative interpolation processes. *Constructive Approximation 5*, 1, 49-68.

DOO, D., AND SABIN, M. 1978. Analysis of the behaviour of recursive division surfaces near extraordinary points. *CAD 10*, 6, 356-360.

DYN, N., LEVIN, D., AND GREGORY, J. A. 1990. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM TOG 9*, 2 (April), 160-169.

FORSEY, D. R., AND BARTELS, R. H. 1988. Hierarchical B-Spline Refinement . *Computer Graphics (Proceedings of SIGGRAPH 88) 22*, 4, 205-212.

FOSTER, N., AND FEDKIW, R. 2001. Practical Animation of Liquids. *Proceedings of SIGGRAPH 2001*, 23-30.

GORTLER, S. J., AND COHEN, M. F. 1995. Hierarchical and Variational Geometric Modeling with Wavelets. *1995 Symposium on Interactive 3D Graphics*, 35-42.

GORTLER, S. J., SCHRÖDER, P., COHEN, M. F., AND HANRAHAN, P. 1993. Wavelet Radiosity. *Proceedings of SIGGRAPH 93*, 221-230.

GOURRET, J.-P., THALMANN, N. M., AND THALMANN, D. 1989. Simulation of Object and Human Skin Deformations in a Grasping Task. *Computer Graphics (Proceedings of SIGGRAPH 89) 23*, 3, 21-30.

HALSTEAD, M., KASS, M., AND DEROSE, T. 1993. Efficient, Fair Interpolation Using Catmull-Clark Surfaces. *Proceedings of SIGGRAPH 93*, 35-44.

HOPPE, H. 1996. Progressive Meshes. *Proceedings of SIGGRAPH 96*, 99-108.

HOUSE, D. H., AND BREEN, D. E., Eds. 2000. *Coth Modeling and Animation*. A.K. Peters.

JOHNSON, C. 2001. Adaptive finite element and local regularization methods for the inverse ECG problem. In *Inverse Problems in Electrocardiology*, WIT Press, P. Johnston, Ed.

KAGAN, P., AND FISCHER, A. 2000. Integrated mechanically based CAE system using B-Spline finite elements. *CAD 32*, 8-9, 539-552.

KOBBELT, L. 2000. $\sqrt{3}$ Subdivision. *Proceedings of SIGGRAPH 2000*, 103-112.

KOBER, C., AND MULLER-HANNEMANN, M. 2000. Hexahedral Mesh Generation for the Simulation of the Human Mandible. In *Proceedings of the 9th International Meshing Roundtable*, Sandia National Laboratories, 423-434.

KRAFT, R. 1997. Adaptive and linearly independent multilevel B-splines. In *Surface Fitting and Multiresolution Methods*, A. L. Méhauté, C. Rabut, and L. L. Schumaker, Eds., vol. 2. Vanderbilt University Press, 209-218.

KRYSL, P., GRINSPUN, E., AND SCHRÖDER, P. 2002. Natural hierarchical refinement for finite element methods. To appear; *IJNME, http://hogwarts.ucsd.edu/~pkrysl/pubs/adref.pdf*.

LANGTANGEN, H. P. 1999. *Computational Partial Differential Equations*. Springer Verlag.

LEE, S., WOLBERG, G., AND SHIN, S. Y. 1997. Scattered Data Interpolation with Multilevel B-Splines. *IEEE TVCG 3*, 3, 228-244.

LOOP, C. 1987. *Smooth Subdivision Surfaces Based on Triangles*. Master's thesis, University of Utah, Department of Mathematics.

LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. 1997. Multiresolution analysis for surfaces of arbitrary topological type. *ACM TOG 16*, 1, 34-73.

MANDAL, C., QIN, H., AND VEMURI, B. C. 1997. Dynamic Smooth Subdivision Surfaces for Data Visualization. In *IEEE Visualization '97*, 371-378.

METAXAS, D., AND TERZOPOULOS, D. 1992. Dynamic deformation of solid primitives with constraints. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2, 309-312.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical Modeling and Animation of Brittle Fracture. In *Proceedings of SIGGRAPH 99*, 137-146.

RIVARA, M., AND INOSTROZA, P. 1997. Using Longest-side Bisection Techniques for the Automatic Refinement of Delaunay Triangulations. *IJNME 40*, 4, 581-597.

STAM, J. 2001. On Subdivision Schemes Generalizing Uniform B-Spline Surfaces of Arbitrary Degree. *CAGD 18*, 5 (June), 383-396.

STRANG, G., AND FIX, G. 1973. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press.

STRANG, G., AND NGUYEN, T. 1996. *Wavelets and Filter Banks*. Wellesley-Cambridge Press.

STRELA, V., HELLER, P. N., STRANG, G., TOPIWALA, P., AND HEIL, C. 1999. The Application of Multiwavelet Filterbanks to Image Processing. *IEEE TIP 8*, 4, 548-563.

TERZOPOULOS, D., AND QIN, H. 1994. Dynamic NURBS with Geometric Constraints for Interactive Sculpting. *ACM TOG 13*, 2, 103-136.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically Deformable Models. *Computer Graphics (Proceedings of SIGGRAPH 87) 21*, 4, 205-214.

TIMOSHENKO, S., AND WOINOWSKY-KRIEGER, S. 1959. *Theory of Plates and Shells*. McGraw-Hill Book Company Inc.

VELHO, L., AND ZORIN, D. 2001. 4-8 Subdivision. *CAGD 18*, 5 (June), 397-427.

WARFIELD, S. K., FERRANT, M., GALLEZ, X., NABAVI, A., JOLESZ, F. A., AND KIKINIS, R. 2000. Real-time biomechanical simulation of volumetric brain deformation for image guided neurosurgery. In *SC 2000: High performance networking and computing conference*, vol. 230, 1-16.

WELCH, W., AND WITKIN, A. 1992. Variational Surface Modeling. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2, 157-166.

WU, X., DOWNES, M. S., GOKTEKIN, T., AND TENDICK, F. 2001. Adaptive Nonlinear Finite Elements for Deformable Body Simulation Using Dynamic Progressive Meshes. *Computer Graphics Forum 20*, 3, 349-358.

YSERENTANT, H. 1986. On the multilevel splitting of finite-element spaces. *Numerische Mathematik 49*, 4, 379-412.

ZIENKIEWICZ, O. C., AND TAYLOR, R. L. 2000. *The finite element method: The basis*, 5 ed., vol. 1. Butterworth and Heinemann.

ZORIN, D., AND SCHRÖDER, P., Eds. 2000. *Subdivision for Modeling and Animation*. Course Notes. ACM Siggraph.

ZORIN, D., AND SCHRÖDER, P. 2001. A Unified Framework for Primal/Dual Quadrilateral Subdivision Schemes. *CAGD 18*, 5, 429-454.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1996. Interpolating Subdivision for Meshes with Arbitrary Topology. *Proceedings of SIGGRAPH 96*, 189-192.

ZORIN, D. 2000. Smoothness of Subdivision on Irregular Meshes. *Constructive Approximation 16*, 3, 359-397.