The Basis Refinement Method

Thesis by

Eitan Grinspun

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2003

(Defended 16 May 2003)

For my parents, Doris and Ricardo Grinspun.

Acknowledgements

Thank you Advisor: Peter Schröder; you are my mentor, my mirror, my agent, my boss, my supporter.

Thank you Committee: Peter Schröder, Mathieu Desbrun, Al Barr, Petr Krysl, and Jerry Marsden;
your direction, confidence and enthusiasm is a tremendous gift.

Thank you Cheerleaders: Zoë Wood, Steven Schkolne and Mathieu Desbrun;
you give me energy for the uphill runs.

Thank you Mathematics Advisory Board: Al Barr, Tom Duchamp, Igor Guskov, Anil Hirani, Andrei Khodakovsky, Adi Levin, Jerry Marsden, Peter Schröder and Denis Zorin, for keeping some truth.

Thank you Dear Collaborator: Petr Krysl; you have been my advocate, my sounding board, my teacher.
Your belief, foresight, and sweat helped bring to life a young idea.

Thank you Sage: Al Barr; you have shared with me many assorted life learnings.

Thank you Outside Advocates: Jerry Marsden, Jos Stam; our contact has been brief, and vital.

Thank you Multi-Res Group et al.: Burak Aksoylu, Al Barr, Jeff Bolz, Fehmi Cirak, Mathieu Desbrun, Dave Breen, Ian Farmer, Ilja Friedel, Tran Gieng, Igor Guskov, Sylvain Jaume, Andrei Khodakovsky, Petr Krysl, Adi Levin, Nathan Litke, Mark Meyer, Patrick Mullen, Mark Pauly, Steven Schkolne, Peter Schröder, Zoë Wood;
oh the excitement, the drama, the insights.

Thank you U. of Toronto DGP: Eugene Fiume, Karan Singh et al.; you are generous hosts.

Thank you Admins: Elizabeth Forrester, Diane Goodfellow, Christopher Malek, Jeri Chittum; you have saved me.

Thank you Dear Colleagues: Matthew Hanna, Joe Kiniry, Mika Nyström; I cannot start to count the things you have taught me. You cannot start to count the ways in which they have influenced my work.

Thank you Producer: Steven Schkolne; who will make it happen?

Thank you Gang for Interactive Graphics: Paul Kry, Bob Sumner, Victor Zordan;
you keep things interesting, very interesting.

Thank you Friends, Old and New: Sascha Matzkin, Alexander Nicholson, Zoë Wood, Matthew Hanna,
Roman Ginis, Dule Misevic, Weng Ki Ching, Julia Shifman, Steven Schkolne, Nathan Litke,
Tasos Vayonakis, Mathieu Desbrun, Jeroo Sinor, Santiago Lombeyda,
Jyl Boline, Robert MacDonald, Victor Zordan, Paul Kry;
you paint my life so many colors.

Thank you Guardian Angel: Mathieu Desbrun; there you are, again and again.

Thank you Dancer: Julia Shifman; your smile is my music.

Thank you Mechanic: Tasos Vayonakis; how can we be so different yet so similar?

Thank you Traveler: Zoë Wood; quite the voyage, it was, it is. So much...

Thank you Council: Gary W. Richman; you show me me.

Thank you Family: right, left, up, down; you come from all directions, showering me with love.

Thank you Grandparents and Great-grandparents: Diana e Isaac Grinspun, Sarita y Leib Rappaport, Mime y
Pola Selmick, Rebeca y Lazaro Selmick: vuestro apoyo, ideas, opiniones, orgullo y amor siempre
me acompañarán.

Thank you Brother: Yuval Grinspun;
you are my favorite brother!
You are so much, to me.

Thank you Parents: Doris and Ricardo Grinspun;
without you,
I would be nothing.

Abstract

Finite element solvers are critical in computer graphics, engineering, medical and biological application areas. For large problems, the use of adaptive refinement methods can tame otherwise intractable computational costs. Current formulations of adaptive finite element mesh refinement seem straightforward, 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 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 to novel discretizations where traditional mesh refinement concepts are not of much help, for instance on subdivision surfaces.

For classical finite-elements, adaptive refinement is typically carried out by splitting mesh elements in isolation. While so-called mesh refinement is well-understood, it is considered cumbersome to implement for unstructured three-dimensional meshes, among other settings, in particular because mesh compatibility must be explicitly maintained. Furthermore, element-splitting does not apply to problems that benefit from higher-order B-spline discretizations and their more general counterparts, so-called subdivision elements. We introduce a simple, general method for adaptive refinement which applies uniformly in all these settings and others. The basic principle of our approach is to refine basis functions, not elements. Our method is *naturally compatible*: unlike mesh refinement, basis refinement never creates incompatible meshes. Our contributions are (a) a minimal mathematical framework, with (b) associated algorithms for basis refinement; furthermore, we (c) describe the mapping of popular methods (finite-elements, wavelets, splines and subdivision) onto this framework, and (d) demonstrate working implementations of basis refinement with applications in graphics, engineering, and medicine.

Our approach is based on compactly supported refinable functions. We refine by augemnting the basis with narrowly-supported functions, not by splitting mesh elements in isolation. This removes a number of implementation headaches associated with element-splitting and is a general technique independent of domain dimension, element type (e.g., triangle, quad, tetrahedron, hexahedron), and basis function order (piecewise linear, quadratic, cubic, etc..). The (un-)refinement algorithms are simple and require little in terms of data structure support. Many popular disretizations, including classical finite-elements, wavelets and multi-wavelets, splines and subdivision schemes may be viewed as refinable function spaces, thus they are encompassed by our approach.

Our first contribution is the specification of a minimal mathematical framework, at its heart a sequence of nested approximation spaces. By construction, the bases for these spaces consist of refinable functions. From an approximation theory point of view this is a rather trivial statement; however it has a number of very important and highly practical consequences. Our adaptive solver framework *requires only* that the basis functions used be refinable. It makes *no assumptions* as to (a) the dimension of the domain; (b) the tesselation of the domain, i.e., the domain elements by they triangles, quadrilaterals, tetrahedra, hexahedra, or more general domains; (c) the approximation smoothness or accuracy; and (d) the support diameter of the basis functions. The specification of the nested spaces structure is sufficiently weak to accomodate many practical settings, while strong enough to satisfy the necessary conditions of our theorems and algorithms.

Our second contribution is to show that basis refinement can be implemented by a small set of simple algorithms. Our method requires efficient data structures and algorithms to (a) keep track of interactions between basis functions (i.e., to find the non-zero entries in the stiffness matrix), and (b) manage a tesselation of the domain suitable for evaluation of the associated integrals (i.e., to evaluate the entries of the stiffness matrix). We provide a specification for these requirements, develop the relevant theorems and proofs, and invoke these theorems to produce concrete, provably-correct pseudo-code. The resulting algorithms, while capturing the full generality (in dimension, tesselation, smoothness, etc.) of our method, are surprisingly simple.

Our third contribution is the mapping of finite-elements, wavelets and multi-wavelets, splines and subdivision schemes onto our nested spaces framework. No single discretization fits all applications. In our survey of classical and recently-popularized discretizations we demonstrate that our unifying algorithms for basis refinement encompass a very broad range of problems.

Our fourth contribution is a set of concrete, compelling examples based on our implementation of basis refinement. Adaptive basis refinement may be profitably applied in solving partial differential equations (PDEs) useful in many application domains, including simulation, animation, modeling, rendering, surgery, biomechanics, and computer vision. Our examples span thin shells (fourth order elliptic PDE using a Loop subdivision discretization), volume deformation and stress analysis using linear elasticity (second order PDE using linear-tetrahedral and trilinear-hexahedral finite elements respectively) and a subproblem of electrocardiography (the generalized Laplace equation using linear tetrahedral finite elements).

Contents

List of Figures

Notation

Indices

n, m	(integer subscript) an index to a space in a sequence of spaces
p, q	(integer superscript) an index to a space in a hierarchy of spaces
i, j, k	(integer subscript) an index to a function in a set of functions
N, M	(integers) the cardinality of a finite set

Spaces

S_n	the trial space after n refinement operations
$V^{(p)}$	the nested space at hierarchy level p

Basis functions, elements, tiles

ϕ	a basis function
ε	a domain element
t	a domain tile
$\mathcal{B}$	the set of active basis functions
$\mathcal{E}$	the set of active elements
$\mathcal{T}$	the set of active tiles

Domain and subdomains

Ω	the parametric domain
$\Omega\|_{\varepsilon}$	the parametric domain occupied by ε
$\Omega\|_{\mathcal{S}(\phi)}$	the parametric domain occupied by the natural support set of ϕ

Maps from basis functions to basis functions

$\mathcal{C}(\phi)$	the children of ϕ
$\mathcal{C}^{\star}(\phi)$	the parents of ϕ

Maps from elements to elements

$\mathcal{D}^\star(\varepsilon)$	the ancestors of ε
$\mathcal{D}(\varepsilon)$	the descendants of ε

Map from basis functions to elements

$\mathcal{S}(\phi)$	the natural support set of ϕ
	i.e., the same-level elements overlapping the support of ϕ

Maps from elements to basis functions

$\mathcal{S}^\star(\varepsilon)$	the set of same-level basis functions supported by ε
$\mathcal{B}^s(\varepsilon)$	the set of same-level active basis functions supported by ε
$\mathcal{B}^a(\varepsilon)$	the set of ancestral active basis functions supported by ε

Maps from elements to tiles

$\mathcal{L}(\varepsilon)$	finer-link of ε
	i.e., the set of overlapping resolving tiles at the same level as ε
$\mathcal{L}^*(\varepsilon)$	coarser-link of ε
	i.e., the set of overlapping resolving tiles at the next-coarser level from ε

Maps from tiles to elements

$\mathcal{L}(t)$	finer-link of resolving-tile t
	i.e., the single overlapping element tile at the next-finer level from t.
$\mathcal{L}^*(t)$	coarser-link of resolving-tile t
	i.e., the single overlapping element tile at the same level as t.

System snapshots

$\overline{S}$	snapshots of the entire system in the pre-condition state
S	snapshots of the entire system in the post-condition state

Conventions

OP*	the adjoint of the operator OP
$\overline{\mathcal{E}},\ \overline{\mathcal{B}^s(\varepsilon)},\ etc.$	barred quantities refer to the precondition state
	i.e., the state before an algorithm executes

Preface

This work is about basis refinement. This is a simple, natural, general way to transform a continuous problem into a discrete problem, capturing all (and only) the essential features of the continuous solution. It is an alternative to mesh refinement, focusing on basis functions instead of mesh elements. We will explain the advantages (and limitations!) of this method. For finite element discretizations, its adoption is optional (and often desirable). For a broader class of discretizations, its adoption is almost unavoidable. There are already concrete, compelling applications of this method, such as those illustrated in the figure below.

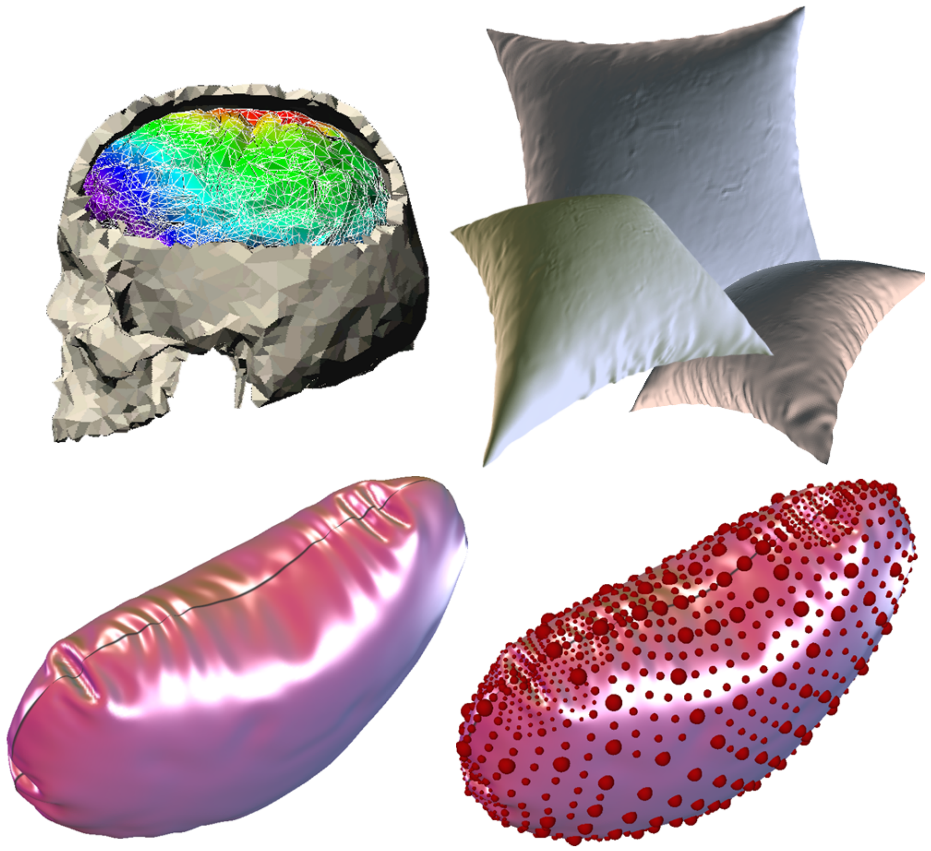


Figure 1: Example problems solved adaptively using basis refinement. A variety of physical simulations benefit from our general adaptive solver framework: crushing cylinder, medical planning, surgery simulation, and pillows. For details see Chapter 5.

In this preface we develop a motivating example. Many of the key ideas will show up. Starting with Chapter 1 we will explain everything again, in more detail. In this preface, we assume that the reader is familiar with the method of *finite elements* (FEs) [Strang and Fix 1973].

Piecewise Linear Approximation in One Dimension

We consider a simple example to elucidate the difference between finite-element and basis-function refinement. As a canonical example of a second order elliptic PDE consider the Laplace equation with an essential boundary condition,

$$\nabla^2 u(x) = 0, \quad x \in \Omega, \quad u|_{\partial\Omega} = \bar{u}\,.$$

To keep things simple, consider for now a one dimensional domain, $\Omega \subset \mathbb{R}$, with boundary conditions $u(0) = u(1) = \bar{u}$. An FE method typically solves the weak form of this equation, selecting from the *trial space* the solution U which satisfies

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

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 $u(0) = u(1) = 0$. In the Galerkin method, which we adopt in this preface, 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, i.e., C^0 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 (see Figure 2). 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.

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.

Finite Elements In the element point of view, the approximation function is described by its restriction onto each element (see Figure 2-*left*).

Basis Functions In the basis point of view, the approximation function is chosen from the space spanned by the nodal basis functions (see Figure 2-*right*).

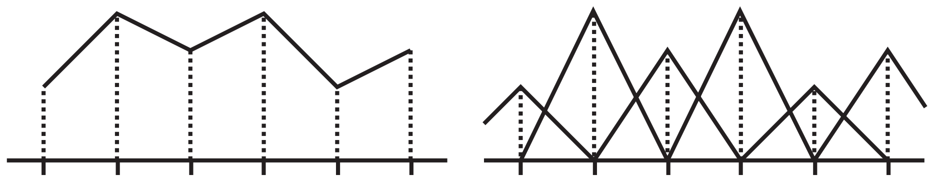


Figure 2: Illustration of the finite-*element* (left) and *basis*-function (right) points of view using linear B-splines. In the element point of view, the solution is described over each element as a linear function interpolating the function values at the endpoints of the element. In the basis point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

Adaptive Refinement

We consider adaptive refinement. The two perspectives lead to very different strategies! An *adaptive solver*, guided by an *error indicator*, refines the solution process by *locally adjusting the resolution of the discretization*: in the element perspective, this is *refinement of the domain partition;* in the basis perspective, this is *enrichment of the approximation space.*

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* (see Figure 3, left).

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 spatially-localized) 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 effect the same space as element bisection (see Figure 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.

Alternatively we use the *refinability* of the linear B-spline basis functions:

Refinement relation

The *hat* function can be written as the *sum of three dilated hats* (see Figure 3).

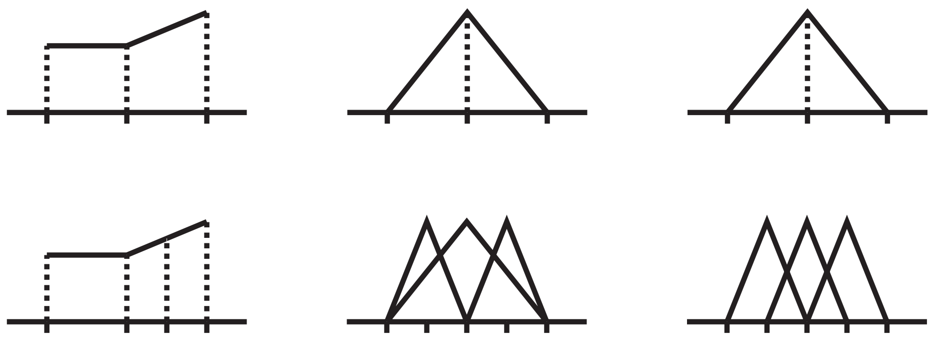


Figure 3: Comparison of element refinement (left), and basis refinement with *details* (middle) or *substitution* (right). Observe that for linear B-splines, each introduction of a finer odd basis function (middle) effects the same change as element bisection (left). Element refinement does not change the current approximation if the introduced coefficient is chosen as the mean of its neighbors; likewise for detail refinement if the coefficient of the detail function is zero, and for substitution refinement if the even and odd coefficients are chosen as u_i and $\frac{1}{2}u_i$ respectively, where u_i is the coefficient of the removed function.

We may *substitute* one of the basis functions by three dilated versions, *following the prescription of the refinement relation.*. Once again with appropriately chosen coefficients the solution is unaltered. Here too we will have entries in the stiffness matrix which correspond to basis functions from different levels. In contrast to refinement with *details*, this refinement using *substitution* in practice leads to little disparity between (coarse and fine) levels, since coarser functions are *entirely replaced by finer functions*.

Higher Order Approximation in One Dimension

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, excepting differing approaches to adaptive refinement. Things are dramatically different when we work with a fourth order elliptic problem! Consider the biharmonic equation with interpolation and flux boundary conditions,

$$\nabla^4 u(x) = 0\,, \quad x \in [0,1]\,, \quad u|_{\partial\Omega} = \bar{u} \quad \vec{n}\cdot\nabla u|_{\partial\Omega} = 0\,.$$

This kind of functional is often used, e.g., in geometric modeling applications. Its weak form involves second derivatives, necessitating[1] basis functions which are in H^2.

As before, we begin with a one dimensional example: $\Omega \subset \mathbb{R}$, $u(0) = u(1) = \bar{u}$, and $u'(0) = u'(1) = 0$. 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

[1]Even though the biharmonic equation requires derivatives of fourth order to vanish, any solution to it can be *approximated* by functions which are only C^1.

Hermite cubic interpolant (see Figure 4). Two degrees of freedom (DOFs), displacement and derivative, are now associated with each vertex. The resulting piecewise cubic function is clearly C^1 since the appropriate *compatibility conditions* are satisfied between elements incident on a given vertex (see Figure 4). 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.

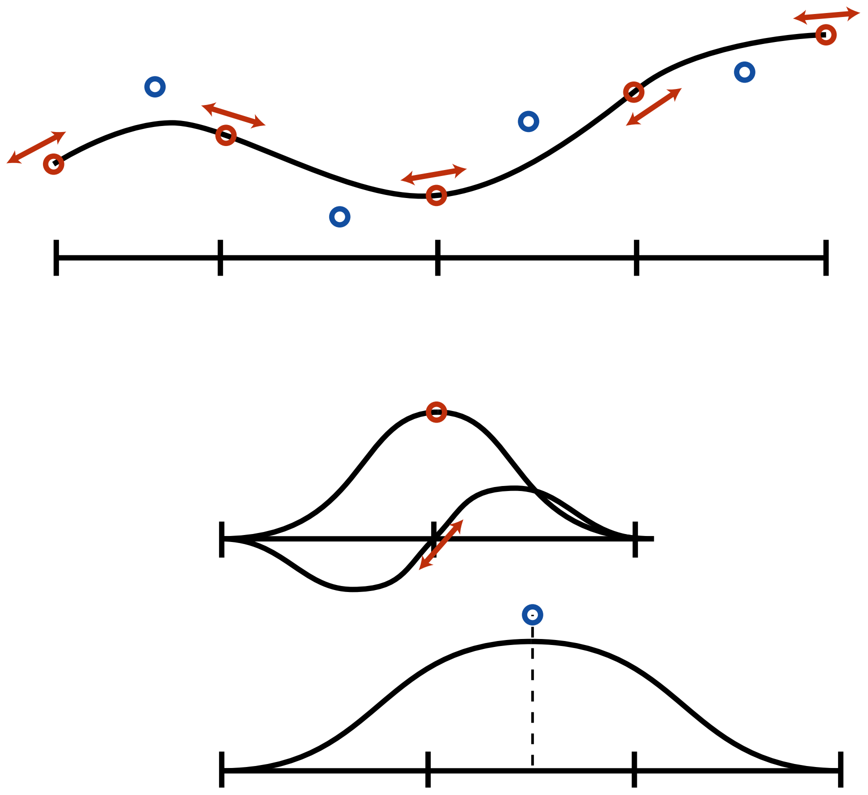


Figure 4: Basis functions of cubic Hermites (top row) and quadratic B-splines (middle row) give C^1 approximations (bottom row). The Hermite basis functions are centered at nodes and supported over adjacent elements hence allow either element or basis refinement, but they require non-displacement DOFs (red arrows denoting tangents) as well as displacement DOFs (red circles) and do not easily generalize to higher dimensions. The B-splines have only displacement DOFs (blue diamonds) but the curve is non-interpolating. There are two kinds of Hermite basis functions (associated to displacement and derivative coefficients, respectively); there is one kind of B-spline basis function. The B-spline basis functions have larger support hence allow only basis refinement.

As an alternative basis we can use quadratic B-splines (see Figure 4). They satisfy the C^1 requirement, require only displacement DOFs, and lead to smaller linear systems. If this is not enough motivation for B-splines, we will soon learn 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 element and basis approaches to adaptive refinement, and learn that basis refinement outruns element refinement:

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 impossible!* B-spline basis functions of degree two or higher overlap more than two elements; this is trouble for isolated element refinement.

Basis Refinement Hermite basis functions are refinable thus admit basis refinement. The picture is the same as in the hat function case, except that two basis functions are associated with each vertex, and a different (*matrix*) refinement relation holds. Quadratic (and higher order) B-splines, which do not admit isolated element refinement, do admit basis refinement since they all observe a refinement relation. So long as a refinement relation holds, basis refinement doesn't care what discretization we use, be it linear, Hermite, quadratic B-Spline, etc..

Piecewise Linear Approximation in Two Dimensions

In the two dimensions, we find new differences between the element and basis perspectives that were not apparent in one dimension. We return to Laplace's equation with an essential boundary condition, this time with $\Omega \subset \mathbb{R}^2$.

Again we may approximate the solution using a piecewise linear, i.e., C^0 function this time over a *triangulation* (or some other *tesselation*) 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 (finite) *element* and *basis* (function) 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*.

Comparing the two perspectives in two dimensions sheds new light on the simplicity of basis refinement:

Element Refinement One possibility is to quadrisect *only* those triangles that are *excessively coarse* (as determined by some error indicator). A new problem appears that did not reveal itself in one dimension: this approach produces an *incompatible* mesh, i.e., incompatibly placed nodes (known as *T-vertices* after the T shape formed by incident edges), shown in Figure 5. Such nodes are problematic since they introduce discontinuities. Introduction of conforming edges (e.g., *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 very cumbersome in higher dimensions [Arnold et al. 2001].

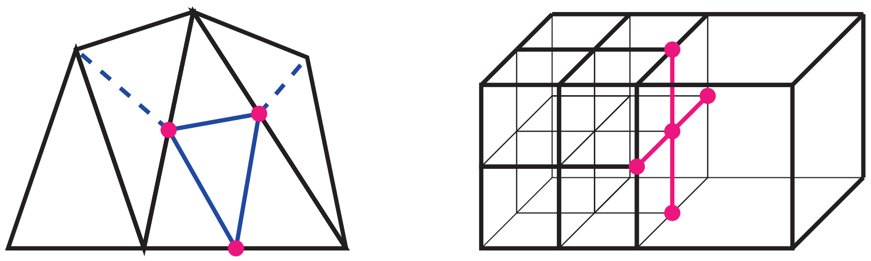


Figure 5: Refinement of an element *in isolation* produces T-vertices, or incompatibilities with adjacent elements. In the case of 2D triangulations (left) incompatibilities may be addressed by introducing conforming edges; in other settings, e.g.. quadrilateral meshes, 3D tetrahedral meshes or hexahedral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

As before, we may *augment* the current basis with *odd* finer basis functions (i.e., *details*), or instead we may *substitute* a coarser function with all finer (even *and* odd) functions of its refinement relation.

Higher Order Approximation in Two Dimensions

The weak form for the Laplace equation requires only C^0 basis functions (integrable first derivatives). This changes as we consider fourth order elliptic equations, which appear in thin plate and thin shell problems. For example, thin plate energies are used extensively in geometric modeling. The weak form of the associated membrane and bending energy integrals involves second derivatives, necessitating basis functions which are in H^2.

In this setting 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 (and is one of the methods we pursue). Such basis functions are supported on more than a one-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 cumbersome, intractable or even impossible as the number of dimensions or approximation order is increased. In contrast, basis refinement applies naturally to any refinable function space. We invite the reader to fill whichever details interest them most: Chapter 1 presents a background and overview, Chapter 2 lays out the basic theory which leads naturally to the simple algorithms presented in Chapter 4; these apply to a general class of discretizations (Chapter 3) and have compelling applications (Chapter 5) in graphics, engineering, and medicine.

Chapter 1

Background and Motivation

Partial differential equations (PDEs) model fascinating problems; they are the foundation of critical applications in computer graphics, engineering, and medical simulation [Eriksson et al. 1996]. Although *adaptive* solvers for PDEs and integral equations promise to reduce computational cost while improving accuracy, they are not employed broadly. Building adaptive solvers can be a daunting task, evidence the large body of literature on *mesh refinement* methods. We argue for a paradigm shift: our method refines basis functions *not* mesh elements. This removes several implementation headaches associated with other approaches and is a general technique independent of domain dimension and tesselation as well as approximation space smoothness and accuracy. Our approach *unifies* earlier ideas from the wavelets and hierarchical splines literature; it *generalizes* these approaches to novel settings via the theory and structure of *subdivision*.

1.1 Introduction

Many computer applications involve modeling of physical phenomena with high visual or numerical accuracy. Examples from the graphics literature 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. Examples from the mechanics literature include elasticity of continuous media such as solids, thin-plates and -shells [Malvern 1969], conductive thermal transfer [Hughes 1987, Lewis et al. 1996], and turbulent fluid flow [Bernard and Wallace 2002]. 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]. Alternatively the underlying formulation is in terms of *integral equations*, e.g., Kajia's rendering equation [1986] and boundary integral equations for heat conduction [Divo and Kassab 2003], or ordinary differential equations (ODEs) which appear, e.g., in control problems [Dullerud and Paganini 2000].

Most often the continuous equations are discretized with the finite difference (FD) or Galerkin, e.g., finite element (FE), method before a (non-)linear solver can be used to compute an approximate solution to the original problem [Strang and Fix 1973, Eriksson et al. 1996]. For example, Terzopoulos and coworkers described methods to model many physical effects for purposes of realistic animation [1987b, 1988]. Their discretization was mostly based on simple, uniform FD approximations. Later Metaxas and Terzopoulos did employ finite element (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, Chen and Zeltzer 1992, Keeve et al. 1996, Koch et al. 1996, Roth et al. 1998, Azar et al. 2001]).

To reduce computation and increase accuracy we use *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, or *mesh*, should be adjusted based on local error estimators [Babuška et al. 1986]. 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 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 (e.g., for multigrid solvers [Bank et al. 1988, Wille 1996]) we have adopted a generalized form of refinement (and unrefinement) as our basic strategy.

Typical mesh refinement algorithms approach the problem of refinement as one of splitting mesh *elements* in isolation. Unfortunately this leads to a lack of *compatibility* (also known as *cracks* caused by *T-vertices*); to deal with this issue one may: (a) constrain T-vertices to the neighboring edge; (b) use Lagrange multipliers or penalty methods to numerically enforce compatibility; or (c) 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 [Carey 1997]. 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 1995, Bey 2000]. As a result various different, specialized algorithms exist for different element types such as triangles [Rivara and Iribarren 1996, Bank and Xu 1996, Rivara and Inostroza 1997], tetrahedra [Wille 1992, Liu and Joe 1995, Liu and Joe 1996, Plaza and Carey 2000, Arnold et al. 2001] and hexahedra [Langtangen 1999].

This lack of a general approach coupled with at times 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 co-workers [1991, 1992] used higher order finite elements 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: for B-spline or subdivision bases, elements *cannot* be refined individually without losing nestedness of the approximation spaces. 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 an adaptive solver are specialized instances of *basis refinement*, in which basis functions *not* elements are refined. From an approximation theory point of view this is a rather trivial statement; however it has a number of very important and highly practical consequences. Our adaptive solver framework *requires only* that the basis functions used be refinable. It makes *no assumptions* as to (a) the dimension of the domain; (b) the tesselation of the domain, i.e., meshes made of triangles, quadrilaterals, tetrahedra, hexahedra or more general meshes; (c) the approximation smoothness or accuracy; and (d) the support diameter 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. Our method applies whenever a finite-basis discretization is suitable (e.g., this is the case for Ritz, Galerkin, or collocation methods), and accommodates both ODE, PDE and integral formulations. We demonstrate its versatility by applying it to several different PDEs, involving both surface and volume settings (see Figure 1 and Chapter 5).

1.2 Background and Overview

PDEs appear frequently in graphics problems including simulation, modeling, visualization, and animation. Integral equations are less common but very important in problems including global illumination, or *radiosity* [Greenberg et al. 1986]. In Chapter 5 we survey these applications as well as others, spanning graphics, mechanics, medicine, vision and control of dynamical systems.

The continuous problem must be made *discrete* before it is numerically solved by the computer. Our work builds on the idea of a *finite-basis discretization*, in which the unknown continuous solution $u(x)$ is projected onto the *trial space* of linear combinations of a fixed *finite* set of *basis* functions, i.e., $P_N u(x) = \{\sum_N u_i \varphi_i(x)\}$, where P_N is the projection or *discretization* operator (see Section 2.1). Chapter 3 describes different finite-basis discretizations including wavelets, multiwavelets, finite elements, and subdivision schemes. Discretizations which do not explicitly use basis functions, e.g., *finite-differences*, are beyond the scope of this work and their description we leave to the literature [Strang and Fix 1973, Mickens 2000].

In *mesh-based* approaches, the basis functions are defined piecewise over some tesselation of the domain. There is a large body of literature, [Field 1995, Carey 1997, Thompson et al. 1999], on the theory and practice of mesh generation; it remains a costly component of mesh-based approaches, because (a) the domain of the underlying problem may have complex geometry, (b) the domain may change over time, e.g., when Lagrangian coordinates are used, and (c) adaptive computations repeatedly modify the mesh. The cost of managing meshes motivates recent developments in *mesh-less* methods. Here the basis functions are defined without reference to a tesselation. Babuška et al. [2002] recently presented a unified mathematical theory and review of meshless methods and the closely related *generalized finite elements*.

Having chosen a particular mesh-based or mesh-less method, the discrete problem must be posed, typically as a *collocated*, *weak* (e.g., Galerkin, Petrov-Galerkin), or *variational* formulation (see Section 3.1.2), and the appropriate numerical solver invoked. In general the discrete formulation may be *linear*, in which case a numerical linear-system solver is used [Press et al. 1993], or *non-linear,* in which case a specialized solver or numerical optimizer is required [Eriksson et al. 1996].

In all cases, functions in the approximation space are interrogated —most often *integrated*— via exact or approximate *numerical quadrature* (see Section 2.3 and [Richter-Dyn 1971, Atkinson 1989]). This involves evaluating the approximate solution over an appropriate tesselation of the domain, which in the mesh-based case may (or may not!) be the same as the function-defining mesh.

An *adaptive* solver modifies the discretization over the course of the computation, *refining* to increase precision, and *unrefining* to economize computation. The *error indicator* guides these decision to (un)refine [Eriksson et al. 1996]. Although error criteria continue to be an active and interesting area for improvement, they are not our focus here: we use standard and widely-accepted error indicators in all our examples, as our approach is compatible with existing error indicators.

The solver may refine the discretization in a *structured* or *unstructured* manner. In structured refinement the discretization, at any instant of the computation, may be reconstructed in a systematic way from some fixed underlying *multi-resolution structure* (see Chapter 2). For example, wavelet bases are always a subset of the complete multi-resolution set of wavelets (see Section 3.2 and [Strang 1989]). A key consequence is that the solver may arrive at a particular adapted discretization through various permutations of a sequence of local refinements. In contrast, *un*structured approaches are free of an underlying multi-resolution structure; this gives freedom of refinement, at some costs. For example, an unstructured discretization might carry no record, or *history*, of the sequence of refinements that it experienced. A straightforward example of unstructured approaches with no history are *remeshing* methods, which adapt the discretization by reconstructing from scratch the entire tesselation [Thompson et al. 1999]. Structured methods inherit much of the approximation and multi-resolution theory that has been developed over the past decades [Méhauté et al. 1997, DeVore 1998, Cohen 2003a]; that is our primary motivation for adopting a structured approach.

For mesh-based discretizations, most approaches to adaptivity focus either on *mesh* or *basis* refinement, the former increasing the resolution of the tesselation and consequently the basis, the latter increasing the resolution of the basis directly. In either case, once the resolution of the basis is increased, the resolution with which numerical quadrature is performed must be appropriately adjusted. This is automatic for mesh refinement in the typical case that the function-defining mesh is also the quadrature mesh; that is so, e.g., for finite elements.

Among the most popular mesh-based discretizations are *finite elements*, which produce piecewise polynomial trial spaces; they are featured in many graphics applications, most recently in real-time animation [Halstead et al. 1993, Müller et al. 2002, Kry et al. 2002, Capell et al. 2002a], simulation of ductile fracture [O'Brien et al. 2002], human tissue [Snedeker et al. 2002], and the sound of vibrating flexible bodies [O'Brien et al. 2001]; also in global illumination [Troutman and Max 1993, Szirmay-Kalos et al. 2001], computer-aided design (CAD) applications [Qian and Dutta 2003], as well as computer-aided modeling (CAM) applications such as procedural modeling [Cutler et al. 2002], interactive surface design [Halstead et al. 1993, Terzopoulos and Qin 1994, Mandal et al. 2000, Mandal et al. 1997], and so forth [Celniker and Gossard 1991, Celniker and Welch 1992, Terzopoulos et al. 1987b, Terzopoulos and Fleischer 1988],
[Metaxas and Terzopoulos 1992, Welch and Witkin 1992, Gortler and Cohen 1995].

State of the art *adaptive* finite-elements use element-centric discretizations: the basis functions are defined piecewise over each element, with some *compatibility condition* for the interfaces between element subdomains, e.g., the function must be continuous on the edge between two elements, alternatively its mean value must be the same on both sides of the edge, and so forth; for examples of compatibility conditions please see Section 3.4 and [Crouzeix and Raviart 1973, Rannacher and Turek 1992, Raviart and Thomas 1977], [Brezzi and Fortin 1991].

For finite-element approaches, refinement is *geometric division of mesh elements.* Unfortunately, local element splitting does not in general ensure global compatibility of the modified mesh. As mentioned earlier, several number of approaches (constraints, Lagrange multipliers, etc.) are used to resolve this issue [Carey 1997]. Local element splitting appears simple at first. But it does not generalize easily, evidence an abundance of algorithms specialized to particular tesselations, e.g., [Wille 1992, Liu and Joe 1995, Liu and Joe 1996, Plaza and Carey 2000, Arnold et al. 2001, Langtangen 1999, Rivara and Iribarren 1996], [Bank and Xu 1996, Rivara and Inostroza 1997]. A critical review of the existing mesh-based adaptive algorithms suggests they tend to be complex (adding constraints or splitting neighboring elements), or lead to undesirable algorithmic features (Lagrange multipliers or penalty methods). A general, flexible, and robust mesh refinement algorithm should at the same time be *simple* to implement.

Mesh-*less* discretizations discard the function-defining mesh thus avoiding these difficulties. Here the trial spaces are again linear combinations of basis functions with either global or local support, but the basis functions no longer have simple forms over domain elements, e.g., in contrast to the piecewise polynomial basis functions of the finite elements. In mesh-less methods, numerical integration is not straightforward: the function supports are not aligned to (geometrically simple) mesh elements, hence numerical quadrature might require a more involved tesselation built *not* from simple shapes such as triangles, simplices, etc.. Furthermore, these methods require special care in the presence of *essential* boundary conditions [Strang and Fix 1973]. Finally, the resulting linear systems may be singular, preventing the use of (among others) multigrid solvers. These difficulties can (and have been) overcome [Babuška et al. 2002], but at the loss of some of the simplicity of the mesh-less idea.

Since for these methods the mesh is absent, the natural approach to adaptivity is *basis*-refinement. Typically, this means *augmenting the basis with more basis functions*: such a process *by construction* strictly enlarges the approximation space; consequently a sequence of refinements produces *nested* approximation spaces. Recall that for numerical quadrature, mesh-less approaches still require tesselation. But it is *in the background.* It is modified as a *consequence* of changes to the basis.

Our work is inspired by this idea: the basis leads, the structures for numerical quadrature follow. Our method is *not* mesh-less. That is its simplicity. Meshes lead to simple basis functions thus simple algorithms and quadrature schemes. Our basis functions are made of simple forms over simple mesh elements. But we learn from the mesh-less paradigm: *focusing on the mesh elements is harmful; focusing of the basis functions is conceptually desirable*. We demonstrate that it is also *algorithmically* desirable.

Chapter 4 presents simple algorithms for implementing basis refinement on mesh-based discretizations. We implemented these algorithms and applied them to several pragmatic problems 5. We begin, in the following chapter, by laying down the basic structure that unifies mesh-based multi-resolution finite-basis discretizations.

Chapter 2

Natural Refinement

Finite-basis discretization are a basis for many popular approximation strategies. When it is used to approximately solve a variational or boundary-value problem, two distinct concepts must collaborate: *approximation* space and numerical *integration*. If we begin with a hierarchy of *nested approximation spaces*, then adaptive refinement follows naturally. The remaining goal is to make numerical *integration* easy for this class of approximations. We present a lightweight framework to make this possible, first (in this chapter) in the abstract, then (in the following chapter) using concrete examples.

2.1 The Finite-Basis Discretization

Consider some variational problem in a Hilbert space H over a parametric domain Ω—we wish to find a function $u(x) \in H(\Omega)$ minimizing some expression of potential energy. Alternatively, consider some boundary value problem—we wish to find a $u(x)$ satisfying some partial differential equation (PDE) and boundary conditions. In either case, computing the exact solution is often impossible or intractable, and we settle for an approximate solution.

The finite-basis discretization is to choose a finite linearly-independent[1]set of *basis* functions $\mathcal{B}_0 = \{\varphi_1(x), \ldots, \varphi_N(x)\}$, and to approximate $u(x) \in H(\Omega)$ as a linear combination of these basis functions.

The weighted residual method is to choose among the basis functions of the *trial space* $S_0 = \{\sum_N u_i\varphi_i(x)\}$ the minimizing function; in the variational case minimizing refers to potential energy, in the PDE case minimizing refers to some measure of the residual. This minimizing function is the best approximation to $u(x)$ for the given finite-basis discretization—that is the principle of the weighted residual method[2]. Specializations of this approach include the Ritz, Galerkin, Petrov-Galerkin and collocation methods [Strang and Fix 1973].

In general, the weighted residual method forms a *sequence* of trial spaces, $S_0,\ S_1,\ S_2, \ldots$, which in the limit is dense in the solution space, i.e.,

$$\lim_{n\to+\infty} \|u - P_n u\|_H = 0 \ ,$$

where P_n is the orthogonal projector onto S_n. The above property of *diminishing errors* is the necessary and sufficient condition for convergence of the weighted residual method: for every admissible $u \in H$, the distance to the trial spaces S_n should approach zero as $n \to \infty$. [3]

The weighted residual method turns a search for a continuous function u into a search for a finite set of coefficients $\{u_i | 1 \leq i \leq N\}$ which correspond to a given finite-basis discretization. In many cases the search may be formulated as a system of N discrete algebraic equations—a tractable and well-understood computational problem.

2.2 Refinement

The choice of trial space determines the approximation error as well as the computational cost of finding the approximate solution. The former and latter can be traded-off by employing *un*refinement.

In the broadest sense, refinement is an alteration of a given approximation space to reduce the approxi-

[1]In many applications we relax this requirement and use a set of linearly-dependent "basis" functions (in an abuse of terminology, here we write "basis" but mean "spanning set"). For the remainder of this discussion we will use this relaxed definition, explicitly treating linear-independence only in Section 2.4.

[2]In general, the test functions of the weighted residual method may also be constructed by a finite-basis discretization [Strang and Fix 1973]. We elaborate on this in Section 3.1.2. For simplicity, we discuss only the trial spaces in this chapter, however basis refinement applies also to the test spaces.

[3]A sufficient, but not necessary, condition for diminishing errors is the construction of *nested* spaces, $S_0 \subset S_1 \subset S_2 \subset \ldots$, which assures that approximation error is never increasing; this is stronger than required for the weighted residual method.

mation error—at the expense of increased computational cost. Conversely, *un*refinement is an alteration to reduce computational cost—at the expense of increased approximation error.

More precisely, *refinement of S_n creates a larger trial space $S_{n+1} \supset S_n$*. Conversely, unrefinement creates a smaller trial space $S_{n+1} \subset S_n$. The difficulty lays in imposing sufficient structure, simplifying implementation on a computer, while keeping the idea and its implementation broadly applicable in many settings.

2.2.1 Nested Spaces

We build our structure over a given sequence of nested function spaces[4].This sequence, when combined with associated Riesz bases, make up our refinement *scheme*.

Nested Spaces We are given an infinite sequence of nested spaces, $V^{(0)} \subset V^{(1)} \subset V^{(2)} \subset \dots$, which in the limit is dense in the solution space.

Detail Spaces We define an associated sequence, $D^{(0)},\ D^{(1)},\ D^{(2)}, \dots$, where the *detail space* $D^{(p)}$ is the complement of $V^{(p+1)}$ onto $V^{(p)}$, i.e., the *finer* space $V^{(p+1)}$ may be expressed as the direct sum $V^{(p)} \oplus D^{(p)}$ of the *immediately coarser* space and its details.

Riesz Bases A *Riesz basis* of a Hilbert space H is defined as any set $\{\phi_i | i \in \mathbb{Z}\}$ such that

1. $\{\phi_i | i \in \mathbb{Z}\}$ spans H, i.e., finite linear combinations $\sum u_i \phi_i$ are dense in H, and
2. there exist $0 < C_1 \leq C_2$ such that for all finitely supported sequence $\{u_i | i \in \mathbb{Z}\}$,

$$C_1 \sum_i |u_i|^2 \leq || \sum_i u_i \phi_i ||_H^2 \leq C_2 \sum_i |u_i|^2 \; .$$

We associate with every space $V^{(p)}$ a Riesz basis $\{\phi_i^{(p)}\}$ consisting of *scaling functions*, likewise with every detail space $D^{(q)}$ a Riesz basis $\{\psi_i^{(q)}\}$ consisting of *detail functions*[5]. For some applications, it is convenient to have every level-q detail, $\psi_i^{(q)}$, orthogonal to every level-q scaling function, $\phi_j^{(q)}$, however in general *orthogonality is not essential*[6].

In the next chapter we will examine various practical discretization schemes (among them subdivision schemes, wavelets, and multiscaling functions) which naturally give us a hierarchy of nested spaces $V^{(p)}$. Any of these discretization schemes are fertile ground for applying our approach.

[4]This approach can be generalized to the broad *discrete framework* introduced by Harten in 1993. This framework is summarized in Section 2.5

[5]In some special settings, $\{\psi_i^{(p)}\} \subset \{\phi_j^{(p+1)}\}$, i.e., the level-$p$ details are level-$p+1$ scaling-functions. Here one might be eager to blur the lines between scaling-functions and details; do not succumb to this temptation.

[6]A stable basis can be orthogonalized in a shift-invariant way [Strang and Fix 1973]. However, the orthogonalized basis may be less convenient; hence our adoption of the weaker Riesz requirement.

2.2.2 Natural Refinement

Suppose that initially the basis functions are the *coarsest-level* scaling functions $\mathcal{B}_0 := \{\phi_i^{(0)}\}$, i.e., $S_0 := \mathrm{Span}(\mathcal{B}_0) = V^{(0)}$.

To refine S_0, we might choose as the new set of basis functions the level-1 scaling functions $\{\phi_i^{(1)}\}$; this satisfies our definition of refinement, i.e., $S_1 := V^{(1)} \supset V^{(0)} = S_0$.

However, we wish to make the refinement process as gradual as possible—each refinement step should enlarge the approximation space but only a little bit.

This is critical. Gradual control of refinement, or *adaptive* refinement, leads to improved convergence under fixed computational budget.

There are two natural ways to make a gradual transition to the finer space $V^{(1)}$: *augmenting* $\mathcal{B}_0$ with a *detail* function, or *substituting* in $\mathcal{B}_0$ several level-1 *scaling* functions in place of one level-0 scaling function.

Augmentation with Details We can refine S_0 by introducing to its spanning set a single detail function, i.e., $\mathcal{B}_1 := \{\phi_1^{(0)}, \ldots, \phi_N^{(0)}, \psi_j^{(0)}\}$ for some chosen index j.

In general, we may refine some S_n with the detail $\psi_j^{(p)} \notin S_n$, forming the space S_{n+1} spanned by $\mathcal{B}_{n+1} := \mathcal{B}_n \cup \{\psi_j^{(p)}\}$.

Substitution With Scaling Functions Another approach refines S_0 using only scaling functions: start with $\mathcal{B}_0$, remove a particular level-0 scaling function $\phi_j^{(0)} \in \mathcal{B}_0$, and replace it with just enough level-1 scaling functions such that the new space contains S_0. Which functions of $V^{(1)}$ are necessary to ensure that $S_1 \supset S_0$? The key is the *nesting* of the spaces $V^{(n)}$: since $V^{(p)} \subset V^{(p+1)}$, any level-$n$ scaling function can be uniquely expressed in the level-$(n+1)$ basis:

$$\phi_j^{(p)} = \sum a_{jk}^{(p)} \phi_k^{(p+1)} \, . \tag{2.1}$$

This is the *refinement relation* between a *parent* $\phi_j^{(p)}$ and its children $\mathcal{C}(\phi_j^{(p)}) := \{\phi_k^{(p+1)} | k \in \mathbb{Z} \wedge a_{jk}^{(p)} \neq 0\}$. Note that in general every function has multiple children—it also has *multiple* parents given by the adjoint relation $\mathcal{C}^\star(\cdot)$.

Responding to our question above: if we remove $\phi_k^{(1)}$, then we add $\mathcal{C}(\phi_k^{(1)})$, so that the refined space is spanned by $\mathcal{B}_1 := \mathcal{B}_0 \backslash \phi_j^{(0)} \cup \mathcal{C}(\phi_j^{(0)})$.

In general, we refine some S_n by substituting some scaling function, $\phi_j^{(p)} \in \mathcal{B}_n$, by its children, forming the space S_{n+1} spanned $\mathcal{B}_{n+1} := \mathcal{B}_n \backslash \phi_j^{(p)} \cup \mathcal{C}(\phi_j^{(p)})$.

Both the augmentation and substitution operations are atomic, i.e., they cannot be split into smaller, more gradual refinement operations.

2.2.3 Bookkeeping

With repeated application of these atomic operations we can gradually effect a change of trial space. As a concrete example consider the transition from $S_0 = V^{(0)}$ to $S_M = V^{(1)}$ via application of M augmentation operations. Each refinement introduces a detail chosen from $D^{(0)}$, and after $M = \text{Dim}(D^{(0)}) = \text{Dim}(V^{(1)}) - \text{Dim}(V^{(0)})$ steps we have $S_M = V^{(0)} \oplus D^{(0)} = V^{(1)}$. Consider instead the same transition effected via repeated substitution operations. Each step replaces a scaling function of $V^{(0)}$ by its children in $V^{(1)}$; in this case at most $\text{Dim}(V^{(0)})$ steps are required.[7]

At any stage n of the weighted residual method the approximation space S_n is spanned by the *active functions*,

$$\mathcal{B}_n \subset \bigcup_p V^{(p)} \cup D^{(p)} \, ,$$

i.e., $S_n = \left\{ \sum_{\varphi_i \in \mathcal{B}_n} u_i \varphi_i \right\}$. We will refer to a scaling or detail function as *active* whenever it is in $\mathcal{B}$. With that we can succinctly summarize the two kinds of refinement:

detail-refinement activates a single detail function;

substitution-refinement deactivates a scaling function and activates its children.

Note that in general S_n is spanned by active functions from multiple (possibly not consecutive!) levels of the nesting hierarchy. Further refinements of S_n may introduce details at *any* level of the hierarchy. Substitution refinements always replace an active function by functions from the next-finer level, however one may apply substitution recursively, replacing a function by its grandchildren.

2.2.4 Compact Support Gives Multiresolution

The theory and algorithms presented herein do not make assumptions about the parametric support of the scaling- and detail-functions. Despite this, our discussions will focus on refinement schemes that give rise to basis functions obeying two properties:

Compact Support every scaling- or detail-function f has an associated radius $\epsilon(f)$, such that its parametric support is contained in some $\epsilon(f)$-ball $Q_\epsilon \subset \Omega$ of the domain, i.e., $\text{Supp}(f) \subset Q_\epsilon$, and

Diminishing Support there is a single constant $K < 1$ such that for each scaling- or detail-function f the support of every child $g \in \mathcal{C}(f)$ is geometrically smaller, i.e., $\text{Supp}(g) \leq K\text{Supp}(f)$. Thus for every level-$p$ scaling function, ϕ_i^p, the support of all level-q descendants is bounded by $K^{(q-p)}\text{Supp}(\phi_i^p)$.

Combined, these two properties imply that (i) a refinement operation does not affect the approximation space except in some (parametrically-)localized region of the domain, and (ii) inside that region the *resolution*

[7]For some refinement schemes, the parent-children structure is such that replacing some but not all level-0 functions results in the introduction of all level-1 functions. In this case less than $\text{Dim}V^{(0)}$ steps are required.

of the approximation space is enhanced. Property (i) follows from the compact support, and (ii) follows from the diminishing support. Together these properties recast adaptive refinement as a means to selectively and locally enhance the resolution of the approximation space. Concretely, the refinement directly from V_0 to V_1 is refinement over the entire domain, whereas the gradual refinement using details or substitution (parametrically-)locally-delimited steps.

Scaling- and detail-functions with compact and diminishing support form a *multiresolution analysis*: the solution u may be decomposed into parametrically- and spectrally-localized components. Put in the context of a physical simulation, these two kinds of locality give each scaling function a physically-intuitive role (e.g., representing oscillations at some frequency band in a specific piece of the material) and give rise to techniques for choosing specific refinements over others. For more on multiresolution analyses the interested reader may refer to the text by Albert Cohen [2003a].

For the remainder of this thesis, we will make references to *local* refinement, i.e., refinement in the context of a multiresolution analysis. Although the theory and algorithms do not require compact or diminishing support[8][9], discussions in the context of a multiresolution analysis are more intuitive and lead to pragmatic observations. All of the applications that we implemented (see Chapter 5) use a multiresolution analysis.

2.3 Integration

The weighted residual approximation of our variational formulation requires our finding in the trial space S_n the minimizing function $P_n u(x)$. For this reason among others, we must have the facility to integrate over the domain integrands involving functions from the trial space. In our discussion we will focus on integrands involving only functions from the trial space S_n. Some useful formulations require integrands involving both functions from a trial space S_n and from a *test* space T_n; the framework presented here is easily extended to such settings.

2.3.1 Domain Elements

The trial space is in general spanned by functions from several levels of the nesting hierarchy $V^{(p)}$. The compact support of these functions permits an economical means to integration using the following construction: to every level of the nesting hierarchy we associate a partition of the domain, i.e., a set of subdomains (=*elements*) which together comprise the entire domain; progressively finer levels have finer (and more nu-

[8]The exception is our algorithm for building a basis given a sequence of nested subdomains (see Section 2.4.2)—the problem addressed by that algorithm inherently tied to locality in the parametric domain.

[9] From a theoretical point of view, many of the theorems of multiresolution can be proved (albeit not as easily) without reference to *compact support*, so long as a notion of *locality* still holds [Cohen 2003a, Cohen 2003b], i.e., the basis function, $\varphi_i(x)$, is localized about $x_i \in \Omega$ whenever we have an estimate of the type

$$|\varphi_i(x)| \leq C_m(1 + |x - x_i|)^{-m} , \qquad \forall\, m > 0 .$$

However, from a computational perspective, *compact support* leads (for our algorithms) to significantly better performance than a weaker notion of locality.

merous) elements. Given a set of active functions spanning the trial space, we assemble an integration scheme gathering appropriate elements from the various levels (see Figure 2.1).

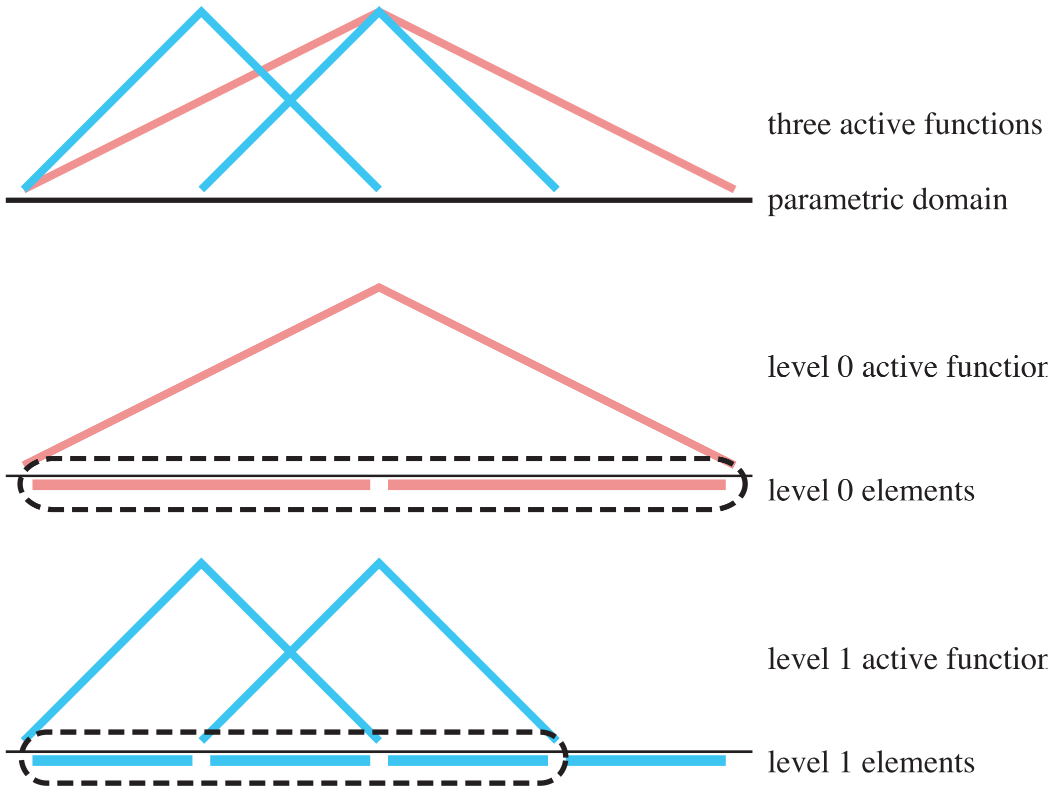


Figure 2.1: Given a set of active functions (top row, hat functions), for numerical quadrature the domain (top row, horizontal line) must be partitioned into elements. We partition the domain into progressively finer *elements* associated to the nested spaces (middle and bottom rows, horizontal line segments). Given a set of active functions spanning the trial space, we assemble an integration scheme gathering appropriate elements from the various levels. The dashed lines encircle the elements gathered to intergrate bilinear forms given these three active functions.

A (disjoint-)partition of a domain is a set of (subdomain) elements whose disjoint union is the domain. To every nested space $V^{(p)}$ we associate a partition of the domain into (the level-p) elements, $E^{(p)} = \{\varepsilon_i^{(p)} | 0 < i \leq N\}$.

Our partitions are chosen by construction to have the following critical property: every function in $V^{(p)}$ has a *simple form* over every element. A simple form is one that is easy to (approximately or exactly) integrate over the element, e.g., the functions are piece-wise cubic polynomials over the elements—the restriction of a function over any element is a piece of a cubic polynomial. It is sufficient that every scaling function, $\phi_i^{(p)}$, has a simple form over every same-level element, $\varepsilon_j^{(p)}$. Then every function in $V^{(p)}$ will have a simple form over the same-level elements. Furthermore, since the level-p details, $D^{(p)} \subset V^{(p+1)}$, are spanned by the level-$(p+1)$ scaling functions, every detail, $\psi_i^{(p+1)}$, has a simple form over every immediately-finer element,

$\varepsilon_j^{(p+1)}$.

The *natural support set,* $\mathcal{S}(\phi_i^{(p)})$, of a scaling function is the minimal set of level-p elements which contain the parametric support of the function[10]. Similarly, the natural support set, $\mathcal{S}(\psi_i^{(p)})$, of a detail function is the minimal set of level-$(p+1)$, i.e., *immediately-finer,* elements which contain the parametric support of the detail. In integrating a basis function, its natural support set is the coarsest valid partition at our disposal: (a) by construction of the level-m domain elements, $\phi_i^{(p)}$ is simple over every member of $\mathcal{S}(\phi_i^{(p)})$, and (b) its support is fully contained in $\mathcal{S}(\phi_i^{(p)})$. The adjoint operation $\mathcal{S}^\star(\varepsilon_j^{(p)})$ returns the set of basis functions whose natural support contains the element $\varepsilon_j^{(p)}$.

The *descendants of an element,* $\mathcal{D}(\varepsilon_i^{(p)})$, are all elements at levels finer than m which in the parametric domain have non-zero intersection with the given element. The *ancestor* relation is defined through the adjoint, $\mathcal{D}^\star(\varepsilon_i^{(p)})$.

To numerically resolve the active functions we gather the appropriate elements. We define the set of *active elements,* $\mathcal{E} = \bigcup_{f\in\mathcal{B}} \mathcal{S}(f)$, as the union of the natural support sets of the active functions.

2.3.2 Domain Tiles

In general the integration must be carried out over a partition of the domain. Since trial space has functions from different levels, we need a partition that will resolve these different resolutions (see Figure 2.2). We construct a minimal set of *active tiles* such that (i) the tiles partition the domain, and (ii) the partition has sufficient resolution: every active element is a disjoint union of tiles. Consequently every basis function has a simple form over every tile. To carry out the integration, we consider every basis function only on the active tiles which partition its natural support set, i.e., over every tile we consider only those basis functions whose parametric support overlaps the tile.

We have two kinds of tiles. The *element tiles,* associated to every level p, are the level-p elements. The *resolving tiles,* associated to every level p, form the minimal set of tiles which (i) partitions the domain, (ii) contains a unique partition for every level-p element, and (iii) contains a unique partition for every level-$(p+1)$ element (see Figure 2.3). Observe that the intersection between any level-p tile and any level-$p+1$ element can be expressed (*resolved*) as a disjoint union of level-p resolving tiles[11]. With this construction in place, a simple and general algorithm can translate a set of active elements into an optimal domain tiling.

Links We need words to refer to the relationship between a level-p element tile and its overlapping level-$(p-1)$ and level-p resolving tiles, i.e., a child-parent relationship. We already have a word (*descendant*) to describe finer-level elements which overlap a given element. To avoid abiguity, when we describe relationships between tiles we will use the terms *finer link* and *coarser link.* Tiles of one type (*element* or *resolving*)

[10]For some schemes, but not all, a function's support is the disjoint union of its natural support set. In other exotic but practical settings this is not the case.

[11]In the special case where the element partitions of levels $1, 2, \ldots$ are nested (i.e., whenever the level-$(p+1)$ elements contain a unique partition for every level-p element), the level-p resolving tiles are by construction the level-$(p+1)$ elements. The resolving tiles figure in more exotic settings such as the $\sqrt{3}$-Subdivision Scheme.

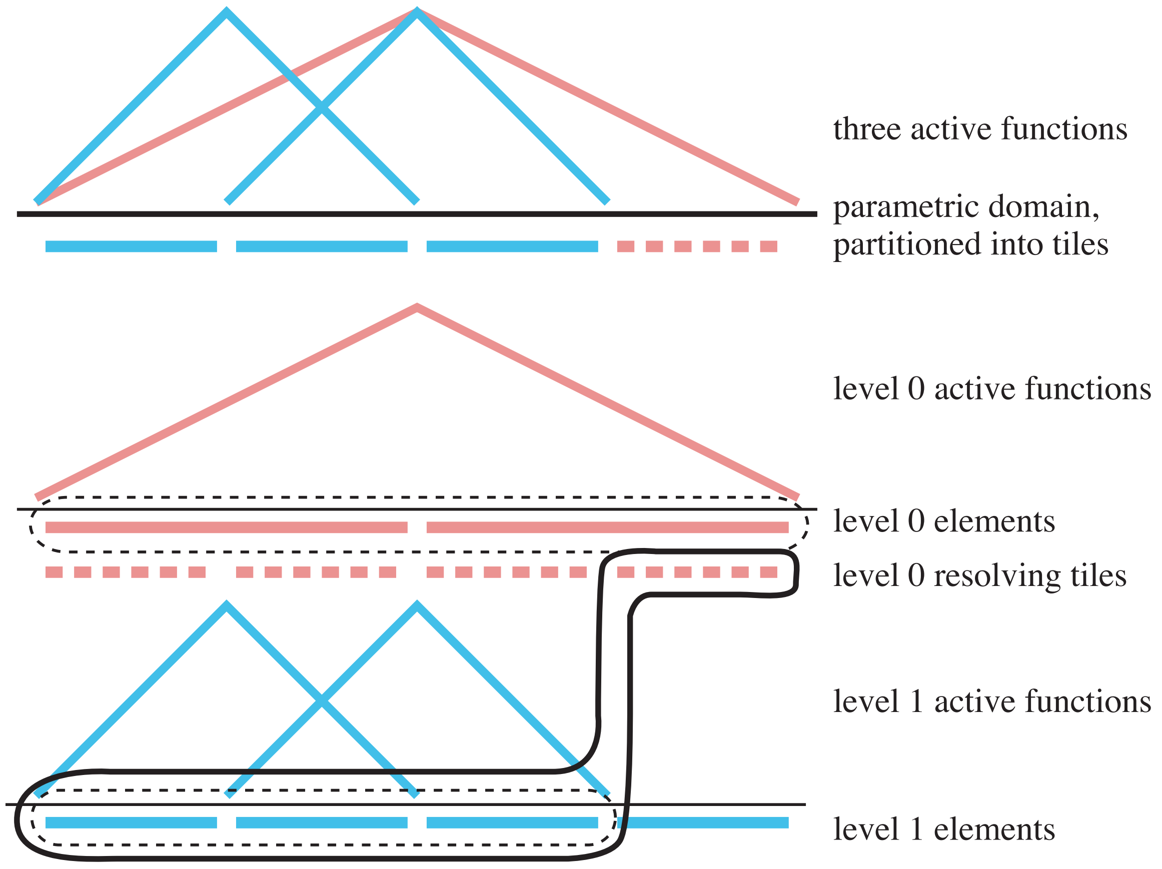


Figure 2.2: The domain is partitioned into a minimal set of *active tiles*, chosen from a set of *element tiles* (solid) and *resolving tiles* (dashed). A *partition* made of *tiles* (unlike a *set of elements*) must be disjoint, i.e., the tiles must not overlap. The partition, comprised of element and resolving tiles, must be sufficiently and necessarily fine to resolve every active function. For the three active functions, we have indicated the appropriate partition by encircling elements and resolving tiles with a black curve; the resulting partition is reproduced on the top row.

are only linked to tiles of the *other type* (see Figure 2.3):

- The finer-link, $\mathcal{L}(t)$, of resolving-tile, t, is the single overlapping element-tile at the next-finer level.
- The adjoint relationship gives the coarser-link, $\mathcal{L}^*(t)$, i.e., the single overlapping element tile at the same level as t.
- $\mathcal{L}(\varepsilon)$ is the finer-link of ε, i.e., the set of *potentially multiple* overlapping resolving tiles at the same level as ε.
- $\mathcal{L}^*(\varepsilon)$ is the coarser-link of ε, i.e., the set of overlapping resolving tiles at the next-coarser level from ε.

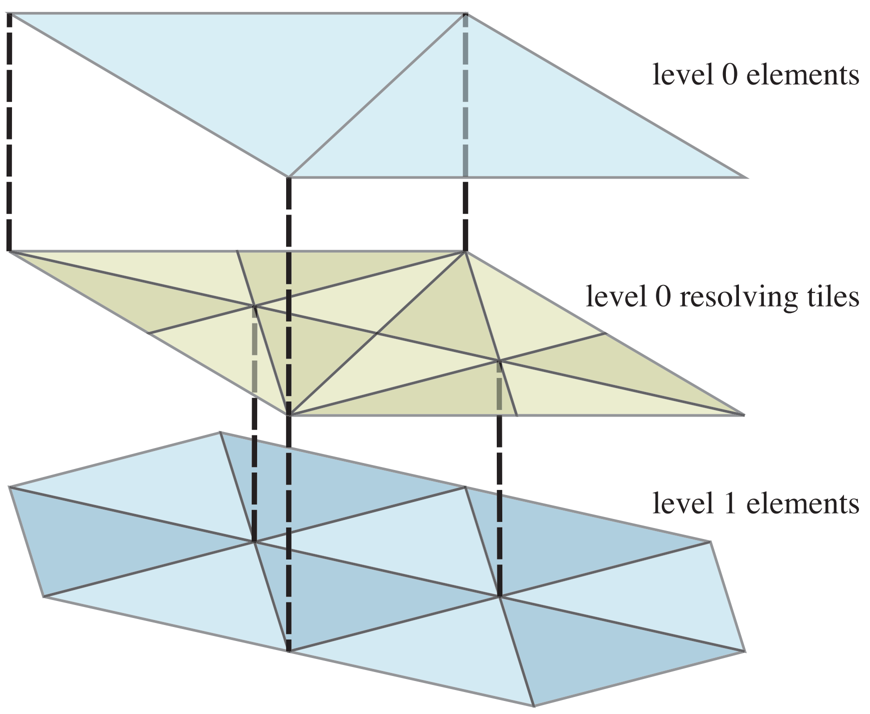


Figure 2.3: There are two kinds of tiles. *Element tiles* (top and bottom rows) are simply elements. Their new name serves as a reminder that those chosen to be active should not overlap. *Resolving tiles* (middle row) for level-p live “in between” level-p and level-$(p+1)$: they exist to resolve gaps between regions partitioned with level-p tiles and others partitioned with level-$(p+1)$ tiles. Note that in general, the level-1 elements are *not* nested in the level-0 elements: this is depicted here, using the triangle meshes of $\sqrt{3}$ subdivision.
Links are parent/child relations between element tiles and resolving tiles. Every level-p element tile (one triangle in solid blue) is the disjoint union of some level-p resolving tiles (the *finer link* of the blue triangle are the two triangles in solid red). Furthermore, every level-p resolving tile overlaps exactly one level-(p) element tile (the *coarser link* of a solid red triangle is the solid blue triangle).

2.3.3 Multiple Approximation Spaces

In some formulations (such as the Galerkin method) the integrand involves functions from two or more approximation spaces. We do not address these formulations in detail. However we claim that the structures constructed herein accommodate such settings. Briefly:

Multiple approximation spaces, single nested-space sequence If several approximation spaces, constructed from the same nested-spaces structure, appear simultaneously in the integrand, our theory and algorithms apply without modification.

Multiple approximation spaces, multiple nested-space sequences If the approximation spaces appearing in the integrand are constructed from various nested-space structures, there are two options. First, one can choose the domain elements such that they give simple forms for all the spaces; from there the theory and algorithms apply without modification.

Alternatively, one can use different sets of domain elements for each of the nested-spaces structures, and introduce new kinds of resolving tiles to resolve intersections between tiles and elements of the different spaces, following closely the approach taken above for resolving intersections between level-p tiles and level-$(p+1)$ elements.

2.3.4 Bilinear Forms

In the special case where the integrand is a bilinear form $a(\cdot,\cdot)$ with arguments $v(x), w(x)$ chosen from the same trial space (e.g., this is the case for linear Galerkin formulations), we can forgo tiling the domain and integrate directly over the active elements. In this case it does not matter that the integration is not carried out strictly over a partition of the domain—although they always cover the domain, in general *some active elements overlap*. Since $v(x) = \sum_i v_i f_i(x)$ and $w(x) = \sum_j w_j f_j(x)$ the integrand is bilinear, we rewrite our integral as a double summation of integrals of pairs of particular basis functions: $\sum_i \sum_j v_i w_j \int a(f_i, f_j)$. Each of these integrals considers interactions between a pair of basis functions and it is carried out *at the coarsest level that captures the interaction:* if coarser function f_1 overlaps finer function f_2, we evaluate the bilinear form over cells in the natural support set of f_2 which also support f_1: $\{\varepsilon \,|\, \varepsilon \in \mathcal{S}(f_2) \wedge \mathcal{D}^\star(\varepsilon) \cap \mathcal{S}(f_1) \neq \emptyset\}$. With this approach every interaction is considered exactly once, at a necessarily and sufficiently fine resolution. Please note: here the integration is *not* carried out over a partition of the domain! Although they always cover the domain, and although the active elements are chosen from partitions, $E^{(p)}$, of the domain, in general *the active elements overlap*, with elements from different levels covering overlapping subdomains.

2.4 Basis

In some settings, it is important that the active functions are linearly independent, i.e., they from a basis for the trial space.[12]Here we lay out additional structure that, as we see in the next Chapter, paves the way to efficient algorithms for ensuring that the active functions are linearly independent.

Our discussion of linear-dependence differs from the preceding sections of this chapter in that *here we assume compact and diminishing support* and rely heavily on this assumption in developing efficient basis-maintenance algorithms. It is possible to lift this assumption by reducing the linear-independence problem to a linear-algebra matrix problem. However, if this assumption is not replaced by another, then resulting linear-algebra problem is computationally expensive.

[12]This is the case, for example, in classical FE applications, as a linear dependency in the basis leads to a singular stiffness matrix.

2.4.1 Refinement With Details

When we restrict ourselves to only refine by adding details, the active set of basis functions is always, by construction, a basis for the trial space. To prove this, consider the following. Since the spaces, $V^{(0)}, D^{(0)}, D^{(1)}, \ldots, D^{(M)}$, the union of their bases, the so-called *multiresolution analysis*, is the basis of $H^M := V^{(0)} \oplus D^{(0)} \oplus D^{(1)} \oplus \cdots \oplus D^{(M)}$. The active set consists of the coarsest-level scaling functions and details from the first M levels (for some M), i.e., the active set is a subset of the basis of H^M, consequently it is also linearly independent.

2.4.2 Refinement by Substitution

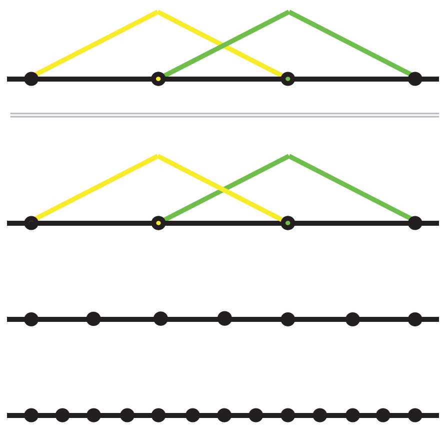
Instead of working with detail functions, we might work with bases built of scaling functions, i.e.choosing for the active set functions from each of the bases of $V^{(0)}, V^{(1)}, \ldots, V^{(M)}$. We are motivated to use such bases because they often lead to increased system sparsity thus better performance. This is because subdivision refinement produces *locally single-resolution* bases. Consider, for instance, starting with the trial space $S_0 = V^{(0)}$, and refining by substitution everywhere over the domain. The resulting trial space, $S_1 = V^{(1)}$, is single-resolution, i.e., choose any point on the domain; only level-1 functions overlap this point. In contrast, if $S_0 = V^{(0)}$ is refined by adding the level-1 details, every point on the domain is overlapped by level-0 as possibly level-1 functions. Typically, we will not refine the domain everywhere to the same depth (that defeats adaptivity!), however, to the extent that the error indicator is in some neighborhood smooth, then over that neighborhood substitution-refinement produces a single resolution of basis functions. The locally single-resolution property of substitution-refinement is often advantageous for computational performance.

In the case of detail functions, we argued that the basis functions are linearly independent because the spaces (of the multiresolution analysis) are disjoint. Here the spaces, $V^{(p)} \subset V^{(p+)}$, are nested *not* disjoint; linear independence is not guaranteed.

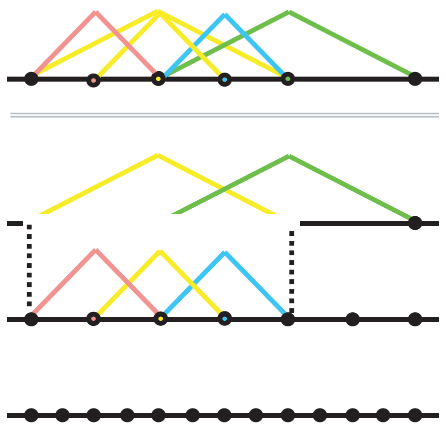
Suppose that we wish to refine by substituting scaling functions with their children. In general this will *not* produce a linearly independent active set. Here we describe additional structure, introduced previously by Kraft in the context of multilevel B-splines, that allows us to easily build a linearly independent active set given a sequence of substitution refinements [Kraft 1997]. This approach may be generalized to refinements with a mix of substitution and details.

We can summarize Kraft's construction very simply in our framework, and consequently implement it easily and efficiently: after every substitution refinement, *we automatically refine by substitution every function with natural support set covered by the active elements from finer levels* (see Figure 2.4). It is possible to show that this iterative process eventually halts.

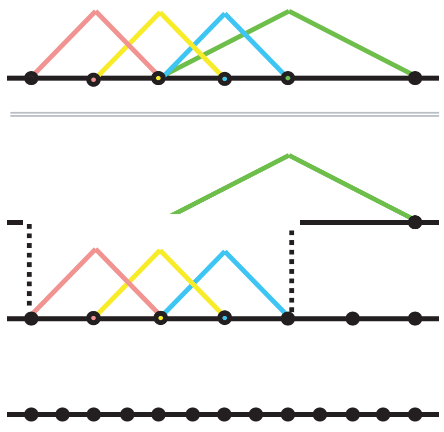
Kraft's construction is as follows. We are given a list of indices of scaling functions to be refined by substitution, grouped per level: $\mathcal{R}^{(0)}, \mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(M)}$. Such a sequence makes sense only if every finer-level function that we are asked to refine was introduced at some earlier point when one of its parents was refined,



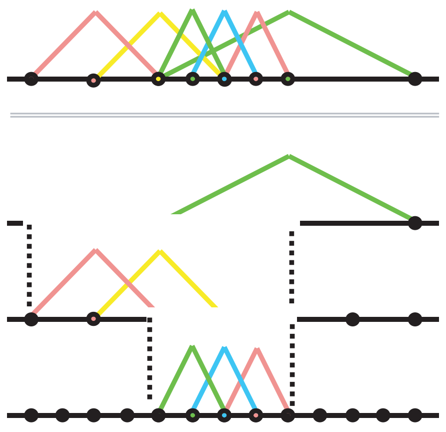
(a)



(b)



(c)



(d)

Figure 2.4: Kraft developed a construction for linearly-independent B-splines. We interpret his construction within our framework as follows: every substitution refinement consists of two steps: first, we activate the children of the basis function (a→b); second, we automatically refine by substitution every function with natural support set covered by the active elements from finer levels (b→c). These two steps, combined (a→c, likewise c→d), form a substitution refinement operation that preserves the linear independence of the basis.

i.e., $i \in \mathcal{R}^{(p+1)} \Rightarrow \exists j \in \mathcal{R}^{(p)} \wedge \varphi_i^{(p+1)} \in \mathcal{C}(\phi_j^{(p)})$

From this list of substitution refinements we construct a sequence of nested subdomains, $\Omega := \Omega^{(0)} \subseteq \Omega^{(1)} \subseteq \cdots \subseteq \Omega^{(M+1)}$, where every subdomain contains exactly the supports of the immediately-coarser refined functions,

$$\Omega^{(p+1)} = \bigcup_{j \in \mathcal{R}^{(p)}} \mathrm{Supp}(\phi_j^{(p)}) \, , \qquad p \geq 0 \, .$$

We interpret the level-p subdomain as follows: inside $\Omega^{(p)}$ we wish to approximate with at least the resolution of space $V^{(p)}$, and outside with only coarser resolutions.

More precisely, we will choose those level-p scaling functions which are supported completely over $\Omega^{(p)}$ and at least partially over $\Omega^{(p)} \backslash \Omega^{(p+1)}$:

$$I^{(p)} := \{k \in \mathbb{Z} | \mathrm{Supp}(\phi_k^{(p)}) \subseteq \Omega^{(p)} \wedge \mathrm{Supp}(\phi_k^{p}) \nsubseteq \Omega^{(p+1)}\} \, .$$

In other words, we throw away every level-p function that is not completely in the designated zone for level p, $\Omega^{(p)}$; and we throw away every level-p function that can be fully represented by level-$p+1$ functions, i.e., it is completely inside the designated zone for level $p+1$, $\Omega^{(p+1)}$.

The resulting multilevel, hierarchical space $S := \mathrm{Span}\{\phi_k^{(p)} | p \geq 0 \wedge k \in I^{(p)}\}$. It is multilevel in that is can appear locally-flat, in regions where the refinement has been applied uniformly, while at the same time functions from different levels participate. It is hierarchical, over regions where the refinement depth is not uniform, in that functions from different levels overlap at a point.

One can give a "physical" interpretation to Kraft's original description of this construction (let us reinterpret Figure 2.4). Kraft begins with a sequence of nested subdomains, $\Omega := \Omega^{(0)} \subseteq \Omega^{(1)} \subseteq \cdots \subseteq \Omega^{(M+1)}$, corresponding to regions of desired minimum detail at each level of the nesting hierarchy. Stack all the domains, with the coarsest at the top (Figure 2.4d). Every domain creates a hole in those above it. Thus every domain appears as a region—with holes induced by the domains below. Start with all scaling functions at the coarsest level (Figure 2.4a), and let them fall "downward under gravity" (Figure 2.4, a→c, c→d). A function stops falling if it can't fit completely though a hole. Every time a function falls to the level below, it splits into its children (Figure 2.4b), and they continue the fall.

2.5 Harten's Discrete Framework

Many popular discretizations, among them point samples, finite-volume/average-samples, and finite-elements, after being extended to an adaptive, multi-scale setting, may be viewed as specific instances of a *discrete framework*, introduced by Harten [1993, 1996], which generalizes our nested spaces structure.

Definition A a *generalized discrete multiresolution approximation* is[13]

[13]We take the liberty to adapt Harten's notation to more clearly delineate the parallels between our frameworks.

1. a sequence of index sets $\Gamma^{(p)}$, $p \geq 0$,
2. equipped with a *restriction operator*, $P^{(p)}_{(p+1)}$, acting from $l^\infty(\Gamma^{(p+1)})$ to $l^\infty(\Gamma^{(p)})$,
3. and a *prolongation operator*, $P^{(p+1)}_{(p)}$, acting from $l^\infty(\Gamma^{(p)})$ to $l^\infty(\Gamma^{(p+1)})$,
4. such that the restriction operator is a left inverse to the prolongation operator, i.e., $P^{(p)}_{(p+1)} \circ P^{(p+1)}_{(p)} = I$.

Intuitively, the sequence of (discrete) index sets, $\Gamma^{(p)}$, is similar to our concept of a sequence of scaling-function bases; the prolongation operator is similar to a refinement relation mapping one level-p scaling function to its level-$(p+1)$ children, and condition (4) corresponds to a nesting of the spaces, $V^{(p)} \subseteq V^{(p+1)}$.

It is straightforward to derive for this framework the details between consecutive levels, $\Gamma^{(p)}$, and consequently a multiscale decomposition of a function into a coarse representation and multiple levels of details. Similarly to our observation in Section 2.2.4 that efficient implementations make use of assumptions of *locality*, Cohen notes that for efficiency Harten's framework must be specialized with notions of locality [Cohen 2003a].

It would be worthwhile to fully explore the connections between Harten's discrete framework and our framework.

2.6 Summary and Preview

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 refinement 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 *buffer 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].

In Chapter 4 we will explore these issues. First, however, we illustrate the ideas of nested spaces and basis refinement using popular as well as recently-introduced approaches to discretizing PDEs.

Chapter 3

Discretization Zoo:
Constructions for Nested Spaces

We build a trial space by activating scaling and detail function from the nested spaces; we evaluate integrals using elements and tiles. As a whole these structures—scaling functions, details, elements, and tiles—form a *multiresolution discretization.* Until now we have presented these ideas *in general*. Now we are concrete. We describe well-established as well as more recent discretizations (finite- or spectral-elements, wavelets, etc.), and discuss the properties which characterize them (symmetry, locality, etc.). We examine several discretizations in more depth, mapping them onto the multiresolution structures of Chapter 2.

3.1 Overview

We discuss only discretizations constructed explicitly from basis functions, i.e., the unknown function $u(x)$ is approximated by a finite linear combination, $\sum u_i \phi_i(x)$, where the basis functions $\phi_i(x)$ are a subset of some larger (infinite) set of scaling and detail functions which are determined *a priori*. Particular discretizations use particular families of functions, e.g.,

spectral discretizations	polynomials or sinusoids
finite element discretizations	piecewise polynomials
spline discretizations	spline basis functions
subdivision discretizations	subdivision basis functions
wavelet discretizations	scaling functions and wavelets

There are many possible discretizations. Each has strengths and weaknesses; each is appropriate for some problems and not others. How to choose one? Consider the ideal characteristics, *for a particular application*, of the discretization. No discretization is ideal: trading one desirable property for another is part of the game.

3.1.1 Characterizing Discretizations

A discretization is characterized by many properties. These are the most important:

Single- vs. Multi-resolution A discretization may display the data in *single*- or *multi*-resolution, e.g., so-called *nodal* finite-element bases display data at a single resolution, whereas wavelets display data at multi-resolution. Some single-resolution discretizations can be easily generalized to multi-resolution, perhaps in more than one way, e.g., nodal finite-elements generalize to hierarchical finite-elements or to multiscaling bases. Single-resolution methods may suffice for applications which don't require (the implementation and computational overhead of) adaptivity.

Coefficients Characterize the coefficients of a discretization with a qualitative description, e.g., the coefficients, u_i, control: the spectral components, $\hat{u}(w_i)$, as in Fourier's method; or, the function values, $u(x_i)$, at specific points, x_i, as in interpolating discretizations; or, the derivatives, $Du(x_i)$, at specified points, as in Hermite splines [Foley et al. 1995]; or, the mean value of the function, $\|\Omega_i\|^{-1} \int_{\Omega_i} u(x)dx$, over specific subdomains, Ω_i, as in Finite Volumes [Versteeg and Malalasekera 1996]. Some schemes have more than one kind of coefficient: Hermite splines have coefficients, $u_1, \ldots, u_N$ and $v_1, \ldots, v_N$, which control function values and derivatives respectively.

In the context of a particular physical PDE the coefficients carry physical meanings. Function values are displacements in elasticity, and temperatures in thermal conduction. Derivatives are strains and forces, and thermal gradients. Engineers choose discretizations which carry meaningful and control-able coefficients.

Elasticity practitioners prefer discretizations with displacements. The most easily understood (but not always most desirable!) set of coefficients belongs to interpolating discretizations.

Interpolation In an *interpolating discretization* the coefficients, $u_1, \ldots, u_N$, associate to domain points, $x_i \in \Omega$, and thereat fix the function value, i.e., $u(x_i) := u_i$. Some interpolating discretizations also carry other coefficients, $v_1, \ldots, v_N$, which may for instance specify the derivatives at the points, x_i. Interpolation comes when the basis functions satisfy the *Kronecker Delta property*, i.e., $\varphi_i(x_j) := \delta_{ij}$.

In contrast, an *approximating discretization* does not have the Kronecker Delta property and does not (in general) interpolate its coefficients. Generally, interpolating methods introduce undesirable oscillations into the approximate solution, $P_N u(x)$, whereas approximating approaches tend to avoid this.

Locality A method has *parametric locality* (or *spatial locality*) if it associates the finite-basis coefficients, u_i, to localized regions of the domain, Ω. For example, point-sampling methods associate the u_i to points $x_i \in \Omega$; finite-element methods associate the u_i to mesh nodes $x_i \in \Omega$ and to their incident mesh elements; meshless methods associate the u_i to (e.g., radial basis-)functions centered at $x_i \in \Omega$. In all these examples, the u_i inherit the locality of their associated basis functions (here we view point-sampling as a finite-basis discretization using as basis functions the Dirac distributions centered at x_i). A basis function, $\varphi_i(x)$, is localized about $x_i \in \Omega$ whenever we have an estimate of the type

$$|\varphi_i(x)| \leq C_m(1 + |x - x_i|)^{-m} , \qquad \forall\, m > 0 .$$

Similarly a method has *spectral locality* if it associates the u_i to localized bands of the spectral domain, e.g., Fourier methods associate every u_i to a single frequency. The Heisenberg principle states that we must trade spatial- for spectral-locality. Wavelets carry coefficients with (a compromise between) spatial and spectral locality.

Locality gives efficient algorithms. When it comes in the form of multiresolution, it also gives sparse, spatially- or spectrally-*adaptive*, representations of multiscale data, with double benefit: first, multiresolution representations of data often provoke novel insight into the modeled phenomena; second, they are computationally economic. The second point cannot be stressed enough: it is the secret to the efficiency of the algorithms in Chapter 4.

Sampling Uniformity A method with parametric locality has *parametrically-uniform sampling* whenever the coefficients, u_i, are localized at equal intervals over the parameter domain, e.g., if the x_i are equidistant from their mesh neighbors. Similarly spectrally-localized coefficients, uniformly distributed over the spectral domain, give *spectrally-uniform sampling*.

Consider, for example, wavelet and subdivision methods (see Sections 3.2 and 3.5 respectively). Both give multiresolution discretizations, with consequent spatial- and spectral-locality at each level of resolution.

Wavelets are typically pursued in the regular setting, i.e., over regular meshes with uniform samples. In contrast subdivision discretizations are usually semiregular and non-uniform.

In some settings uniform sampling introduces additional symmetries with consequent numerical *super*convergence.

Dimension A discretization may be *dimension-specific*, e.g., bivariate quartic box splines are appropriate for *two-dimensional* domains. Alternatively, it is *dimension-independent*, e.g., tensor product quadratic B-splines of dimension D are appropriate for D*-dimensional* domains.

Tessellation A *meshless* method does not tesselate the domain; a *mesh-based* method partitions (an approximation of) the domain into simple pieces, e.g., intervals of the real line in one dimension; rectangles, triangles, etc.. in two dimensions; tetrahedra, hexahedra, etc.. in three dimensions; etc.. By design, mesh-based methods have spatial locality: they associate the finite-basis coefficients, u_i, to mesh entities, e.g., coefficients may "live" at mesh faces, edges, or vertices(=nodes). In contrast, there are meshless methods with and without spatial locality, e.g., Radial Basis Function [Buhmann 2003] and Fourier discretizations [Strang and Fix 1973] respectively.

Some meshes are *refinable tilings*: they can be repeatedly refined into finer, nested, self-similar refinable tilings. Lave proved that $\mathbb{R}^2$ has only eleven refinable tilings; each is isohedral and at each vertex equiangled.

Connectivity Mesh-based methods may be characterized by the incidence relations between mesh elements. In two dimensions, a mesh is *regular* if all like vertices are equivalenced, e.g., a triangle mesh is regular if all interior vertices have valence six and all boundary vertices have valence four; *semireqular* or *quasiregular*[1] if the mesh is regular away from some (small) set of isolated vertices; otherwise the mesh is *irregular*. Regular connectivity invites economy of computation and storage at the expense of flexibility offered by irregular connectivity, e.g., regular connectivity meshes cannot tesselate domains of arbitrary genus; semi-regular connectivity strikes a compromise.

Parametric Support A discretization is characterized by the parametric support, $\mathrm{Supp}(\varphi_i(x)) \subseteq \Omega$, of its basis functions. Is it compact? Is the boundary, $\partial\mathrm{Supp}(\varphi_i(x))$, open or closed? Is the support fractal, i.e., of finite measure but with boundary of unbounded length? What is the diameter of the parametric support? Sometimes the diameter for coefficient u_i is expressed as an n*-ring*[2] around the mesh entity associated to u_i.

Consider, for example, Loop's triangle subdivision and Kobbelt's $\sqrt{3}$ triangle subdivision [Kobbelt 2000a, Loop 1987]. In both schemes, each coefficient, u_i, associates to a vertex, x_i, and the corresponding parametric support of $\varphi_i(x)$ is compact, centered around x_i, and has an open boundary. In Loop's scheme the basis

[1] While less popular, some prefer *quasi* in direct analogy to crystalline structures.

[2] The *one-ring* of faces around a mesh vertex are the mesh faces incident to the vertex. In general, the n-ring of faces around a vertex are the faces in or incident to the $n-1$-ring. One may also refer to the *one-ring* of vertices around a vertex; usually the context makes the writer's intention clear.

functions, $\varphi_i(x)$, are supported on the two-ring of the associated vertex. In Kobbelt's scheme their parametric support is a fractal subset of the two-ring.

For discretizations that define a hierarchy of mesh levels, we may ask whether the support is diminishing (recall Section 2.2.4). Compact diminishing support leads to *controlled overlapping*: a point $x \in \Omega$ is contained in at most $M \in \mathbb{Z}$ supports of basis functions from level p, with M independent of x and p. Controlled overlapping gives a bound on per-coefficient computation and storage.

Smoothness A discretization is characterized by the *smoothness* of its basis functions, equivalently by the smoothness of the generated trial spaces, S_i. There are different measures of smoothness, including C^s (parametric continuity), G^s (geometric continuity), and tangent-plane continuity. The commonest is parametric smoothness: a function is piecewise C^s over a partition of the domain whenever its restriction onto each partition-subdomain has s continuous derivatives [Strang and Fix 1973], e.g., a C^0-everywhere function is continuous over its domain; Loop's basis functions are C^2, except at irregular vertices A particular PDE, coupled with particular numerical method, will rule out discretizations that do not have some sufficient smoothness.

Approximation Power A discretization is characterized by its *accuracy*, i.e., fixing the number, N, of coefficients, $u_1, \ldots, u_N$, what is the approximation error, $\|u(x) - P_N u(x)\|$, for this compared to other discretizations? Typically, accuracy is measured as *polynomial exactness*: a discretization which *locally* reproduces polynomials of order p is said to have *approximation power* p. Write the Taylor series of $u(x)$ about a point, x_i: the polynomial with exponent p is the first to give an error. Expressions of this form are common in specifying the accuracy of a discretization:

$$\|u(x) - P_N u(x)\| \leq C(\Delta x)^p \|\frac{\partial^p u(x)}{\partial x^p}\| \; ,$$

where C is a constant and Δx describes the parametric distance between coefficients, e.g., in one dimension, $\Delta x = \max_{1 \leq i < N}(x_{i+1} - x_i)$. When the details are orthogonal to the scaling functions, the Strang-Fix condition is that *the details have p vanishing moments*. We will say more in Section 3.2.

Local Boundedness A mesh-based discretization satisfies the *convex hull property* whenever the function value over a subdomain, Ω_i, is bounded by neighboring coefficient values, $u_j, \ldots, u_k$, e.g., the value over a Linear Finite Element is bounded by the coefficients incident to the element. An interpolating discretization that has the convex hull property must have $s \leq p \leq 1$, i.e., it cannot have high-order smoothness nor accuracy. This can be a severe limitation for interpolating methods! In contrast, approximating approaches may be smooth, high-order accurate, and have the convex hull property.

Symmetry A discretization may be characterized by the symmetries of its basis functions. For example, the Fourier method's basis functions, $\varphi_i(x) := \sin(ix)$, are invariant under the translations $\cdot - 2\pi k$. In mesh-based methods, symmetries are often tesselation-dependent, e.g., a regular grid with uniform sampling will have translational symmetries which are lost with non-uniform sampling.

Multi-resolution discretizations may have symmetries in scale, e.g., the set of Haar scaling functions (Section 3.2.3), $\cup_p V^{(p)}$, is closed under dyadic scaling, i.e., $\varphi_i(x) \in \cup_p V^{(p)} \Rightarrow \varphi_i(2x) \in \cup_p V^{(p)}$.

The projection operator, P_N, is invariant under the action of the symmetry group of the discretization, i.e., symmetries in $u(x)$ which exist also in the trial space are *preserved* during discretization. For example, the symmetries of a cylindrical surface are preserved by some two-dimensional quadrilateral-tesselations but not triangular-tesselations.

Orthogonality A discretization may have orthogonal basis functions, i.e., $< \varphi_i, \varphi_j > = 0$ whenever $i \neq j$. This gives an obvious space of dual basis functions, which are required for projecting (*analyzing*) into the trial space, i.e., $u(x) \mapsto P_N u(x)$. Having a simple dual space is important for *un*refinement, which uses P_N. Practitioners often use an approximate dual space when the true dual is impractical, as is sometimes (but not always!) the case when orthogonality is lost.

3.1.2 Formulations

Having established a discretization, there are several methods to find the best approximation, $\sum u_i \phi_i(x)$, to the unknown solution, $u(x)$, of the posed PDE. The most popular formulations are:

Collocation [Prenter 1975] Apply the PDE at N points on the domain to formulate a system of N equations with N unknowns, $u_1, \ldots, u_N$, e.g., $\nabla u(x) = b(x),\ x \in \Omega \subset \mathbb{R}^2$ (with appropriate boundary conditions matching the basis functions), becomes

$$\sum_i u_i \nabla \phi(x_j) = b(x_j) \ , \qquad x_j \in \Omega \subset \mathbb{R}^2 \qquad 1 \leq j \leq N \ .$$

Weak Formulation [Strang and Fix 1973, Eriksson et al. 1996] We choose a *test space* defined by a *test basis*, $w_1(x), \ldots, w_N(x)$, and formulate an $N \times N$ system of *weighted residual equations* by taking inner products of the left and right hand sides of the PDE, e.g., $\nabla u(x) = b(x),\ x \in \Omega \subset \mathbb{R}^2$ (with appropriate boundary conditions matching the test and basis functions), becomes

$$\sum_i u_i \int \nabla \phi_i(x) w_j(x) dx = \int b(x) w_j(x) dx \ , \qquad 1 \leq j \leq N \ .$$

Usually the integrals are approximated via numerical quadrature. Note that the weak form simplifies to collocation when the test functions are the Dirac delta distributions, i.e., $w_i(x) = \delta(x - x_i)$.

For clarity, our presentation of basis refinement in Chapter 2 focused on the trial space, however for methods which use separate trial and test spaces the refinement framework may be applied independently to both the trial and the test space.

Variational Formulation [Strang and Fix 1973, Prenter 1975] We express the solution, $u(x)$, as a critical point of a (nonlinear) functional, $E[u(x)]$, and minimize $\|DE\left[\sum_i u_i\phi_i(x)\right]\|$ numerically over the coefficients, u_i, aided by the N partial derivatives,

$$\frac{\partial E\left[\sum_i u_i\phi_i(x)\right]}{\partial u_j}\ , \qquad 1 \leq j \leq N\ .$$

If $E[\cdot]$ is (approximated by a) quadratic in the u_i then the partial derivatives are linear and the search for the (unique) critical point is expressed as an $N \times N$ linear system (again!). Note that the variational formulation is often used for highly non-linear problems; however it is not uncommon to temporarily approximate $E[\cdot]$ by a quadratic form inside the loop of a non-linear solver.

Having formulated a discrete problem, we invoke the appropriate numerical solver. In the case of the linear problem of the collocated, weak, or (linearized) variational formulations, we use a numerical linear-system solver [Press et al. 1993]; otherwise we use the appropriate numerical optimizer or non-linear solver [Press et al. 1993, Eriksson et al. 1996]. Because we have *nested* approximation spaces, a large body of multi-resolution techniques is at our disposal, for instance multigrid solvers [Bank et al. 1988, Wille 1996].

There are other popular methods which do not explicitly approximate $u(x)$ with basis functions. Finite-difference [Strang and Fix 1973] and discrete-operator [Meyer et al. 2003] methods deal directly with point samples, u_i. They do not make explicit the value of the function in between the point samples, although downstream applications often associate (interpolating) basis functions to the coefficients, u_i, e.g., to portray an approximate (piecewise) smooth solution to the user. Here we are concerned only with methods which use finite-basis discretizations; our techniques are not immediately applicable to finite-difference and discrete-operator formulations.

3.1.3 Summary and Preview

No single discretization fits all applications. Some problems inherently involve non-uniform samples, e.g., when the boundary conditions are given non-uniformly. Other problems crucial symmetries which must not be lost, e.g., compressing an elastic cube from two ends gives compressive- but not shear-deformation. Choose the discretization and formulation which best fit the application.

We turn now to various popular discretizations. We examine wavelets, multiwavelets, finite elements, splines and subdivision schemes, and frame each within our framework.

3.2 Wavelets

Wavelets (Fr. *ondelettes*, little waves) emerged in the 1980's as a formalization of multiscale methods developed earlier in the century [Strang and Nguyen 1996b, Cohen 2003a]. Traditionally, the theory of wavelets (e.g., Strang and Nguyen [1996b]) is pursued in the *regular* Euclidean setting, i.e., with scaling- and detail-functions mapping $\Omega \subseteq \mathbb{R}^d$ to $\mathbb{R}$, coupled with a *regular*, e.g., integer lattice, tessellation.

3.2.1 Introduction

Consider the nested spaces $V^{(0)} \subset V^{(1)}$ spanned by translates and dilates, over a regular tessellation of the domain, of a single *scaling function* $\phi(x)$. Furthermore, consider a *wavelet subspace* $D^{(0)}$ with special properties: (1) it completes the coarser space: $V^{(0)} + D^{(0)} = V^{(1)}$; (2) it is spanned by the translates and dilates of a single, localized, zero-mean function called the *mother wavelet*.

More generally, we can have a (possibly infinite) sequence of nested spaces $V^{(0)} \subset V^{(1)} \subset V^{(2)} \subset \ldots$, and associated wavelet subspaces $D^{(k)} \subset V^{(k+1)}$, such that $V^{(k)} + D^{(k)} = V^{(k+1)}$. Consequently, we can express a space $V^{(k)}$ in *multiresolution:*

$$V^{(0)} + D^{(0)} + \ \ldots \ + D^{(k-1)} = V^{(k)} \ ,$$

that is as the sum of a coarse representation and progressively finer details. Because the spaces $V^{(k)}$ are nested, their basis functions must obey refinement relation (2.1). In the wavelets literature this relation is known as the *dilation equation* and it is written out in a form that explicitly shows the dilation and translation; for example Daubechies scaling function satisfies the dilation equation [Daubechies 1992, Strang and Nguyen 1996b]

$$\phi(x) = \sqrt{2} \sum_k a_k \phi(2x - k) \ .$$

3.2.2 Theory

We interpret wavelet theory as a specialization of our (Chapter 2) nested-spaces structure. Traditionally, wavelet theory deals with nested function spaces constructed over regular tessellations and subject to *four additional requirements.* First, the nesting of the spaces is characterized by *scale-invariance*:

Requirement 1: Scale Invariance The spaces are rescaled copies of the coarsest space. $V^{(p+1)}$ consists of all dilated functions in $V^{(p)}$,

$$u(x) \in V^{(p)} \Leftrightarrow u(2x) \in V^{(p+1)} \ .$$

Since the basis for $V^{(p+1)}$ has roughly twice as many functions as that of $V^{(p)}$, then every space and its associated detail space have roughly the same dimensions, i.e., Dim$(V^{(p)}) \approx$ Dim$(D^{(p)})$.

Consider, for example, the sequence of nested spaces formed by progressively adding a single term to the Fourier series. The spaces are nested, but *not* scale invariant. *The frequencies must double.* The Littlewood-Paley decomposition, which splits the Fourier series into frequency octaves, gives nested scale-invariant spaces.

Second, every space is characterized by *shift-invariance*:

Requirement 2: Shift Invariance A function is accompanied by its integer translates,

$$u(x) \in V^{(p)} \Rightarrow u(x-k) \in V^{(p)} , \quad k \in \mathbb{Z} .$$

This means that we must work with the whole line, $-\infty < x < \infty$, or with a periodic interval; alternatively, we may adjust this requirement at the boundaries to allow for finite, non-periodic, intervals.

Combined, scale- and shift-invariance mean that any function, $u(x) \in V^{(0)}$, is accompanied by its dilates, $u(2^p x) \in V^{(p)}$, and their integer translates, $\{u(2^p x - k) | k \in \mathbb{Z}\} \subseteq V^{(p)}$. In particular, we require that a *single* function, accompanied by its translates and dilates, form a stable basis for the nested spaces:

Requirement 3: Canonical Scaling Function There exists a scaling function, $\phi(x)$, with $\{\phi(x-k) | k \in \mathbb{Z}\}$ a Riesz basis for the coarsest space, $V^{(0)}$.

Consequently scale- and shift-invariance give a Riesz basis, $\{\phi_k^{(p)} := 2^{p/2}\phi(2^p x - k) | k \in \mathbb{Z}\}$, for every space $V^{(p)}$. The canonical scaling function obeys a *dilation equation*,

$$\phi(x) = \sqrt{2} \sum_k a_k \phi(2x - k) \tag{3.1}$$

which is just another way of writing our refinement relation,

$$\phi^{(p)}(x) = \sum_k a_k \phi_k^{(p+1)}(x)$$

in this context also called the *two-scale equation* in explicit reference to the scale invariance of the spaces. If (and only if) the scaling function is compactly supported, the set of non-zero coefficients, a_k, is finite. In that case, each scaling function has support of diameter $O(2^{-j})$ and satisfies a property of *controlled overlapping*: a point $x \in \Omega$ is contained in at most $M \in \mathbb{Z}$ supports of basis functions from level p, with M independent of x and p. If (and only if) the scaling function is symmetric about $x = 0$, the coefficients are likewise symmetric, i.e., $a_k = a_{-k}, \; k \in \mathbb{Z}$.

Similarly, the detail spaces are spanned by translates and dilates of the *mother wavelet*:

Requirement 4: Mother Wavelet With Vanishing Moments The detail functions, or *wavelets*, are defined as translated dilates, $\psi_k^{(p)}(x) = \psi(2^p x - k)$, of the *mother wavelet*,

$$\psi(x) = \sqrt{2}\sum_0^N d_k \phi(2x-k) \; , \qquad d_k := (-1)^k a_{(N-k)} \; . \tag{3.2}$$

This is the *wavelet equation.* The level-p details are expressed by combining linearly the level-$(p+1)$ scaling functions with weights d_k, obtained by an *alternating flip* of the refinement relation coefficients, a_k. Wavelets always have integral zero. In general, they have $P > 0$ vanishing moments:

$$\int x^i \psi(x) dx = 0 \; , \qquad 0 \le i \le P \; .$$

Wavelets with P vanishing moments come from scaling functions with approximation order P, i.e., the spaces $V^{(q)}$ reproduce perfectly piecewise polynomials up to degree $p-1$. See also the Strang-Fix condition, Section 3.4.

These four requirements characterize the subclass of nested-spaces structures which are *wavelet systems*. Our theory (Chapter 2) and algorithms (Chapter 4) do not rely on these four characteristics.

There are other (very useful) properties that are interesting but not required. Typically, (anti)symmetries in the shape of a wavelet mean better performance near the boundaries of the domain. Sometimes the spaces $D^{(p)}$ are orthogonal to the spaces $V^{(p)}$. These are *orthogonal wavelets*. Here the direct sum, $V^{(p)} \oplus D^{(p)} = V^{(p+1)}$, is an orthogonal sum, and the algorithms for analysis and synthesis of $u(x)$ are simplified. Orthogonality is not a requirement. Other settings include *biorthogonal wavelets* [Strang and Nguyen 1996a]. Here orthogonality of $D^{(p)}$ to $V^{(p)}$ is lost, but the two spaces intersect only at zero, hence it remains true that a function in $V^{(p+1)}$ is uniquely decomposable into components in $V^{(p)}$ and $D^{(p)}$. The structure of the direct sum is preserved. Sometimes, we sacrifice orthogonality in exchange for other desirable properties, e.g., compact support, symmetry, smoothness, and approximation power.

3.2.3 Example: The Haar System

The Haar system defines a piecewise constant approximation to $u(x)$ at scale 2^{-p} by measuring its mean value over each interval[3]$I_k^{(p)} := [k2^{-p}, (k+1)2^{-p}[$,

$$u^{(p)}(x) = 2^p \int_{I_k^{(p)}} u(x) dx \; , \quad \forall x \in I_k^{(p)} \; , \quad k \in \mathbb{Z} \; .$$

[3]Closed on the left, open on the right.

This is the L^2-orthogonal projection of $u(x)$ onto the space

$$V^{(p)} = \{u(x) \in L^2 \mid u(x) \text{ is constant over } I_k^{(p)},\ \forall k \in \mathbb{Z}\}\ ,$$

spanned by the orthogonal basis induced by the canonical Haar scaling function $\phi(x) := \chi_{[0,1]}$, where $\chi_{[s,t]}$ is the box function defined as unity over $[s,t]$ and zero elsewhere.

The Haar scaling function satisfies the dilation equation, (3.1), with two non-zero coefficients $a_0 = a_1 = 1/\sqrt{2}$. Equivalently, it satisfies the refinement relation with two non-zero coefficients $a_{ij}^{(p)} = 1/\sqrt{2},\ j \in \{2i, 2i+1\}$. Following (3.2), we flip signs of odd-indexed coefficients, arriving at the wavelet equation

$$\psi(x) = \phi(2x) - \phi(2x-1) = \chi_{[0,\frac{1}{2}]} - \chi_{[\frac{1}{2},1]}\ ,$$

or equivalently, a definition of the details in terms of finer scaling functions:

$$\psi_i^{(p)} = \phi_{2i}^{(p+1)} - \phi_{2i+1}^{(p+1)}\ .$$

The Haar scaling functions are constructed over a tesselation: the construction of the elements follows naturally. To every nested space $V^{(p)}$ we associate a partition of the domain into elements,

$$E^{(p)} := \{\varepsilon_i^{(p)} := I_i^{(p)} \mid i \in \mathbb{Z}\}\ .$$

The restriction of any scaling function $\phi_i^{(p)} \in V^{(p)}$ onto any element of $E^{(p)}$ is constant, i.e., a *very* simple form! The natural support set of a Haar scaling function is a single element: $\mathcal{S}(\phi_i^{(p)}) = \{\varepsilon_i^{(p)}\}$. Similarly, the restriction of any detail $\psi_i^{(p)} \in D^{(p)}$ onto any element of the finer partition, $E^{(p+1)}$, is constant. The natural support set of a Haar wavelet has two elements: $\mathcal{S}(\psi_i^{(p)}) = \{\varepsilon_{2i}^{(p+1)}, \varepsilon_{2i+1}^{(p+1)}\}$.

Finally, we define the tiles. Only the resolving tiles require clarification (the element tiles are always the elements $\varepsilon_i^{(p)}$). In this case, the elements satisfy a nesting relation: $\varepsilon_i^{(p)} = \varepsilon_{2i}^{(p+1)} \cup \varepsilon_{2i+1}^{(p+1)}$. Our job is easy. The level-p resolving tiles are the level-$(p+1)$ elements.

With these definitions in place, we have everything we need to apply the algorithms of Chapter 4 to the Haar system.

3.3 Multiwavelets

Haar's wavelets are orthogonal, antisymmetric, compactly supported, and piecewise constant. Often we want similar *smoother* wavelets. Daubechies proved that we won't find them. Higher-order wavelets cannot be simultaneously orthogonal, (anti)symmetric, and compactly supported. This is too much to ask from a *single* mother wavelet. These properties can peacefully coexist if we turn to the theory of *multi*scaling functions and *multi*wavelets.

3.3.1 Theory

We start with *multiple* canonical scaling functions, $\phi_1(x), \ldots \phi_L(x)$. With careful design, *all these functions have both symmetry, orthogonality, vanishing moments (smoothness), and compact support* [Strela et al. 1999]. Furthermore, the support diameter of single wavelets grows with the number of vanishing moments and the smoothness; multiwavelets typically have shorter supports than single-wavelets thus offering another design parameter to control support diameter.

Each canonical function is accompanied by its translated and dilated "clones," $\phi_{i,j}^{(p)},\ 1 \leq i \leq L,\ j \in \mathbb{Z},\ p \geq 0$. Every space $V^{(p)}$ has as its basis the level-p clones of the L different canonical scaling functions:

$$V^{(p)} := \left\{ \sum_{i=1}^{L} \sum_{j} u_{i,j}^{(p)} \phi_{i,j}^{(p)} \;\middle|\; u_{i,j}^{(p)} \in \mathbb{R} \right\} ,$$

We have a *matrix* dilation equation,

$$\mathbf{\Phi}(x) = \sqrt{2} \sum_{k} \mathbf{A}_k \mathbf{\Phi}(2x - k) ,$$

where $\mathbf{\Phi} = [\phi_1\ \phi_2 \cdots \phi_L]^T$, and the scalar coefficients, a_k, of (3.1) have been replaced by matrices, $\mathbf{A}_k$. Equivalently, the refinement relation is

$$\mathbf{\Phi}^{(p)}(x) = \sum_{k} \mathbf{A}_k \mathbf{\Phi}_k^{(p+1)}(x) .$$

In general, for a particular $1 \leq i \leq L$, the children of $\phi_{i,j}^{(p)}(x)$ are a mix of *all* L kinds of clones of $\phi_1 \ldots \phi_L$ *not* just clones of $\phi_i(x)$.

Now there are L kinds of wavelets,

$$\mathbf{\Psi}(x) = \sqrt{2} \sum_{0}^{N} \mathbf{D}_k \mathbf{\Phi}(2x - k) ,$$

where $\mathbf{\Psi} = [\psi_1\ \psi_2 \cdots \psi_L]^T$ and the scalar coefficients, d_k, of (3.2) have been replaced by "high-pass" matrices, $\mathbf{D}_k$. Every mother wavelet is expressed as a linear combination of all L canonical scaling functions.

3.3.2 Example: Haar's Hats

Building on the exercises in Strang and Nguyen, we present a simple example of multiwavelets based on Haar's scaling functions.

We extend Haar's space of discontinuous piecewise constants to the space of discontinuous piecewise linears,

$$\begin{aligned} V^{(p)} &:= \left\{ u(x) \text{ linear over } I_i^{(p)} \mid i \in \mathbb{Z} \right\} \\ &= \left\{ \sum_j u_{1,j}^{(p)} \phi(x)_{1,j}^{(p)} + \sum_j u_{2,j}^{(p)} \phi(x)_{2,j}^{(p)} \;\middle|\; u_{i,j}^{(p)} \in \mathbb{R} \right\} . \end{aligned}$$

spanned by translates and dilates of two canonical scaling functions (see Figure 3.1):

$$\text{Mean:} \quad \phi_1(x) = \chi_{[0,1]} = \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \textit{(the usual Haar box)}$$

$$\text{Slope:} \quad \phi_2(x) = \begin{cases} 2x-1 & 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \textit{(antisymmetric line with 1 vanishing moment)} .$$

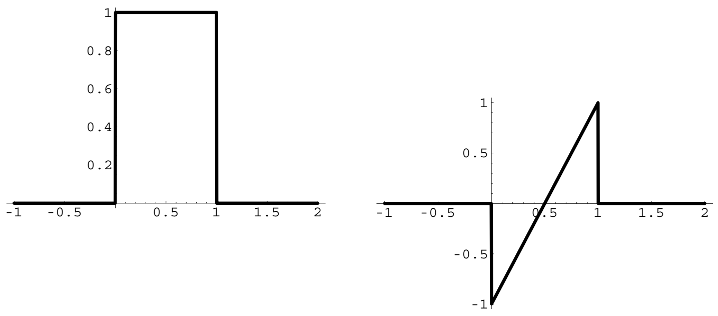


Figure 3.1: The “Haar’s Hats” system has *two* cannonical scaling functions, corresponding to (left) mean and (right) slope.

There are two mother wavelets, orthogonal to each other as well as to the canonical scaling functions, and piecewise linear over $I_0^{(1)}$ and $I_1^{(1)}$ (see Figure 3.2):

$$\text{Symmetric:} \quad \psi_1(x) = \begin{cases} 4x-1 & 0 \le x < \frac{1}{2} \\ 3-4x & \frac{1}{2} \le x < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \textit{(a hat with 1 vanishing moment)}$$

$$\text{Antisymmetric:} \quad \psi_2(x) = \begin{cases} 4x-1 & 0 \le x < \frac{1}{2} \\ 4x-3 & \frac{1}{2} \le x < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \textit{(a zig-zag with 1 vanishing moment)} .$$

The two canonical scaling functions satisfy the matrix dilation equation

$$\begin{bmatrix} \phi_1(x) \\ \phi_2(x) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \phi_1(2x) \\ \phi_2(2x) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \phi_1(2x-1) \\ \phi_2(2x-1) \end{bmatrix} ,$$

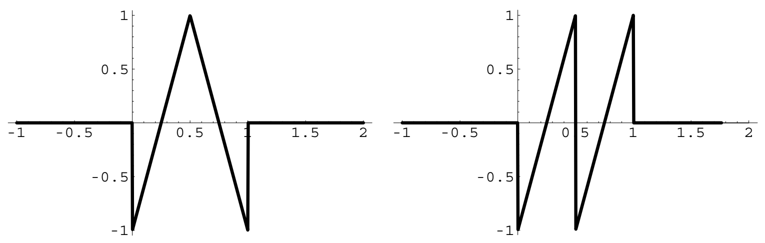


Figure 3.2: The “Haar’s Hats” system has *two* mother wavelets, (left) symmetric and (right) antisymmetric.

and the two mother wavelets satisfy the matrix wavelet equation,

$$\begin{bmatrix} \psi_1(x) \\ \psi_2(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_1(2x) \\ \phi_2(2x) \end{bmatrix} + \begin{bmatrix} 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_1(2x-1) \\ \phi_2(2x-1) \end{bmatrix} .$$

Following the pattern for the original Haar system: the two scaling functions are constructed over a tesselation, leading naturally to a partition of the domain into elements,

$$E^{(p)} := \{\varepsilon_i^{(p)} := I_i^{(p)} \mid i \in \mathbb{Z}\} ,$$

associated to every space $V^{(p)}$. Note that these are the elements of the original Haar system. The restriction of any scaling function $\phi_{1,i}^{(p)} \in V^{(p)}$ or $\phi_{2,i}^{(p)} \in V^{(p)}$ onto any element of $E^{(p)}$ is linear hence simple (but *not* constant like the original Haar function). Similarly, the restriction of any detail $\psi_i^{(p)} \in D^{(p)}$ onto any element of the finer partition, $E^{(p+1)}$, is linear.

The natural supports of the scaling and detail functions are the same as in the original Haar system. The natural support set of either kind of scaling function has one element, $\mathcal{S}(\phi_i^{(p)}) = \{\varepsilon_i^{(p)}\}$. The natural support set of either kind of detail has two elements: $\mathcal{S}(\psi_i^{(p)}) = \{\varepsilon_{2i}^{(p+1)}, \varepsilon_{2i+1}^{(p+1)}\}$. Since the elements are the same as for the original Haar system, then so are the tiles. The level-p resolving tiles are the level-$(p+1)$ elements.

With these definitions in place, we have everything we need to apply the algorithms of Chapter 4 to the multi wavelet “Haar’s Hats” system.

3.4 Finite Elements

Consider a tesselation of the domain, e.g., a two-dimensional triangle mesh. The finite element space consists of all functions which are piecewise polynomials of some order p, i.e., the restriction of $P_N u(x)$ onto any element is a polynomial of degree $p-1$.

The mesh elements are *finite*: the restriction of $P_N u(x)$ onto an element is determined *only by coefficients contained in the element*, i.e., $\{u_i | x_i \in \Omega_{\text{elem}},\ 1 \leq i \leq N\}$. The approximation is locally determined. Its value depends only on coefficients on the interior of the element (hence unique to that element) and the boundary of the element (thus shared with incident elements).

Equivalently, the basis functions are chosen such that they have a small support, covering at most all the elements incident on a mesh vertex. Each basis function is defined by its restriction (the *shape function*) over each element in its domain of support, with some *compatibility condition* for the interfaces between element subdomains, e.g., continuity along a face between two volume elements (for H^1-conforming elements in $\mathbb{R}^3$), equality of the mean value of the shape function on both sides of a face [Crouzeix and Raviart 1973, Rannacher and Turek 1992] or its normal component [Raviart and Thomas 1977, Brezzi and Fortin 1991].

In general there are two kinds of coefficients, corresponding to (a) function values (interpolated points), with associated basis functions satisfying the Kronecker Delta property, and (b) function derivatives (interpolated tangent points, curvature points, etc..), with partial derivatives of the basis function satisfying the Kronecker Delta property. The finite element space consists of continuous functions if (and only if) every mesh node(=vertex) carries an interpolated coefficient. This is the typical compatibility condition for the most common flavor of finite elements. Similarly, the space consists of C^1 functions if the nodes carry also the appropriate tangent coefficients. Thus, by choosing to place coefficients on nodes, boundaries between elements, or the interior of an element, one may introduce some independence between the element-local accuracy versus the global smoothness.

A finite element mesh, on its own, is single-resolution. Of the various strategies for introducing multi-resolution, *element splitting*—mesh division—is the commonest. Consider an element on its own, i.e., let Ω_{elem} be temporarily the entire domain. Divide dyadically the element into progressively finer regular uniform tesselations: each sub-element carries dilations of the original polynomial basis. The coarsest element's basis functions are the multiscaling functions of a multiwavelet system! This multiwavelets are local to (and different for) every element. In summary, for every element *in isolation* we have a complete construction of multiresolution finite elements.

Considering elements in isolation leads to problematic (lack of) smoothness and accuracy at the element boundaries during *adaptive* refinement. The theory of subdivision extends multi-scaling functions to irregular domains; with this theory in place we will return to this topic, considering *all* the elements *not* in isolation.

3.5 Subdivision

Until now we have pursued the theory of refinable functions in the *regular* Euclidean 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. In contrast, we pursue here 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, Dyn and Levin 2002]. 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.

3.5.1 Theory

The basic ingredients are *topological*- and *coefficient-refinement operators,* acting on the *topological entities* and *coefficients*, respectively, of a *mesh*:

Mesh 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 (mesh-)*element* refers to a highest-dimensional topological entity, i.e., face in the bivariate and cell in the trivariate setting.

Coefficients 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 entities (and more than one coefficient may live on a given entity). 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.

Topological Refinement 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 refinement operators[4]split elements or vertices (Fig. 3.3). Less typical (but accommodated here) are 4-8 [Velho and Zorin 2001] and $\sqrt{3}$ [Kobbelt 2000a] schemes.

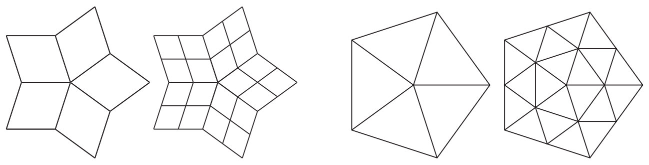


Figure 3.3: Examples of topological refinement operators: quadrisection for quadrilaterals and triangles.

Coefficient Refinement 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

[4] Note: Yannis Ivrissimitzis at MPI Saarbrücken (http://www.mpi-sb.mpg.de/∼ivrissim/) is doing work on decomposing the topological operators into atomic operations. Topological refinements and associated refinement relations may consequently be constructed as repeated simple atomic operators.

mesh. We assume that these operators are linear, finitely supported, of local definition, and depend only on connectivity. Typically these are specified as *subdivision stencils* (see Figure 3.4).

Subdivision Scheme 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. 2002b, Bajaj et al. 2002a] generalizations; Loop [1987], Butterfly [Dyn et al. 1990, Zorin et al. 1996], and $\sqrt{3}$ [Kobbelt 2000a] 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.

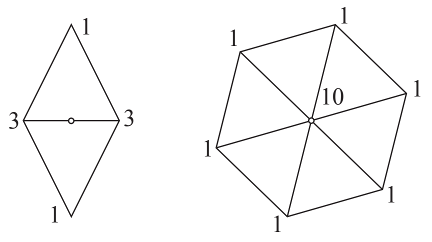


Figure 3.4: Examples of stencils for coefficient refinement. Here the case of quartic box splines with the odd stencil on the left and the even stencil on the right, indicating how the highlighted points in the center are computed as weighted averages (not normalized here) of neighboring points.

Basis Function 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[5] or Alpert's multi-scaling functions [1993].

Refinement Relation 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

$$\phi_i^{(j)}(x) = \sum_k a_{ik}^{(j+1)} \phi_k^{(j+1)}(x) \tag{3.3}$$

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.

[5]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.

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

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

Natural Support Set Recall that 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 the given vertex as its natural support set (Fig. 3.5); 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, $\mathcal{D}^\star(\varepsilon_i^j)$.

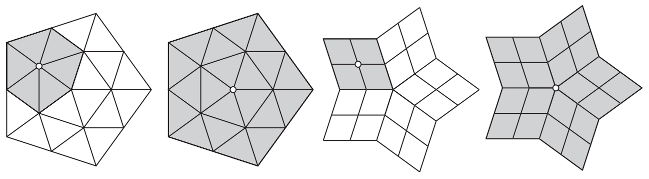


Figure 3.5: Examples of natural support sets. Left to right: linear splines, Loop basis, bilinear spline, and Catmull-Clark basis.

3.5.2 Example: The Loop Scheme

In 1987, Loop proposed a primal subdivision scheme for manifold triangle meshes, generalizing the quartic Box-Splines [Loop 1987, de Boor et al. 1993]. We summarize the salient features of Loop's subdivision scheme and describe the constructions necessary to use this scheme as part of our adaptive solver framework.

Description Loop uses simplicial complexes, or "triangle meshes," with vertices, $V = \{v_i\}$, edges, $E = \{e_j\}$, and triangular faces, $F = \{f_k\}$. The incidence relations define a 2-manifold (with boundary). The initial (coarsest-level) mesh is denoted by $M^{(0)}$.

The topological refinement operator bisects edges and quadrisects faces, as shown in Figure 3.6. This operator maps the level-p mesh to the level-$(p+1)$ mesh, thus constructing an infinite sequence of meshes

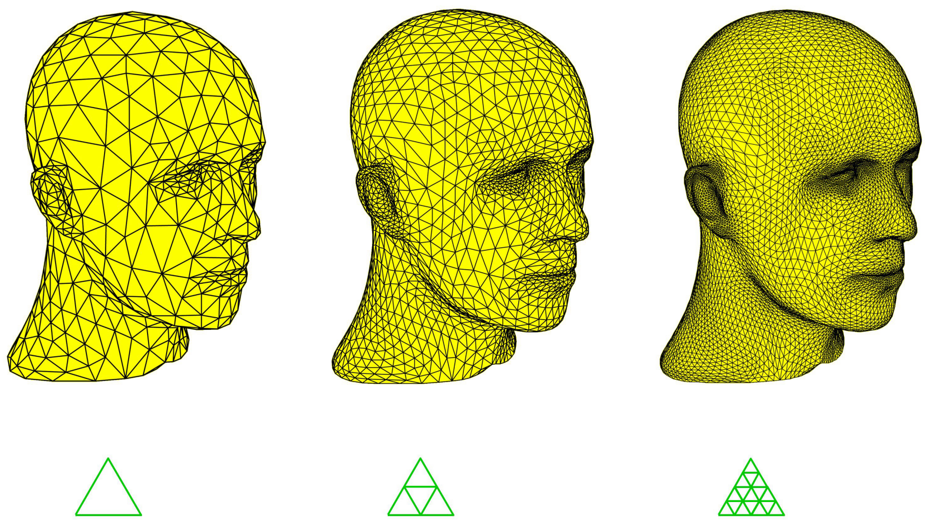


Figure 3.6: Loop's subdivision scheme is based on triangle meshes. Topological refinement, using edge bisection and face quadrisection, produces a sequence of progressively finer meshes, $M^{(0)}, M^{(1)}, M^{(2)}, \ldots$.

$M^{(0)}, M^{(1)}, M^{(2)}, \ldots$. The associated topological entities corresponding to every mesh $M^{(p)}$ are denoted $V^{(p)}$, $E^{(p)}$, $F^{(p)}$, etc.. This is a *primal* scheme, i.e., $V^{(p+1)} \subset V^{(p)}$. The introduced vertices $V^{(p+1)} \backslash V^{(p)}$ are the *odd* vertices at level-$(p+1)$; the other vertices are *even*.

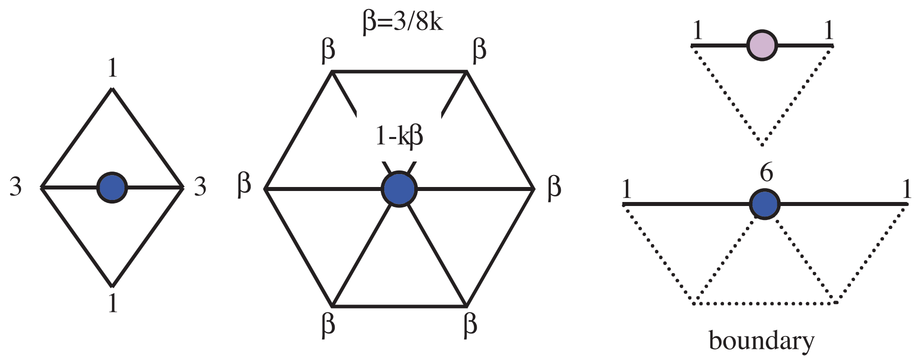


Figure 3.7: Coefficient refinement given by stencil. Note this is linear, finitely supported, of local definition, and depend only on connectivity.

Loop's meshes carry coefficients assigned to the vertices, $V^{(p)}$. Consequently, every basis function is associated to (and centered about) a vertex, as shown in Figure 3.8. The coefficient refinement operator is a linear map from the coefficients of $V^{(p)}$ to those of $V^{(p+1)}$. Its action is easily summarized by stencils for even and odd vertices, shown in Figure 3.7.

For numerical quadrature we require a means to evaluate exactly the limiting value of subdivision at specific points on the domain, i.e., at some given point on or inside a mesh entity. Zorin [2002] has recently demonstrated a particularly flexible approach to exact evaluation; building on the pioneering work of

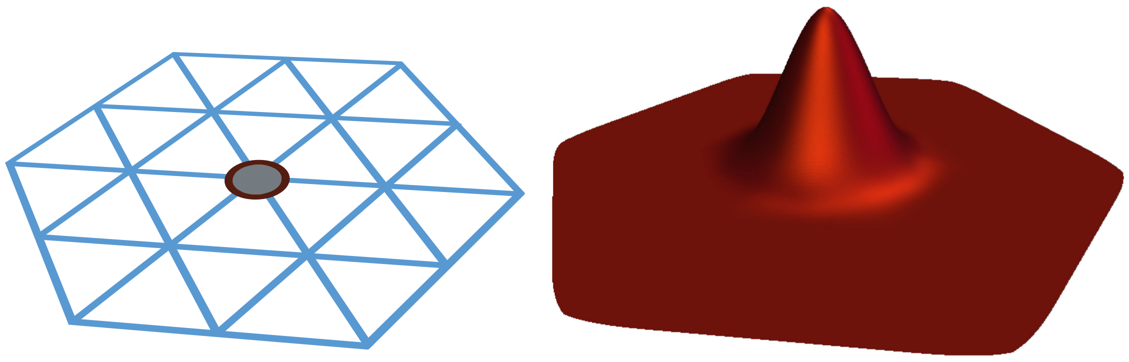


Figure 3.8: The control mesh for Loop's subdivision scheme carries coefficients assigned to vertices. Shown is the basis function associated to the indicated vertex.

Stam [Stam 1998], Zorin shows how to define evaluation operators for parametric families of rules without considering an excessive number of special cases.

This completes our description of Loop's subdivision scheme. We turn to the additional structure required to implement algorithms for natural refinement.

Additional Structure The level-p scaling functions are the basis functions associated to the level-p vertices, $V^{(p)}$. The level-p details are the level-$(p+1)$ odd basis functions. It is straightforward to show that consequently $V^{(p)} \oplus D^{(p)} = V^{(p+1)}$; here $\oplus$ denotes a direct *not* orthogonal sum. The natural support set of a scaling or detail function associated to vertex $v_i^{(p)}$ is the *two-ring* around $v_i^{(p)}$, i.e., the *one-ring* around $v^{(p)}$ are the level-p faces incident to $v_i^{(p)}$, and the two-ring adds also the faces incident to the one-ring.

The integration elements associated to $V^{(p)}$ are the level-p faces, $F^{(p)}$. The restriction of any scaling function onto any *regular* element is a quartic box spine with well-established rules for numerical quadrature [de Boor et al. 1993], i.e., the scaling functions take on a simple form over every (regular) element. An element is regular if its three incident vertices have valence six. In practice, our experimentation has shown that, for purposes of numerical integration, *all* elements may be treated as regular without destroying convergence or accuracy.

Finally, we define the tiles. Only the resolving tiles require clarification (the element tiles are always the elements $\varepsilon_i^{(p)}$). Since our elements are constructed by quadrisection, there is a natural nesting relation between every level-p element and its four subelements on level-$(p+1)$. Whenever such a nesting relation holds, we define the resolving tiles trivially. The level-p resolving tiles are the level-$(p+1)$ elements.

With these constructions we are equipped to apply the algorithms of Chapter 4 to Loop's subdivision scheme.

3.5.3 Example: The Doo-Sabin Scheme

In 1978 Doo and Sabin presented a subdivision scheme generalizing bi-quartic B-Spline refinement to the setting of irregular meshes [Doo and Sabin 1978].

Duals In contrast to our previous examples of primal schemes, the Doo-Sabin is usually presented as a *dual scheme*. The name *dual* comes from the observation that subdivision is performed on the faces of the *dual mesh*.

The dual of an N-dimensional mesh is constructed by associating each d-dimensional entity of the original *primal mesh* to an $(N-d)$-dimensional entity of the *dual mesh,* keeping all the incidence relations, i.e., two vertices incident (connected by an edge) in the primal mesh have two corresponding dual faces incident (along an edge) in the dual mesh, and vice versa.

Description The Doo-Sabin scheme starts with a polygonal mesh with vertices, $V = \{v_i\}$, edges, $E = \{e_j\}$, and polygonal faces, $F = \{f_k\}$. The incidence relations define a 2-manifold (with boundary). The initial (coarsest-level) mesh is denoted by $M^{(0)}$. The dual mesh has *faces* $\hat{V} = \{\hat{v}_i\}$ corresponding to the primal vertices V; edges $\hat{E} = \{\hat{e}_j\}$ corresponding but *not* identical to the primal edges E; and vertices $\hat{F} = \{\hat{f}_k\}$.

Doo-Sabin's topological refinement operator acts on the dual faces $\hat{V}$, splitting every dual face $\hat{v}_i$ into K dual faces, where K is the valence of v_i, i.e., the number of sides to the face $\hat{v}_i$ identically the number of edges incident to $\hat{v}_i$. Other descriptions of Doo-Sabin (equivalently) represent topological refinement as splitting every *primal* vertex into K primal vertices. In describing the constructions needed for our natural refinement algorithms we will find the dual face-split view more useful than the primal vertex-split view.

Repeated application of the refinement operator produces a sequence of primal meshes $M^{(0)}, M^{(1)}, M^{(2)}, \ldots$ with corresponding duals.

Like the two previous primal examples[6], Doo-Sabin's meshes carry coefficients assigned to the vertices, V. Equivalently, the coefficients are assigned to the dual faces, $\hat{V}$. Every basis function is associated to a dual face (equivalently a primal vertex).

The coefficient refinement operator is a linear map from the coefficients of $\hat{V}^{(p)}$ to those of $\hat{V}^{(p+1)}$. Its action is easily summarized by a single stencil [Doo and Sabin 1978].

With this basic description of Doo-Sabin's scheme, we turn to the additional structure needed for our refinement algorithms.

Additional Structure As usual, the level-p scaling functions are the basis functions associated the level-p dual faces, $\hat{v}^{(p)}$. Unlike the earlier primal schemes, which gave us "odd" and "even" and a natural choice for details, the dual schemes do not inherit vertices from the coarser mesh. We lack an obvious way to define

[6]This is a coincidence: in general both primal and dual schemes may carry coefficients associated to *any* topological entity.

the details. Since the spaces $V^{(p)}$ are nested, and we have proper (scaling function) bases, we can surely find bases for the detail spaces, $D^{(p)} := V^{(p+1)} \backslash V^{(p)}$. Since we do not see an obvious choice, We shall not explicitly declare a detail basis. This does not prevent us from continuing and applying our algorithms, although refinement by adding details will remain undefined until we choose some basis for $D^{(p)}$. Meanwhile, we may refine by substitution.

In contrast to the previous primal examples, here we choose as the level-p elements the *dual* faces, $\hat{V}$. Consider a basis function associated to a regular vertex (i.e., a vertex with valence four): its restriction onto any element is a quadratic polynomial. For primal schemes, basis functions have simple forms over primal faces. For dual schemes, basis functions have simple forms over dual faces.

The natural support set of a scaling function associated to $\hat{v}^{(p)}$ is the one-ring of $\hat{v}^{(p)}$, i.e., $\hat{v}^{(p)}$ and all dual faces incident to $\hat{v}^{(p)}$ along a dual edge; equivalently, the dual of the one-ring of $v^{(p)}$.

As in our previous examples, the level-$(p+1)$ elements are nested in the level-p elements, i.e., the parametric support of a level-p dual face may be expressed as the disjoint union of the parametric supports of some level-$(p+1)$ dual faces. As before, the level-p resolving tiles are the level-$(p+1)$ elements, $\hat{V}^{(p+1)}$.

This structure allows us to apply our natural refinement approach to Doo-Sabin's dual scheme.

3.5.4 Finite Elements, Revisited

Recall that considering finite elements in isolation leads to problems at the element boundaries during adaptive refinement.

Consider the basis functions associated to finite element coefficients. Those coefficients internal to an element associate to basis functions which vanish outside the node, i.e., their parametric support is contained in the element. Those coefficients on the boundary of the element associate to basis functions which are supported over (*multiple*) incident elements. To these basis functions dividing an element in isolation is anathema!

Follow the framework of subdivision: divide dyadically each and *every* element. The multiscaling functions will obey a matrix refinement relation. It will not be the dilation and translation of the traditional regular setting, rather the more general relation of subdivision theory. Locally at every element, the subdivision is dyadic as before; globally the concept is meaningless—we are not in the regular setting anymore, Toto! Observe that the generated children functions are always the locally-supported locally-defined basis functions of a *finite*-element mesh.

Having produced a multiresolution basis by considering subdivision of the mesh *as a whole*, never turn back. Adaptive refinement will come from choosing basis functions *not* isolating and splitting individual elements.

Conclusion We reviewed the theories of wavelets, multiscaling functions, finite elements, and subdivision. These far-reaching techniques all serve as constructions for multiresolution discretizations. With so many variations in properties such as locality, smoothness, accuracy, and symmetry, the application must guide the choice.

Chapter 4

Data Structures and Algorithms

What are needed in the basis refinement strategy are efficient data structures and algorithms to (a) keep track of non-zero entries in the stiffness matrices and (b) manage a tesselation of the domain suitable for evaluation of the associated integrals. We provide a semi-formal specification for these requirements, develop the relevant theorems and proofs, and invoke these theorems to produce concrete, provably-correct pseudo-code.

4.1 Preview

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 refinement 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 “buffer 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 our algorithms and we now present our general treatment which relies solely on refinability.

4.2 Specification

To ground our preliminary 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.

4.2.1 Context

```
**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., $\mathbf{K}$.

Now we develop a specification for our system. First the interface. How does the system interact with its environment? What questions can the system and environment ask each other? What commands can each place on the other? Then we specify each interaction. The goal is to make sure that the system (or environment) does what is expected of it; always; not more, not less. Later we will explain the algorithms and data-structures that make up our system, and show that they satisfy the specification.

Correctness = Safety + Progress A correct system has provable *safety* –it does not do unexpected things or reach invalid states– and *progress* –it eventually achieves what is expected. For each interaction, we will include safety and progress specifications. We will also have a global safety specification that applies at all times to entire system. We examine the safety and progress specifications for our system in Sections 4.2.4 and 4.2.5, respectively. First, however, we lay out the key data structures.

4.2.2 Data Structures

Every datum is either *system-global* (instantiated once), *element-local* (instantiated with every active element), or *function-local* (instantiated with every active basis function).

In practice the local data for an element or function exists only when it is active. Here we don't make this distinction so carefully; instead, when deactivating an element or function, we clear its associated data.

Globals At the global scope we instantiate a set of basis functions, a set of elements, and a set of tiles:

$\mathcal{B}$ is the set of active basis functions. Recall from section 2.2.3 that the active functions $\phi \in \mathcal{B}$ span the current trial space.

$\mathcal{E}$ is the set of active domain elements. Recall from section 2.3.1 that the active elements $\varepsilon \in \mathcal{E}$ are the domain elements which support active functions.

$\mathcal{T}$ is the set of active domain tiles. Recall from section 2.3.2 that the active tiles $t \in \mathcal{T}$ form the minimal partition of the domain that resolves the active elements.

Element Locals With every active element $\varepsilon_k^{(r)}$ we instantiate *native* and *ancestral integration tables*. Each table lists the active functions which should be considered when integrating over the element:

$\mathcal{B}^s(\varepsilon_k^{(r)})$ is the native table of $\varepsilon_k^{(r)}$. The table lists all *same-level* active functions $\phi_i^{(r)} \in \mathcal{B}$ with parametric support overlapping the element, $\mathcal{S}(\phi_i^{(r)}) \cap \varepsilon_k^{(r)} \neq \emptyset$.

$\mathcal{B}^a(\varepsilon_k^{(r)})$ is the ancestral table of $\varepsilon_k^{(r)}$. The table lists all *coarser-level* active functions $\phi_i^{(p)} \in \mathcal{B},\ p < r$ with parametric support overlapping the element, $\mathcal{S}(\phi_i^{(p)}) \cap \varepsilon_k^{(r)} \neq \emptyset$.

Function Locals With every active basis function $\phi_i^{(p)}$ we instantiate the coefficient $u_i^{(p)}$ of the finite-basis approximation $\sum u_i^{(p)} \phi_i^{(p)}(x)$ (recall section 2.1). To approximate another function $v(x)$ we store its associated coefficients $v_i^{(p)}$.

System Snapshots The *snapshot* of the system, S, is the entirety of (global and local) state at some instant. Snapshots give us the semantics to describe the effect of an algorithm: we can compare the snapshots, $\bar{S}$, immediately before and, S, immediately after an algorithm executes. In comparing snapshots, we can refer to particular global or local data structures by prepending the snapshot label, e.g., $\overline{S.\mathcal{B}}$ is the set of active functions at snapshot $\bar{S}$. When there is no ambiguity we may drop the prefix and write $\bar{\mathcal{B}}$.

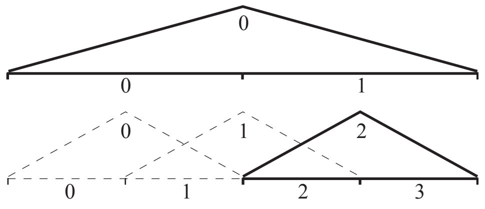


Figure 4.1: Illustrative example of a snapshot. Shown in bold are a pair of active basis functions on mesh levels 0 and 1. The associated data structures are: $\mathcal{B} = \{\phi_0^{(0)}, \phi_2^{(1)}\}$, $\mathcal{E} = \{\varepsilon_0^0, \varepsilon_2^1, \varepsilon_3^1\}$, $\mathcal{S}(\phi_0^{(0)}) = \{\varepsilon_0^0, \varepsilon_1^0\}$, $\mathcal{S}(\phi_2^{(1)}) = \{\varepsilon_2^1, \varepsilon_3^1\}$, $B^s(\varepsilon_0^0) = \{\phi_0^{(0)}\}$, $B^a(\varepsilon_0^0) = \emptyset$, $B^s(\varepsilon_2^1) = \{\phi_2^{(1)}\}$, $B^a(\varepsilon_2^1) = \{\phi_0^{(0)}\}$, $B^s(\varepsilon_3^1) = \{\phi_2^{(1)}\}$, $B^a(\varepsilon_3^1) = \{\phi_0^{(0)}\}$.

4.2.3 System Invariants

Our data structures must embody a consistent, usable description of the trial space and its associated numerical integration scheme. In particular, having chosen some trial space, the element- and tile-related data structures should describe an integration scheme that is appropriate for integrating the approximate solution over the domain, and the element-local integration tables should be up to date. We develop the notion of consistency by assuming a given set of active functions, $\mathcal{B}$, and ensuring that all other data structures are consistent with the assumed set.

Consistency A snapshot is *consistent* ***iff*** the following invariants hold:

Invariant I1 (active elements) *Every active function is fully supported by active elements, and every active element is in the natural support of an active function.*

$$\mathcal{E} = \bigcup_{\phi \in \mathcal{B}} \mathcal{S}(\phi)$$

Invariant I2 (native integration-table) *Every element's native table lists all and only same-level active functions with parametric support overlapping the element.*

$$\mathcal{B}^s(\varepsilon) = \mathcal{B} \cap \mathcal{S}^\star(\varepsilon)$$

Invariant I3 (ancestral integration-table) *Every element's ancestral table lists all and only coarser-level active functions with parametric support overlapping the element.*

$$\mathcal{B}^a(\varepsilon) \quad = \quad \begin{cases} \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \mathcal{B}^s(\varepsilon') & \mathcal{B}^s(\varepsilon) \neq \emptyset \quad \textit{active hence valid} \\ \emptyset & \mathcal{B}^s(\varepsilon) = \emptyset \quad \textit{inactive hence uninitialized} \end{cases}$$

The third invariant explicitly clears the ancestral table for inactive elements. In contrast, in the second invariant, there is no need to explicitly clear the native integration table, it is already empty whenever the element is inactive.

4.2.4 Safety of Algorithms

The effect of every algorithm can be expressed as predicates on the snapshots before and after the algorithm, so-called *pre-* and *post-conditions*. Every condition is classified as checking either *safety* or *progress*.

Safety Predicates may check that the algorithm maps the space of consistent snapshots onto itself. These are *safety conditions.* If it starts in a safe state, a safe algorithm terminates in a safe state. In our setting, pre- and post-conditions for any safe algorithm are always

$$\begin{array}{ll} < \bar{S} \text{ is consistent} > & \textit{pre-condition} \\ S \leftarrow \textbf{Algorithm}(\bar{S}) & \textit{algorithm execution} \\ < S \text{ is consistent} > & \textit{post-condition} \ . \end{array}$$

4.2.5 Progress of Refinement Algorithms

The empty algorithm is a trivially safe algorithm. But it is boring. The progress condition checks that the algorithm is useful.

Progress Predicates that check for useful work are *progress conditions*. Below are the progress specifications for each algorithm.

In the formal description, we explicitly denote each algorithm as taking an entire system snapshot, $\bar{S}$, as input and producing a modified snapshot, $S \leftarrow \mathbf{Algorithm}(\bar{S}, a, b, \ldots)$. In the accompanying narrative we informally omit the references to the snapshots, writing $\mathbf{Algorithm}(a, b)$.

Observe that the progress conditions specify precisely what happens to the active set. The remaining data structures are consequently (uniquely) determined by the safety condition.

Activation of Basis Function The algorithm $\mathbf{Activate}(\hat{\phi})$ adds some inactive basis function $\hat{\phi}$ to the active set; that is the only modification made to the active set.

$$\begin{array}{ll} < \hat{\phi} \notin \overline{S.\mathcal{B}} > & \textit{$\hat{\phi}$ is inactive} \\ S \leftarrow \textbf{Activate}(\overline{S}, \hat{\phi}) & \textit{activation} \\ < S.\mathcal{B} = \overline{S.\mathcal{B}} \cup \{\hat{\phi}\} > & \textit{$\hat{\phi}$ is active} \end{array}$$

Deactivation of Basis Function The algorithm $\mathbf{Deactivate}(\hat{\phi})$ removes some specified active basis function $\hat{\phi}$ from the active set; that is the only modification made to the active set.

$$\begin{array}{ll} < \hat{\phi} \in \overline{S.\mathcal{B}} > & \textit{$\hat{\phi}$ is active} \\ S \leftarrow \textbf{Deactivate}(\overline{S}, \hat{\phi}) & \textit{deactivation} \\ < S.\mathcal{B} = \overline{S.\mathcal{B}} \backslash \{\hat{\phi}\} > & \textit{$\hat{\phi}$ is inactive} \end{array}$$

Refinement by Substitution The algorithm $\mathbf{Refine}(\hat{\phi})$ removes some specified active basis function $\hat{\phi}$ from the active set, and adds the inactive children of $\hat{\phi}$ to the active set; the is the only modification made to the active set.

$$\begin{array}{ll} < \hat{\phi} \in \overline{S.\mathcal{B}} > & \textit{$\hat{\phi}$ is active} \\ S \leftarrow \textbf{Refine}(\overline{S}, \hat{\phi}) & \textit{substitute refinement relation} \\ < S.\mathcal{B} = \overline{S.\mathcal{B}} \backslash \{\hat{\phi}\} \cup \mathcal{C}(\hat{\phi}) > & \textit{$\hat{\phi}$ is inactive and its children are active} \end{array}$$

4.2.6 Progress of Integration Algorithms

The integration algorithms do not alter the state of the data structures. They are trivially safe, and their progress is not evident in comparing the snapshots. Here progress is specified in terms of an *evaluation list* produced by the algorithm.

Integration Over Bilinear Forms (mixed-levels) In the solution of linear (or linearized) problems, a key component of the solver is the evaluation of integrals of some bilinear form, $a(\cdot,\cdot)$, which takes as arguments two functions, $u(x)$ and $w(x)$, both discretized with the same finite basis. Algebra gives more insight here:

$$\begin{aligned} & \int_\Omega a\left(P_n u(x), P_n w(x)\right) dx \qquad (4.1) \\ = \quad & \int_\Omega a\left(\sum_{\phi_i^{(p)} \in \mathcal{B}} u_i^{(p)} \phi_i^{(p)}, \sum_{\phi_j^{(q)} \in \mathcal{B}} u_j^{(q)} \phi_j^{(q)} \right) dx \\ = \quad & \sum_{\phi_i^{(p)} \in \mathcal{B}} \sum_{\phi_j^{(q)} \in \mathcal{B}} u_i^{(p)} u_j^{(q)} \int_\Omega a\left(\phi_i^{(p)}, \phi_j^{(q)}\right) dx\ . \end{aligned}$$

We need to consider the action of $a(\cdot,\cdot)$ on every pair of active basis functions. For efficiency, we should avoid considering pairs which do dot have overlapping parametric support, as their term in the summation is zero. For every pair of active functions, $\phi_i^{(p)}$ and $\phi_j^{(q)}$, we carry out the integration over elements of the finer level, q (without loss of generality assume $q \leq p$). We now show that very level-q element that is required to carry out this integral is active, and has integration tables which list both $\phi_i^{(p)}$ and $\phi_j^{(q)}$. Since $\phi_i^{(p)}$ and $\phi_j^{(q)}$ are active, by the the consistency requirements (by I1) the elements in their natural support sets, $\mathcal{S}(\phi_i^{(p)})$ and $\mathcal{S}(\phi_j^{(q)})$, are active. Furthermore, the local integration tables of every element $\varepsilon_k^{(q)} \in \mathcal{S}(\phi_j^{(q)})$ contain both $\phi_i^{(p)}$ and $\phi_j^{(q)}$, if and only if the parametric supports of $\phi_i^{(p)}$ and $\phi_j^{(q)}$ overlap over $\varepsilon_k^{(q)}$ (by I2 and I3). Thus every level-q element that should be involved in the evaluation of the integral, $\int_\Omega a\left(\phi_i^{(p)}, \phi_j^{(q)}\right) dx$, is active and properly initialized. To evaluate (4.1), we iterate over every active element, and consider all interactions between functions overlapping that element, as recorded by the element's integration tables.

The algorithm $\mathbf{IntegrateBilinearForm}$ evaluates the bilinear form, $a(\cdot,\cdot)$, many times. To analyze the behavior of this algorithm, let us pretend that the algorithm keeps a list of all the evaluations. We label each evaluation as the 3-tuple $(\phi_i^{(p)}, \phi_j^{(q)}, \varepsilon_k^{(r)})$ indicating the integration of $a(\phi_i^{(p)}, \phi_j^{(q)})$ over $\varepsilon_k^{(r)}$. In general $a(\cdot,\cdot)$ is not invariant to permutations of its two arguments, therefore every 3-tuple is unique, i.e., $(\phi_i^{(p)}, \phi_j^{(q)}, \varepsilon_k^{(r)}) \neq (\phi_j^{(q)}, \phi_i^{(p)}, \varepsilon_k^{(r)})$.

With this notation, we require $\mathbf{IntegrateBilinearForm}$ to return the evaluation list

$$\bigcup_{\varepsilon_k^{(q)} \in \mathcal{E}} \quad \bigcup_{\phi_j^{(q)} \in \mathcal{B}^s(\varepsilon_k^{(q)})} \quad \bigcup_{\phi_i^{(p)} \in \mathcal{B}^s(\varepsilon_k^{(q)}) \cup \mathcal{B}^a(\varepsilon_k^{(q)})} (\phi_i^{(p)}, \phi_j^{(q)}, \varepsilon_k^{(q)}) \,. \tag{4.2}$$

Integration Over Finest Cells The algorithm $\mathbf{Integrate}$ evaluates the (potentially non-linear) integral using a partition of the domain into tiles. The integrand is evaluated once per tile, thus specifying the set of active tiles fixes the specification of the algorithm.

Recall that the trial space is spanned by functions from different nesting levels, consequently we need a partition that will resolve these different resolutions. $\mathbf{Integrate}$ evaluates over the minimal set of element- and resolving-tiles such that (a) the tiles partition the domain, and (b) the partition has sufficient resolution: every leaf element is a tile, where *leaf* means an active element with all descendants inactive.

Observe that while the set of active elements *always* covers the domain, the set of leaves in general does not cover the entire domain (see Figure 4.2). With this, we will designate every leaf as an active tile, and in the remaining gaps we will introduce resolving tiles.

Consider the following *tile coloring problem* (TCP). We color every tile in the infinite hierarchy black, red, or white: *black* if the tile is *too coarse* to resolve some finer active descendant, *white* if the tile is *too fine*, or *red* if the tile fits. The coarser black and finer white tiles will form a "sandwich" around a thin sheet of red tiles—these red tiles form our partition.

Tile coloring problem The tile coloring problem is defined as follows. The color of an element tile is

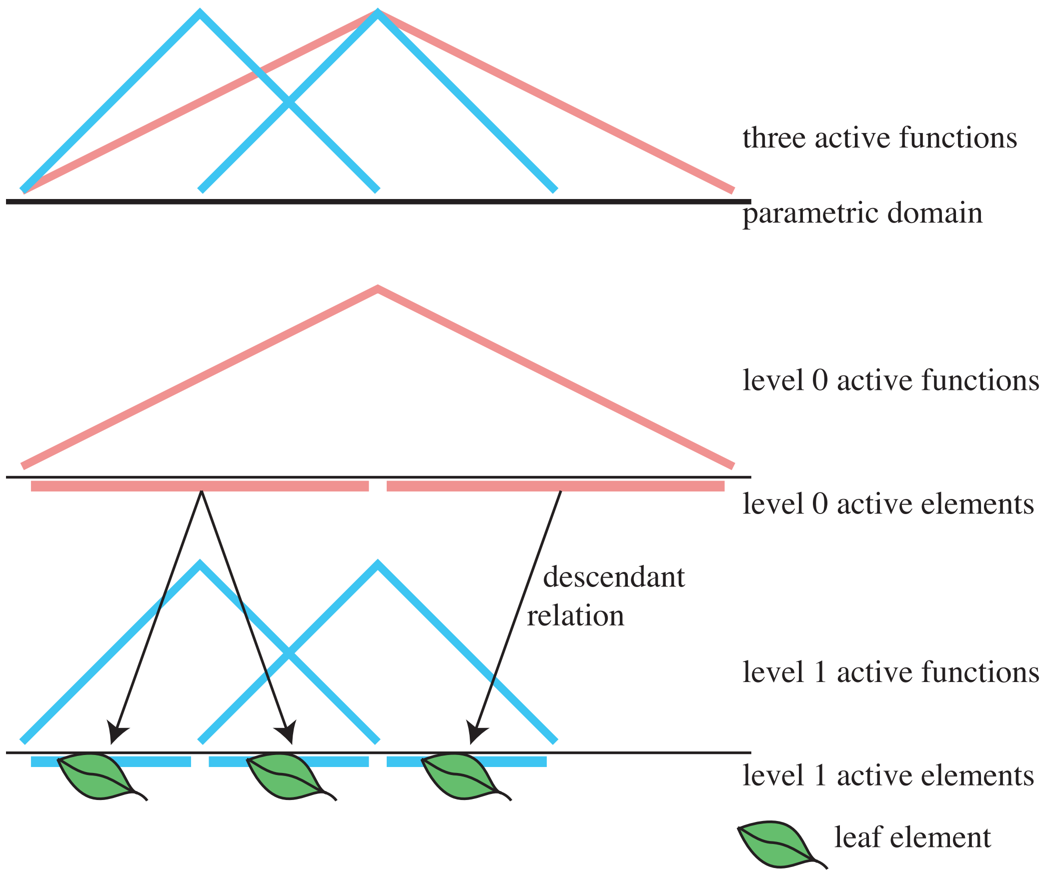


Figure 4.2: Three active functions (top row, hat functions) induce five active elements (thick horizontal bars). Both of the level 0 active elements have active descendants (as shown by the arrows), hence they are not leaves. The level 1 active elements are leaves, and by construction they occupy disjoint pieces of the domain. Furthermore, in general they do not cover the entire domain: for the illustrated case, only 3/4 of the domain are covered by the leaves.

(TCP1) **black** if any of its descendants are active,

(TCP2) else **red** if it is active,

(TCP3) else **white**.

The color of a resolving tile is

(TCP4) **red** if its coarser-link is black and its finer-link is white,

(TCP5) else **black** if its coarser-link is black,

(TCP6) else **white**.

Recall from Section 2.3.2 that the finer-link, $\mathcal{L}(t)$, of resolving-tile, t, is the single overlapping element-tile at the next-finer level; the adjoint relationship gives the coarser-link, $\mathcal{L}^*(t)$, i.e., the single overlapping element tile at the same level as t. The link mapping may also be applied to element-tiles: $\mathcal{L}(\varepsilon)$ is the finer-link of ε, i.e., the set of *potentially multiple* overlapping resolving tiles at the same level as ε; $\mathcal{L}^*(\varepsilon)$ is the coarser-link of ε, i.e., the set of overlapping resolving tiles at the next-coarser level from ε.

The evaluation-list of $\mathbf{Integrate}$ consists of all red tiles, each one accompanied by a table of overlapping active functions.

We shall see that the nature of TCP is that an incremental change in the active set $\mathcal{B}$ leads to an incremental change in the tile coloring: this invites an incremental approach to coloring with consequent economy in refinement and integration.

4.2.7 Maintaining Linear Independence (The Basis Property)

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 detail refinement is applied then the active set is always a proper basis. If other refinement strategies are used (e.g., substitution, and selective (de)activation of individual functions) then maintaining a proper basis requires special care. Our paper [Krysl et al. 2003] treats the specific case of 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 [Kraft 1997]. Finally, in some settings, such as our explicit time-integration of non-linear thin-shells (see Chapter 5), we observe that the solution process remains well-behaved even without linear independence of the active set.

4.3 Theorems

Guided by the above specification, we present and prove theorems that translate fluently into implementable algorithms.

The proofs for activation and deactivation are constructive. While these proofs may be more verbose than those based on contradiction, constructive proofs act as prescriptions for programs.

4.3.1 Activation of Basis Function

Suppose we are in consistent state $\overline{S}$, and we activate some function $\hat{\phi}$. What does the new state S look like? This theorem answers that question. The theorem is easily transcribed into implementable instructions; the associated proof provides insight into the algorithm.

Theorem 1 (activation) *Suppose: (1) $\overline{S}$ is consistent, (2) $\hat{\phi} \notin \overline{\mathcal{B}}$, and (3) $\mathcal{B} = \overline{\mathcal{B}} + \hat{\phi}$. S is consistent* ***iff*** *three conditions hold:*

1. $\mathcal{E} = \overline{\mathcal{E}} \cup \mathcal{S}(\hat{\phi})$

   In activating $\hat{\phi}$, we might activate elements.

2. $\mathcal{B}^s(\varepsilon) = \begin{cases} \overline{\mathcal{B}^s(\varepsilon)} + \hat{\phi} & \varepsilon \in \mathcal{S}(\hat{\phi}) \quad \textit{update required} \\ \overline{\mathcal{B}^s(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) \quad \textit{not affected} \end{cases}$

   In activating $\hat{\phi}$, we might update some native integration-tables.

3. $\mathcal{B}^a(\varepsilon) = \begin{cases} \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \mathcal{B}^s(\varepsilon') & \overline{\mathcal{B}^s(\varepsilon)} = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi}) & \textit{initialize} \\ \overline{\mathcal{B}^a(\varepsilon)} + \hat{\phi} & \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset \wedge \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') & \textit{update} \\ \overline{\mathcal{B}^a(\varepsilon)} & \mathcal{B}^s(\varepsilon) = \emptyset \vee \left(\overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset \wedge \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon')\right) & \textit{not affected} \end{cases}$

   In activating $\hat{\phi}$, we might update ancestral integration-tables.

Proof Please refer to this chapter's Appendix for the formal proof.

4.3.2 Deactivation of Basis Function

Suppose we are in consistent state $\overline{S}$, and we deactivate some function $\hat{\phi}$. What does the new state S look like?

Theorem 2 (deactivation) *Suppose: (1) $\overline{S}$ is consistent, (2) $\hat{\phi} \in \overline{\mathcal{B}}$, and (3) $\mathcal{B} = \overline{\mathcal{B}} - \hat{\phi}$. S is consistent* ***iff*** *three conditions hold:*

1. $\mathcal{E} = \overline{\mathcal{E}} - \left\{\varepsilon \in \mathcal{S}(\hat{\phi}) \mid \mathcal{B}^s(\varepsilon) = \emptyset : \varepsilon\right\}$

   In deactivating $\hat{\phi}$, we might deactivate elements.

2. $\mathcal{B}^s(\varepsilon) = \begin{cases} \overline{\mathcal{B}^s(\varepsilon)} - \hat{\phi} & \varepsilon \in \mathcal{S}(\hat{\phi}) \quad \textit{update required} \\ \overline{\mathcal{B}^s(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) \quad \textit{not affected} \end{cases}$

   In deactivating $\hat{\phi}$, we might update some native integration-tables.

3. $\mathcal{B}^a(\varepsilon) = \begin{cases} \emptyset & \mathcal{B}^s(\varepsilon) = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi}) & \textit{clear} \\ \overline{\mathcal{B}^a(\varepsilon)} - \hat{\phi} & \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') & \textit{update} \\ \overline{\mathcal{B}^a(\varepsilon)} & \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \vee & \\ & \left(\mathcal{B}^s(\varepsilon) = \emptyset \wedge \varepsilon \notin \mathcal{S}(\hat{\phi})\right) & \textit{not affected} \end{cases}$

   In deactivating $\hat{\phi}$, we might update ancestral integration-tables.

Proof Please refer to this chapter's Appendix for the formal proof.

4.3.3 Refinement by Substitution

With theorems for activation and deactivation in place, we can easily prove theorems for compound operations, such as substitution refinement.

Theorem 3 (substitution) *An application of **Deactivate** composed with multiple applications of **Activate**, as shown below, is safe and effects a refinement of $\hat{\phi}$ by substitution:*

$$\begin{array}{ll} < \bar{S} \textit{ is consistent } \wedge \hat{\phi} \in \bar{\mathcal{B}} > & \hat{\phi} \textit{ active, state is consistent} \\ < \{\varphi_1, \ldots, \varphi_N\} = \mathcal{C}(\hat{\phi}) \backslash \bar{\mathcal{B}} > & \textit{give names to the inactive children of } \hat{\phi} \\ S_1 \leftarrow \mathbf{Activate}(\bar{S}, \varphi_1) & \textit{activate first inactive child} \\ \ldots & \ldots \\ S_N \leftarrow \mathbf{Activate}(S_{N-1}, \varphi_N) & \textit{activate } N^{th} \textit{ inactive child} \\ S \leftarrow \mathbf{Deactivate}(S_N, \hat{\phi}) & \textit{deactivate } \hat{\phi} \\ < S \textit{ is consistent} > & \textit{state is consistent} \\ < \hat{\phi} \notin S.\mathcal{B} \wedge \mathcal{C}(\hat{\phi}) \subset S.\mathcal{B} > & \hat{\phi} \textit{ was replaced by its children} \end{array}$$

Proof sketch Compose the deactivation theorem with N applications of the activation theorem.

4.3.4 Integration of Bilinear Forms

The specification of the evaluation list, (4.2), maps directly into an efficient algorithm, presented in Section 4.4.5.

4.3.5 Integration over Tiles

Lemma 1 (black tiles) *Every ancestor element of a black element-tile is black.*

Proof We prove that every element t_2, ancestor of black element-tile t, is black. Since t is black, then it has an active descendant (by TCP1 on t). Any descendant of t is a descendant of t_2, thus t_2 has an active descendant, and t_2 is black (by TCP1 on t_2).

Lemma 2 (red tiles) *Every descendant tile of a red tile is white.*

Proof sketch Case 1: Red element tile t. Consider any particular element descendant t_1. It is inactive (by TCP1 on t) thus it is not red (by TCP2 on t_1). None of its descendants are active, since $\mathcal{D}(t_1) \subset \mathcal{D}(t)$, thus it is not black (by TCP1 on t_1). Therefore it is white. Consider any particular resolving-tile descendant t_2. Its parent is not black (proof: choose t_1 to be the parent). Therefore t_2 is white (by TCP6 on t_2).

Case 2: Red resolving tile t. Its child, the element t_3, is white (by TCP4 on t). Every descendant of t_3 is white (by the same argument as Case 1).

Lemma 3 (white tiles) *The coarser-link of a white tile is not black.*

Proof sketch Case 1: White element tile t. Assume its resolving-tile parent t_1 is black. Then the parent of t_1 is black (by TCP5 on t_1). But then t_1 is red (by TCP4 on t_1) which is a contradiction. Therefore the parent of t is not black.

Case 2: White resolving tile t. Assume its element-tile parent t_2 is black. Then t is either red (by TCP4 on t) or black (by TCP5 on t), which is a contradiction. Therefore the parent of t is not black.

Theorem 4 (tile coloring produces minimal valid partition) *The red tiles, specified by the tile coloring problem, form a minimal partition of the domain that resolves every active element.*

Proof sketch In Part I, we prove that the red tiles form a valid tiling. In part II, we show that the tiles are not excessively fine.

Part I The red tiles form a valid tiling, i.e., (1) the red tiles do not overlap, (2) every leaf element is a red tile, and (3) the red tiles cover the domain. Together (1) and (3) guarantee that the red tiles are a partition of the domain, and (2) guarantees that they resolve the finest active elements.

Assume two red tiles overlap, then one must be a descendant of the other. But by the Lemma 2 the descendants of a red tile are white. Therefore, (1) red tiles do not overlap.

By definition, a leaf element does not satisfy TCP1, therefore, by TCP2, (2) every leaf element is a red tile.

Pick any point P on the domain. We show how to find the red tile that contains P. Choose the coarsest-level element tile containing P. By construction that element tile is red or black, since the active elements cover the domain. If it is red, QED; assume it is black. Traverse down the hierarchy of tiles as follows: arriving at an element tile, proceed to the resolving tile containing P; arriving at resolving tile, proceed to its child tile. At every step of the traversal, examine the color of the tile. It must be black or red: it cannot be white by Lemma 3. If it is red, QED. If it is black, continue the traversal. Assume that the finest element is at level q. Then there are no black tiles at levels $\geq q$. Consequently the traversal must reach a red tile. Therefore, (3) the red tiles cover the domain.

Part II The red tiles are not excessively-fine in the following sense: choose any element which has no active descendants. That element tile has no red descendants.

By construction the element is not black. If it is red then, by Lemma 2, QED. If it is white, then all of its descendants (element- and resolving-tiles) are white. To see this: observe that each of its element-descendants also has no active descendants hence is also white; consequently all its resolving-tile descendants are also white.

4.4 Algorithms

4.4.1 Initialization

Initially, $\mathcal{B} := \{\phi_i^{(0)}\}$ is the set of level-0 scaling functions, $\mathcal{E} := E^{(0)}$ is the set of level-0 elements, and the integration tables of every active element, $\varepsilon \in \mathcal{E} = E^{(0)}$, are initialized (by the consistency requirement) to $B^a(\varepsilon) = \emptyset$ and $B^s(\varepsilon) = \mathcal{S}^\star(\varepsilon)$.

4.4.2 Activation of Basis Function

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, following the prescription of Theorem 1 (activation):

```
Activate($\phi$)
1   $\mathcal{B} \cup= \{\phi\}$
2   ForEach $\varepsilon \in \mathcal{S}(\phi)$ do
3       $B^s(\varepsilon) \cup= \{\phi\}$
4       // upon activation initialize ancestral list
5       If $\varepsilon \notin \mathcal{E}$ then $B^a(\varepsilon) \cup=$ AncestralList($\varepsilon$) ; $\mathcal{E} \cup= \{\varepsilon\}$ fI
6       // add to ancestral lists of active descendants
7       ForEach $\gamma \in (\mathcal{D}(\varepsilon) \cap \mathcal{E})$ do $B^a(\gamma) \cup= \{\phi\}$

AncestralList($\varepsilon$)
1   $\rho := \emptyset$
2   ForEach $\gamma \in \mathcal{D}^\star(\varepsilon) \cap \mathcal{E}$ do
3       $\rho \cup= B^s(\gamma) \cup B^a(\gamma)$
4   return $\rho$
```

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 (code line 3, theorem condition 2). Furthermore since ϕ is active its supporting cells are active (code line 5 and theorem condition 1): they are activated (if inactive) by adding them to the set of active cells and initializing their table of ancestral active-functions (theorem condition 3-*initialize*). 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 (code line 7 and theorem condition 3-*update*).

4.4.3 Deactivation of Basis Function

When a basis function ϕ is deactivated $\mathcal{B}$ and $\mathcal{E}$ as well as B^s and B^a must be updated, following the prescription of Theorem 2 (deactivation):

```
Deactivate($\phi$)
1   $\mathcal{B} \setminus= \{\phi\}$
2   ForEach $\varepsilon \in \mathcal{S}(\phi)$ do
3       $B^s(\varepsilon) \setminus= \{\phi\}$
4       // deactivate element?
5       If $B^s(\varepsilon) = \emptyset$ then $\mathcal{E} \setminus= \{\varepsilon\}$, $B^a(\varepsilon) := \emptyset$ fI
6       // update ancestor lists of active descendants
7       ForEach $\gamma \in \mathcal{D}(\varepsilon) \cap \mathcal{E}$ do $B^a(\gamma) \setminus= \{\phi\}$
```

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 (code line 3 and theorem condition 2) and from the table of ancestral active-functions of every active descendant of ε (code line 7 and theorem condition 3-*update*). Furthermore if the supporting cell is left with an empty active-function table then it is deactivated and its ancestral table is cleared (code line 5 and theorem conditions 1 and 3-*clear*).

4.4.4 Refinement by Substitution

Assuming that an appropriate error estimator is at hand we can consider a wide variety of adaptive solver strategies built on top of **Activate**. Two example strategies are detail- and substitution-refinement. The former is simply activating a detail function. The latter is implemented via compound applications of **Activate** and **Deactivate**. The following algorithm refines an active basis function, $\varphi_i \in \mathcal{B}$, using the method of substitution:

```
Refine($\varphi_i$)
1   ForEach $\varphi_j \in \mathcal{C}(\varphi_i)$ do
2       If $\varphi_j \notin \mathcal{B}$ then Activate($\varphi_j$) ; $u_j := 0$ fI
3       $u_j += a_{ij} u_i$
4   Deactivate($\varphi_i$) ; $u_i := 0$
```

Here each u_j is the coefficient associated with φ_j, and a_{ij} is the weight of φ_j in the refinement relation of φ_i (Eqn. 3.3). Note that the algorithm is semantically a reproduction of Theorem 3 (substitution), garnished with updates to the DOF coefficients u_j.

4.4.5 Integration of Bilinear Forms

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. We interpret (4.2) (see Section 4.2.6) as a literal description of the following algorithm:

```
ComputeStiffness($\mathcal{E}$)
1   ForEach $\varepsilon \in \mathcal{E}$ do
2     ForEach $\phi \in B^s(\varepsilon)$ do
3       $k_{\phi\phi} += $ Integrate($\phi$,$\phi$,$\varepsilon$)
4       ForEach $\psi \in B^s(\varepsilon) \setminus \{\phi\}$ do
5         $k_{\phi\psi} += $ Integrate($\phi$,$\psi$,$\varepsilon$)
6         $k_{\psi\phi} += $ Integrate($\psi$,$\phi$,$\varepsilon$)
7       ForEach $\psi \in B^a(\varepsilon)$ do
8         $k_{\phi\psi} += $ Integrate($\phi$,$\psi$,$\varepsilon$)
9         $k_{\psi\phi} += $ Integrate($\psi$,$\phi$,$\varepsilon$)
```

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: $\{\varepsilon \mid \varepsilon \in \mathcal{S}(\phi_f) \wedge \mathcal{D}^\star(\varepsilon) \cap \mathcal{S}(\phi_c) \neq \emptyset\}$. 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 $\mathbf{K}$.

4.4.6 Integration over Tiles

While the simple specification of TCP can be translated directly into pseudo-code, for better performance it is desirable to use an *incremental* algorithm, locally updating to the tile coloring whenever an element becomes (in)active. Here we present one approach to incremental coloring. The function **UpdateTilesOnElementActivation** should be called immediately after an element is activated.

```
**UpdateTilesOnElementActivation**$(\varepsilon)$
//if element tile was black, we're done
If GetColor$(\varepsilon) \neq$ black
    //element tile changes from white to red
    SetColor$(\varepsilon, \textbf{red})$
    //all ancestors are black
    ForEach $\gamma \in \mathcal{D}^\star(\varepsilon)$ do SetColor$(\gamma, \textbf{black})$
```

We derived this algorithm from the rules TCP1–TCP3 by tracing the consequence of a single element ε becoming active. The consequent effect on any element-tile ε_2 is as follows. If ε_2 is black, it remains black (by TCP1 on ε_2, noting that no descendant has been *de*activated). Otherwise, if ε_2 is an ancestor of ε, then it becomes black (by TCP1 on ε_2). Finally, if $\varepsilon_2 = \varepsilon$, then it becomes red (it was not previously black, hence by TCP1 it has no active descendants; it is active, thus by TCP2 it is red).

Note above that the color of every element stays the same or *moves upward in the ladder of colors* given by TCP1–TCP3, i.e., from top to bottom: black, red, white. In contrast, when we update colors after an element is deactivated, colors which change do so "downward."

Recall from TCP4–TCP6 that the color of a resolving tile depends on the color of its linked element-tiles. For this reason, when an element-tile changes color, we update its linked resolving tiles:

```
**SetColor**$(\varepsilon, c)$
//proceed only if new color differs from current
If GetColor$(\varepsilon) \neq c$
    $\varepsilon$.color $:= c$
    //update the linked resolving-tiles
    ForEach $\gamma \in \mathcal{L}(\varepsilon) \cup \mathcal{L}^*(\varepsilon)$ do UpdateResolvingTile$(\gamma)$
```

To compute the color of a resolving-tile, we apply rules TCP4–TCP6 directly:

```
**UpdateResolvingTile**$(t)$
If GetColor$(\mathcal{L}^*(t)) =$ black
    $t$.color $:=$ (GetColor$(\mathcal{L}(t)) =$ white) ? red : black
else $t$.color $:=$ white; fI
```

Following the same pattern as above, when an element is deactivated, we call **UpdateTilesOnElementActivation**:

```
**UpdateTilesOnElementDeactivation**($\varepsilon$)
//if element tile was black, we're done
**If GetColor**($\varepsilon$) $\neq$ **black**
  //element tile changes from red to white
  **SetColor**($\varepsilon$, **white**)
  //update ancestors
  **ForEach** $\gamma \in \mathcal{D}^\star(\varepsilon)$ **do**
    //update only black ancestors with no active descendants
    **If GetColor**($\gamma$) $=$ **black** $\wedge \mathcal{D}(\gamma) \cap \mathcal{E} = \emptyset$ **then**
      //ancestor becomes red if active, white otherwise
      **SetColor**($\gamma$, $(\gamma \in \mathcal{E})$ ? **red** : **white**); **fI**
```

If ε was black, it remains black (code line 2, by TCP1 on ε, since no descendant of ε has been deactivated), and none of its ancestors are affected (by Lemma 1). Otherwise ε was red (by TCP2 on ε, since it was active), and changes to white (code line 4, by TCP3 on ε, since it is now inactive). The consequent effect on any element-tile ancestor ε_2 of ε is as follows. If ε_2 is red, it is unchanged (by TCP2 on ε_2, since no descendant of ε_2 has been activated, and ε_2 continues to be active). If ε_2 was black, then it is effectively re-evaluated from scratch (code lines 6–10, following rules TCP1–TCP3).

The above algorithms maintain incrementally the coloring of the tiles under arbitrary (de)activations of the elements. In many applications, we have more information about when elements may become (in)active, and we can put that information to effect by simplifying the above algorithms. In particular, applications which enforce a *one level difference*, or *restriction criterion*, can simplify line 6 of **UpdateTilesOnElementActivation** and lines 6 and 8 of **UpdateTilesOnElementDeativation** by replacing the ancestor (resp. descendant) expression with a parent (resp. child) expression. If this convenient simplification cannot be made, then implementation of line 8 of **UpdateTilesOnElementDeativation** requires special care: each element should maintain a local counter of its active descendants, thus permitting rapid evaluation of $\mathcal{D}(\gamma) \cap \mathcal{E} = \emptyset$.

The above code is just one approach to incrementally solving the tile coloring problem. Other incremental approaches may be used instead.

The evaluation-list of $\mathtt{Integrate}$ consists of all red tiles, each one accompanied by a table of overlapping active functions. This table in constructed as follows. For an element tile, concatenate its two integration tables. For a resolving tile, concatenate its coarser-link's integration tables. The set of red tiles, together with their associated integration tables, provides sufficient information to carry out the integration.

4A: Appendix to Chapter 4

4.5 Overview

In this appendix to Chapter 4 we prove the correctness of the (de)activation algorithms. We have chosen to use the hierarchical proof style advocated by Lamport [Lamport 1993] and by Gries [Gries and Schneider 1993]. Although some proofs are more verbose in this style, it is much harder to prove something which is false—this observation is at the heart of Lamport's argument for using this proof style to prove the correctness of algorithms. In a hierarchical proof, each proof step is itself proved by a nested sub-proof. The best way to read such a proof is breadth-first, from coarsest- to finest-level proofs.

We now recall and prove the (de)activation theorems and lemmas.

4.6 Correctness of Activation

Theorem 1 (activation) *Suppose: (1) $\overline{S}$ is consistent, (2) $\hat{\phi} \notin \overline{\mathcal{B}}$, and (3) $\mathcal{B} = \overline{\mathcal{B}} + \hat{\phi}$. S is consistent* ***iff*** *three conditions hold:*

1. $\mathcal{E} = \overline{\mathcal{E}} \cup \mathcal{S}(\hat{\phi})$

   In activating $\hat{\phi}$, we might activate elements.

2. $\mathcal{B}^s(\varepsilon) = \begin{cases} \overline{\mathcal{B}^s(\varepsilon)} + \hat{\phi} & \varepsilon \in \mathcal{S}(\hat{\phi}) \quad \textit{update required} \\ \overline{\mathcal{B}^s(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) \quad \textit{not affected} \end{cases}$

   In activating $\hat{\phi}$, we might update some native integration-tables.

3. $\mathcal{B}^a(\varepsilon) = \begin{cases} \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \mathcal{B}^s(\varepsilon') & \overline{\mathcal{B}^s(\varepsilon)} = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi}) & \textit{initialize} \\ \overline{\mathcal{B}^a(\varepsilon)} + \hat{\phi} & \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset \wedge \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') & \textit{update} \\ \overline{\mathcal{B}^a(\varepsilon)} & \mathcal{B}^s(\varepsilon) = \emptyset \vee \left(\overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset \wedge \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon')\right) & \textit{not affected} \end{cases}$

   In activating $\hat{\phi}$, we might update ancestral integration-tables.

PROOF SKETCH: We write out invariants I1, I2, and I3, then substitute $\mathcal{B} = \overline{\mathcal{B}} + \hat{\phi}$, and finally collect and isolate the quantities of the consistent $\overline{S}$ by using its invariants $\overline{\text{I1}}$, $\overline{\text{I2}}$, and $\overline{\text{I3}}$.

The proof of condition 3 is more elaborate (see step 3) and we sketch it here. We examine three separate cases: (step 3.2) newly-activated elements (their ancestral integration-table must be initialized), (step 3.3) already-active elements (they require an update if their ancestor supports $\hat{\phi}$), and (step 3.4) inactive elements (they remain untouched). First, (step 3.1) we establish two ways of referring to newly-activated elements—one way is better suited for implementation and the other for proofs.

ASSUME: 1. $\overline{S}$ is consistent,

2. $\hat{\phi} \notin \overline{\mathcal{B}}$,

3. $\mathcal{B} = \overline{\mathcal{B}} + \hat{\phi}$.

$\langle 1 \rangle$1. I1 $\Leftrightarrow \mathcal{E} = \overline{\mathcal{E}} \cup \mathcal{S}(\hat{\phi})$

To preserve I1, we might activate elements.

PROOF: I1 $\Leftrightarrow$
$$\begin{aligned}
\mathcal{E} &= \textstyle\bigcup_{\phi \in \mathcal{B}} \mathcal{S}(\phi) && \text{[invariant I1]} \\
&= \textstyle\bigcup_{\phi \in (\overline{\mathcal{B}} + \hat{\phi})} \mathcal{S}(\phi) && \text{[by assumption } \langle 0 \rangle.3] \\
&= \textstyle\bigcup_{\phi \in \overline{\mathcal{B}}} \mathcal{S}(\phi) \cup \mathcal{S}(\hat{\phi}) && \text{[simple algebra]} \\
&= \overline{\mathcal{E}} \cup \mathcal{S}(\hat{\phi}) && \text{[by invariant } \overline{\text{I1}} \text{ and assumption } \langle 0 \rangle.1] \ \square
\end{aligned}$$

$\langle 1 \rangle$2. I2 $\Leftrightarrow \mathcal{B}^s(\varepsilon) = \begin{cases} \overline{\mathcal{B}^s(\varepsilon)} + \hat{\phi} & \varepsilon \in \mathcal{S}(\hat{\phi}) \quad \textit{update required} \\ \overline{\mathcal{B}^s(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) \quad \textit{not affected} \end{cases}$

To preserve I2, we might update native integration-tables.

PROOF: I2 $\Leftrightarrow$
$$\begin{aligned}
\mathcal{B}^s(\varepsilon) &= \mathcal{B} \cap \mathcal{S}^\star(\varepsilon) && \text{[invariant I2]} \\
&= (\overline{\mathcal{B}} + \hat{\phi}) \cap \mathcal{S}^\star(\varepsilon) && \text{[by assumption } \langle 0 \rangle.3] \\
&= \overline{\mathcal{B}} \cap \mathcal{S}^\star(\varepsilon) + \{\hat{\phi}\} \cap \mathcal{S}^\star(\varepsilon) && \text{[by assumption } \langle 0 \rangle.2] \\
&= \overline{\mathcal{B}^s(\varepsilon)} + \{\hat{\phi}\} \cap \mathcal{S}^\star(\varepsilon) && \text{[by invariant } \overline{\text{I2}}] \\
&= \begin{cases} \overline{\mathcal{B}^s(\varepsilon)} + \hat{\phi} & \varepsilon \in \mathcal{S}(\hat{\phi}) \quad \textit{update required} \\ \overline{\mathcal{B}^s(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) \quad \textit{not affected} \end{cases} && \text{[by def. of adjoint]} \ \square
\end{aligned}$$

$\langle 1 \rangle$3. I3 $\Leftrightarrow \mathcal{B}^a(\varepsilon) = \begin{cases} \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \mathcal{B}^s(\varepsilon') & \overline{\mathcal{B}^s(\varepsilon)} = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi}) & \textit{initialize} \\ \overline{\mathcal{B}^a(\varepsilon)} + \hat{\phi} & \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset \wedge \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') & \textit{update} \\ \overline{\mathcal{B}^a(\varepsilon)} & \mathcal{B}^s(\varepsilon) = \emptyset \vee \left(\overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset \wedge \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon')\right) & \textit{not affected} \end{cases}$

To preserve I3, we might update ancestral integration-tables.

$\langle 2 \rangle$1. $\overline{\mathcal{B}^s(\varepsilon)} = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi}) \ \equiv \ \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} = \emptyset$

All previously-inactive elements in the natural support of $\hat{\phi}$ will be activated and initialized.

$\langle 3 \rangle$1. ASSUME: $\overline{\mathcal{B}^s(\varepsilon)} = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi})$

PROVE: $\mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} = \emptyset$

$\langle 4 \rangle$1. $\mathcal{B}^s(\varepsilon) \neq \emptyset$

PROOF: $\varepsilon \in \mathcal{S}(\hat{\phi})$ by assumption $\langle 3 \rangle$
$$\begin{aligned}
\Rightarrow \quad \mathcal{B}^s(\varepsilon) &= \overline{\mathcal{B}^s(\varepsilon)} + \hat{\phi} && \text{[by step } \langle 1 \rangle 2] \\
&\neq \emptyset && \text{[since } \hat{\phi} \notin \emptyset \wedge \hat{\phi} \in \mathcal{B}^s(\varepsilon)] \ \square
\end{aligned}$$

$\langle 4 \rangle$2. Q.E.D.

PROOF: by assumption $\langle 3 \rangle$ and step $\langle 4 \rangle$1. □

$\langle 3\rangle 2$. ASSUME: $\mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} = \emptyset$

PROVE: $\overline{\mathcal{B}^s(\varepsilon)} = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi})$

$\langle 4\rangle 1$. $\varepsilon \in \mathcal{S}(\hat{\phi})$

PROOF: $\mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} = \emptyset$ [by assumption $\langle 3\rangle$]

$\Rightarrow \quad \mathcal{B}^s(\varepsilon) \neq \overline{\mathcal{B}^s(\varepsilon)}$ [by algebra]

$\Rightarrow \quad \varepsilon \in \mathcal{S}(\hat{\phi})$ [by step $\langle 1\rangle 2$]

$\langle 4\rangle 2$. Q.E.D.

PROOF: by assumption $\langle 3\rangle$ and step $\langle 4\rangle 1$. □

$\langle 2\rangle 2$. CASE: $\overline{\mathcal{B}^s(\varepsilon)} = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi})$

PROVE: I3 $\Leftrightarrow \mathcal{B}^a(\varepsilon) = \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \mathcal{B}^s(\varepsilon')$

To preserve I3, we might initialize some ancestral-integration-tables.

$\langle 3\rangle 1$. $\mathcal{B}^s(\varepsilon) \neq \emptyset$ [by assumption $\langle 2\rangle$ and step $\langle 2\rangle 1$]

$\langle 3\rangle 2$. Q.E.D.

PROOF: by $\langle 3\rangle 1$ and invariant I3. □

$\langle 2\rangle 3$. CASE: $\mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset$

PROVE: I3 $\Leftrightarrow \mathcal{B}^a(\varepsilon) = \begin{cases} \mathcal{B}^a(\varepsilon) + \hat{\phi} & \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \\ \mathcal{B}^a(\varepsilon) & \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \end{cases}$

To preserve I3, we might update some integration tables.

PROOF:
$$\begin{aligned}
\text{I3} \Leftrightarrow \mathcal{B}^a(\varepsilon) &= \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \mathcal{B}^s(\varepsilon') && \text{[def'n of I3, assumption } \langle 2\rangle\text{]} \\
&= \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \left(\overline{\mathcal{B}^s(\varepsilon')} + \{\hat{\phi}\} \cap \mathcal{S}^\star(\varepsilon')\right) && \text{[by step } \langle 1\rangle 2\text{]} \\
&= \mathcal{B}^a(\varepsilon) + \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \{\hat{\phi}\} \cap \mathcal{S}^\star(\varepsilon') && \text{[by invariant } \overline{\text{I3}}\text{]} \\
&= \mathcal{B}^a(\varepsilon) + \{\hat{\phi}\} \cap \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \mathcal{S}^\star(\varepsilon') && \text{[by algebra]} \\
&= \begin{cases} \mathcal{B}^a(\varepsilon) + \hat{\phi} & \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \\ \mathcal{B}^a(\varepsilon) & \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \end{cases} && \text{[by definition of adjoint]} \; \square
\end{aligned}$$

$\langle 2\rangle 4$. CASE: $\mathcal{B}^s(\varepsilon) = \emptyset$

PROVE: I3 $\Leftrightarrow \mathcal{B}^a(\varepsilon) = \overline{\mathcal{B}^a(\varepsilon)}$

To preserve I3, some integration tables must remain unchanged.

$\langle 3\rangle 1$. $\overline{\mathcal{B}^s(\varepsilon)} = \emptyset$

PROOF:
$$\begin{aligned}
\overline{\mathcal{B}^s(\varepsilon)} &\subseteq \mathcal{B}^s(\varepsilon) && \text{[by } \langle 1\rangle 2\text{]} \\
&= \emptyset && \text{[by assumption } \langle 2\rangle\text{]}
\end{aligned}$$

$\langle 3\rangle 2$. Q.E.D.

PROOF:
$$\begin{aligned}
\text{I3} \Leftrightarrow \mathcal{B}^a(\varepsilon) &= \emptyset && \text{[by assumption } \langle 2\rangle \text{ and invariant I3]} \\
&= \overline{\mathcal{B}^a(\varepsilon)} && \text{[by step } \langle 3\rangle 1 \text{ and invariant } \overline{\text{I3}}\text{]} \; \square
\end{aligned}$$

$\langle 2\rangle 5$. Q.E.D.

$\langle 3\rangle 1$. Steps $\langle 2\rangle 2$, $\langle 2\rangle 3$, $\langle 2\rangle 4$ cover all possible cases.

PROOF: [case $\langle 2\rangle 2$ or $\langle 2\rangle 3$ or $\langle 2\rangle 4$]

$$\begin{aligned}
& \Big(\overline{\mathcal{B}^s(\varepsilon)} = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi})\Big) \vee \Big(\mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset\Big) \vee \mathcal{B}^s(\varepsilon) = \emptyset \\
\equiv\; & \Big(\mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} = \emptyset\Big) \vee \Big(\mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset\Big) \vee \mathcal{B}^s(\varepsilon) = \emptyset \quad \text{[by step } \langle 2\rangle 1\text{]} \\
\equiv\; & \mathcal{B}^s(\varepsilon) \neq \emptyset \vee \mathcal{B}^s(\varepsilon) = \emptyset \\
\equiv\; & \text{true} \;\Box
\end{aligned}$$

$\langle 3\rangle 2$. Q.E.D.

PROOF: by steps $\langle 2\rangle 2$, $\langle 2\rangle 3$, $\langle 2\rangle 4$, and $\langle 3\rangle 1$. $\Box$

$\langle 1\rangle 4$. Q.E.D.

PROOF: by steps $\langle 1\rangle 1$, $\langle 1\rangle 2$, and $\langle 1\rangle 3$. $\Box$

4.7 Correctness of Deactivation

Lemma 4 (active elements) *Given any system S, if invariant I2 holds, then*

I1 $\quad\equiv\quad \mathcal{B}^s(\varepsilon) \neq \emptyset \Leftrightarrow \varepsilon \in \mathcal{E}$.

Invariant I1: an element is active ***iff*** *its native integration-table is populated.*

PROOF:
$$\begin{aligned}
\varepsilon \in \mathcal{E} \quad \Leftrightarrow \quad & \mathcal{B}^s(\varepsilon) \neq \emptyset \\
= \quad & \mathcal{B} \cap \mathcal{S}^\star(\varepsilon) \neq \emptyset && \text{[by assumed Invariant I2]} \\
= \quad & \langle \vee\, \phi : \mathcal{B} \cap \mathcal{S}^\star(\varepsilon) \mid \varepsilon \in \mathcal{S}(\phi)\rangle && \text{[axiom of choice]}
\end{aligned}$$

Lemma 5 *Given set S and predicates R, $\overline{R}$,*

$$\big\{\varepsilon : {\sim}S \mid \overline{R}\big\} \cup \big(\big\{\varepsilon : S \mid \overline{R}\big\} - \{\varepsilon : S \mid R\}\big) \quad = \quad \big\{\varepsilon \mid \overline{R}\big\} - \{\varepsilon : S \mid R\}$$

Partition (of $\big\{\varepsilon \mid \overline{R}\big\}$) *then excision* (of $\{\varepsilon : S \mid R\}$) *is the same as unpartitioned excision.*

PROOF:
$$\begin{aligned}
& \big\{\varepsilon : {\sim}S \mid \overline{R}\big\} \cup \big(\big\{\varepsilon : S \mid \overline{R}\big\} - \{\varepsilon : S \mid R\}\big) \\
= \; & \big\{\varepsilon : {\sim}S \mid \overline{R}\big\} \cup \big(\big\{\varepsilon : S \mid \overline{R}\big\} \cap {\sim}\{\varepsilon : S \mid R\}\big) && \text{[def'n of minus]} \\
= \; & \big\{\varepsilon : {\sim}S \mid \overline{R}\big\} \cup \big(\big\{\varepsilon : S \mid \overline{R}\big\} \cap ({\sim}S \cup \{\varepsilon : S \mid \neg R\})\big) && \text{[def'n of } {\sim}\text{]} \\
= \; & \big(\big\{\varepsilon : {\sim}S \mid \overline{R}\big\} \cup \big\{\varepsilon : S \mid \overline{R}\big\}\big) \\
& \quad \cap \big(\big\{\varepsilon : {\sim}S \mid \overline{R}\big\} \cup {\sim}S \cup \{\varepsilon : S \mid \neg R\}\big) && \text{[distrib. } \cup \text{ over } \cap\text{]} \\
= \; & \big\{\varepsilon \mid \overline{R}\big\} \cap ({\sim}S \cup \{\varepsilon : S \mid \neg R\}) && \text{[def'n of } \cup\text{]} \\
= \; & \big\{\varepsilon \mid \overline{R}\big\} \cap {\sim}\{\varepsilon : S \mid R\} && \text{[def'n of } {\sim}\text{]} \\
= \; & \big\{\varepsilon \mid \overline{R}\big\} - \{\varepsilon : S \mid R\} && \text{[def'n of minus]} \;\Box
\end{aligned}$$

Theorem 2 (deactivation) *Suppose: (1) $\overline{S}$ is consistent, (2) $\hat{\phi} \in \overline{\mathcal{B}}$, and (3) $\mathcal{B} = \overline{\mathcal{B}} - \hat{\phi}$. S is consistent **iff** three conditions hold:*

1. $\mathcal{E} = \overline{\mathcal{E}} - \left\{\varepsilon \in \mathcal{S}(\hat{\phi}) \mid \mathcal{B}^s(\varepsilon) = \emptyset : \varepsilon\right\}$

   In deactivating $\hat{\phi}$, we might deactivate elements.

2. $\mathcal{B}^s(\varepsilon) = \begin{cases} \overline{\mathcal{B}^s(\varepsilon)} - \hat{\phi} & \varepsilon \in \mathcal{S}(\hat{\phi}) \quad \textit{update required} \\ \overline{\mathcal{B}^s(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) \quad \textit{not affected} \end{cases}$

   In deactivating $\hat{\phi}$, we might update some native integration-tables.

3. $\mathcal{B}^a(\varepsilon) = \begin{cases} \emptyset & \mathcal{B}^s(\varepsilon) = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi}) & \textit{clear} \\ \overline{\mathcal{B}^a(\varepsilon)} - \hat{\phi} & \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') & \textit{update} \\ \overline{\mathcal{B}^a(\varepsilon)} & \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \vee & \\ & \left(\mathcal{B}^s(\varepsilon) = \emptyset \wedge \varepsilon \notin \mathcal{S}(\hat{\phi})\right) & \textit{not affected} \end{cases}$

   In deactivating $\hat{\phi}$, we might update ancestral integration-tables.

$\langle 1 \rangle 1.$ I2 $\Leftrightarrow \mathcal{B}^s(\varepsilon) = \begin{cases} \overline{\mathcal{B}^s(\varepsilon)} - \hat{\phi} & \varepsilon \in \mathcal{S}(\hat{\phi}) \quad \textit{update required} \\ \overline{\mathcal{B}^s(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) \quad \textit{not affected} \end{cases}$

In deactivating $\hat{\phi}$, we might update some native integration-tables.

PROOF: I2 $\Leftrightarrow$

$$\begin{aligned}
\mathcal{B}^s(\varepsilon) &= \mathcal{B} \cap \mathcal{S}^\star(\varepsilon) && \text{[def'n of invariant I2]} \\
&= \left(\overline{\mathcal{B}} - \hat{\phi}\right) \cap \mathcal{S}^\star(\varepsilon) && \text{[by assumption } \langle 0 \rangle.3\text{]} \\
&= \left(\overline{\mathcal{B}} \cap \sim \{\hat{\phi}\}\right) \cap \mathcal{S}^\star(\varepsilon) && \text{[by def'n of minus op.]} \\
&= \overline{\mathcal{B}} \cap \left(\mathcal{S}^\star(\varepsilon) \cap \sim \{\hat{\phi}\}\right) && \text{[by assoc., commut. of } \cap\text{]} \\
&= \begin{cases} \overline{\mathcal{B}} \cap \mathcal{S}^\star(\varepsilon) \cap \sim \{\hat{\phi}\} & \hat{\phi} \in \mathcal{S}^\star(\varepsilon) \\ \overline{\mathcal{B}} \cap \mathcal{S}^\star(\varepsilon) & \hat{\phi} \notin \mathcal{S}^\star(\varepsilon) \end{cases} && [\hat{\phi} \notin \mathcal{S}^\star(\varepsilon) \Rightarrow \mathcal{S}^\star(\varepsilon) \subseteq \sim \{\hat{\phi}\}] \\
&= \begin{cases} \overline{\mathcal{B}^s(\varepsilon)} - \hat{\phi} & \hat{\phi} \in \mathcal{S}^\star(\varepsilon) \\ \overline{\mathcal{B}^s(\varepsilon)} & \hat{\phi} \notin \mathcal{S}^\star(\varepsilon) \end{cases} && \text{[by invariant } \overline{\text{I2}}\text{, def'n of minus op.]} \\
&= \begin{cases} \overline{\mathcal{B}^s(\varepsilon)} - \hat{\phi} & \varepsilon \in \mathcal{S}(\hat{\phi}) \\ \overline{\mathcal{B}^s(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) \end{cases} && \text{[by def'n of adjoint op.]} \ \square
\end{aligned}$$

$\langle 1 \rangle 2.$ ASSUME: invariant I2 holds

PROVE: I1 $\Leftrightarrow \mathcal{E} = \overline{\mathcal{E}} - \left\{\varepsilon \in \mathcal{S}(\hat{\phi}) \mid \mathcal{B}^s(\varepsilon) = \emptyset : \varepsilon\right\}$

In deactivating $\hat{\phi}$, we might deactivate elements.

$\langle 2 \rangle 1.$ $\left\{\varepsilon : \mathcal{S}(\hat{\phi}) \mid \mathcal{B}^s(\varepsilon) \neq \emptyset : \varepsilon\right\} = \left\{\varepsilon : \mathcal{S}(\hat{\phi}) \mid \overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset : \varepsilon\right\} - \left\{\varepsilon : \mathcal{S}(\hat{\phi}) \mid \mathcal{B}^s(\varepsilon) = \emptyset : \varepsilon\right\}$

$\langle 2 \rangle 2.$ Q.E.D.

PROOF: $$\begin{aligned} \text{I1} \equiv \quad \mathcal{E} &= \{\varepsilon \mid \mathcal{B}^s(\varepsilon) \neq \emptyset : \varepsilon\} && \text{[by Lemma 4]} \\ &= \left\{\varepsilon :\sim \mathcal{S}(\hat{\phi}) \mid \mathcal{B}^s(\varepsilon) \neq \emptyset : \varepsilon\right\} \\ &\quad \cup \left\{\varepsilon : \mathcal{S}(\hat{\phi}) \mid \mathcal{B}^s(\varepsilon) \neq \emptyset : \varepsilon\right\} && \text{[by union of complements]} \\ &= \left\{\varepsilon :\sim \mathcal{S}(\hat{\phi}) \mid \overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset : \varepsilon\right\} && \text{[by step ⟨1⟩1]} \\ &\quad \cup \left(\left\{\varepsilon : \mathcal{S}(\hat{\phi}) \mid \overline{\mathcal{B}^s(\varepsilon)} \neq \emptyset : \varepsilon\right\}\right. && \text{[step ⟨2⟩1]} \\ &\quad - \left.\left\{\varepsilon : \mathcal{S}(\hat{\phi}) \mid \mathcal{B}^s(\varepsilon) = \emptyset : \varepsilon\right\}\right) \\ &= \overline{\mathcal{E}} - \left\{\varepsilon : \mathcal{S}(\hat{\phi}) \mid \mathcal{B}^s(\varepsilon) = \emptyset : \varepsilon\right\} && \text{[by Lemma 5]} \ \square \end{aligned}$$

⟨1⟩3. ASSUME: invariant I2 holds

PROVE:

$$\text{I3} \Leftrightarrow \mathcal{B}^a(\varepsilon) = \begin{cases} \emptyset & \mathcal{B}^s(\varepsilon) = \emptyset \wedge \varepsilon \in \mathcal{S}(\hat{\phi}) & \textit{clear} \\ \overline{\mathcal{B}^a(\varepsilon)} - \hat{\phi} & \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') & \textit{update} \\ \overline{\mathcal{B}^a(\varepsilon)} & \mathcal{B}^s(\varepsilon) \neq \emptyset \wedge \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \vee \\ & \left(\mathcal{B}^s(\varepsilon) = \emptyset \wedge \varepsilon \notin \mathcal{S}(\hat{\phi})\right) & \textit{not affected} \end{cases}$$

In deactivating $\hat{\phi}$, we might update ancestral integration-tables.

⟨2⟩1. CASE: $\mathcal{B}^s(\varepsilon) = \emptyset$

PROVE: $$\text{I3} \Leftrightarrow \mathcal{B}^a(\varepsilon) = \begin{cases} \emptyset & \varepsilon \in \mathcal{S}(\hat{\phi}) & \textit{clear} \\ \overline{\mathcal{B}^a(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) & \textit{not affected} \end{cases}$$

⟨3⟩1. $\varepsilon \notin \mathcal{S}(\hat{\phi}) \Rightarrow \overline{\mathcal{B}^a(\varepsilon)} = \emptyset$

PROOF: $$\begin{aligned} \varepsilon \notin \mathcal{S}(\hat{\phi}) \quad \Rightarrow \quad \overline{\mathcal{B}^s(\varepsilon)} &= \mathcal{B}^s(\varepsilon) && \text{[by assumption ⟨1⟩ and step ⟨1⟩1]} \\ &= \emptyset && \text{[by assumption ⟨2⟩]} \\ \Rightarrow \quad \overline{\mathcal{B}^a(\varepsilon)} &= \emptyset && \text{[by assumption ⟨0⟩.1 }(\overline{\text{I3}})\text{ and case ⟨2⟩]} \ \square \end{aligned}$$

⟨3⟩2. Q.E.D.

PROOF: $$\begin{aligned} \text{I3} \Leftrightarrow \mathcal{B}^a(\varepsilon) &= \emptyset && \text{[by def'n of I3 with assumption ⟨2⟩]} \\ &= \begin{cases} \emptyset & \varepsilon \in \mathcal{S}(\hat{\phi}) \\ \overline{\mathcal{B}^a(\varepsilon)} & \varepsilon \notin \mathcal{S}(\hat{\phi}) \end{cases} && \text{[by step ⟨3⟩1]} \ \square \end{aligned}$$

⟨2⟩2. CASE: $\mathcal{B}^s(\varepsilon) \neq \emptyset$

PROVE: $$\text{I3} \Leftrightarrow \mathcal{B}^a(\varepsilon) = \begin{cases} \overline{\mathcal{B}^a(\varepsilon)} - \hat{\phi} & \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') & \textit{update} \\ \overline{\mathcal{B}^a(\varepsilon)} & \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') & \textit{not affected} \end{cases}$$

PROOF: $$\begin{aligned} \text{I3} \Leftrightarrow \mathcal{B}^a(\varepsilon) &= \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \mathcal{B}^s(\varepsilon') && \text{[I3 \& as.⟨2⟩]} \\ &= \bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \begin{cases} \overline{\mathcal{B}^s(\varepsilon')} - \hat{\phi} & \varepsilon' \in \mathcal{S}(\hat{\phi}) \\ \overline{\mathcal{B}^s(\varepsilon')} & \varepsilon' \notin \mathcal{S}(\hat{\phi}) \end{cases} && \text{[⟨1⟩1 \& as.⟨1⟩]} \\ &= \begin{cases} \left(\bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \overline{\mathcal{B}^s(\varepsilon')}\right) - \hat{\phi} & \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \\ \left(\bigcup_{\varepsilon' \in \mathcal{D}^\star(\varepsilon)} \overline{\mathcal{B}^s(\varepsilon')}\right) & \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \end{cases} && \text{[def'n: adjoint]} \\ &= \begin{cases} \overline{\mathcal{B}^a(\varepsilon)} - \hat{\phi} & \varepsilon \in \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \\ \overline{\mathcal{B}^a(\varepsilon)} & \varepsilon \notin \bigcup_{\varepsilon' \in \mathcal{S}(\hat{\phi})} \mathcal{D}(\varepsilon') \end{cases} && \text{[⟨2⟩ \& as.⟨0⟩.1]} \ \square \end{aligned}$$

⟨2⟩3. Q.E.D.

PROOF: by proofs of cases $\langle 2\rangle 1$ and $\langle 2\rangle 2$. $\square$

$\langle 1\rangle 4$. Q.E.D.

$\langle 2\rangle 1$. ASSUME: invariants I1, I2, and I3 hold.
PROVE: conditions 1, 2, and 3 hold.

PROOF: by steps $\langle 1\rangle 2$, $\langle 1\rangle 1$, and $\langle 1\rangle 3$. $\square$

$\langle 2\rangle 2$. ASSUME: conditions 1, 2, and 3 hold.
PROVE: invariants I1, I2, and I3 hold.

$\langle 3\rangle 1$. I2 holds.

PROOF: by step $\langle 1\rangle 1$. $\square$

$\langle 3\rangle 2$. I1 and I3 hold.

PROOF: by steps $\langle 3\rangle 1$, $\langle 1\rangle 2$, and $\langle 1\rangle 3$. $\square$

$\langle 3\rangle 3$. Q.E.D.

PROOF: by steps $\langle 3\rangle 1$ and $\langle 3\rangle 2$. $\square$

$\langle 2\rangle 3$. Q.E.D.

PROOF: by steps $\langle 2\rangle 1$ and $\langle 2\rangle 2$. $\square$

Chapter 5

Applications

Adaptive basis refinement may be profitably applied to many application domains, including simulation, animation, modeling, rendering, surgery, biomechanics, and computer vision. We present concrete, compelling examples based on our implementation of basis refinement. Our examples span thin shells (fourth order elliptic PDE using a Loop subdivision discretization), volume-deformation and stress-analysis using linear elasticity (second order PDE using linear-tetrahedral and trilinear-hexahedral finite elements) and a subproblem of electrocardiography (the generalized Laplace equation using linear tetrahedral finite elements).

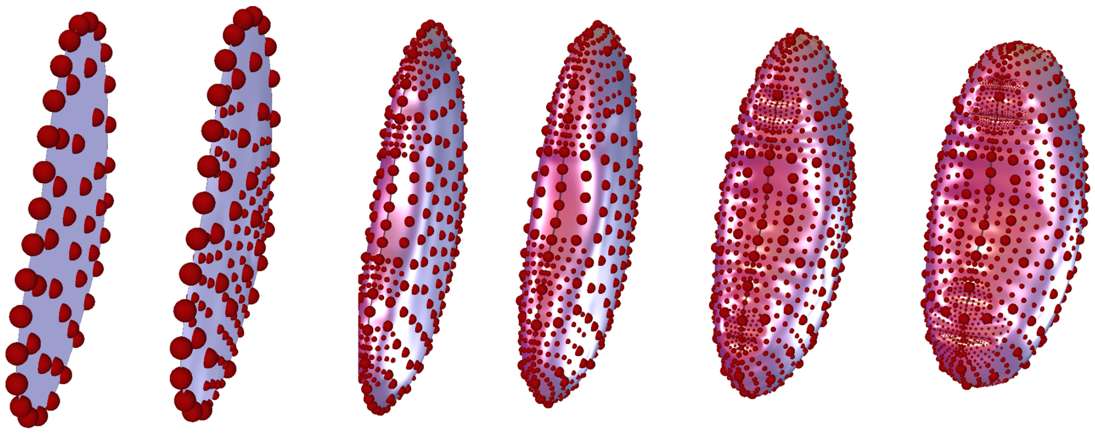


Figure 5.1: A sequence of frames from our adaptive simulation: time evolution of a rapidly inflating a metal-foil balloon. The initial model contains 50 basis functions. Over the coarse of the simulation, substitution refinement is used in regions of high bending energy. By the end of the simulation, 1000 basis functions are active across six levels of the hierarchy.

5.1 Overview

Adaptive basis refinement may be profitably applied to many application domains. Here are just a few:

Simulation Many mechanical simulation problems are posed as PDEs or integral equations and then approximated using a particular weighted residual method. The problem domains include thermodynamics (e.g., convective heat transfer) [Hughes 1987, Lewis et al. 1996], electromagnetism [Humphries 1997], quantum mechanics [Ram-Mohan 2002], and mechanics of continuous media [Malvern 1969, Kagan and Fischer 2000] including fluid flow as well as elasticity of solids, thin plates and thin shells (see Figure 5.1.

In computer graphics, research on elasticity traces back to the early work of Terzopoulos et al. [1987a], who introduced tensorial PDE treatments of elasticity to the graphics community. Later researchers developed treatments of thin plates such as cloth textiles [House and Breen 2000]. Recently, novel numerical treatments of thin shells, significantly more robust than earlier approaches, have been introduced in graphics [Green et al. 2002, Grinspun et al. 2002, Grinspun et al. 2003] and mechanics [Cirak et al. 2000a]. The computer graphics community has developed rapid techniques for fluid simulation based on the Navier-Stokes PDE for incompressible flow [Stam 1999], likewise the shallow water equations [Layton and van de Panne 2002] as well as level-set methods [Foster and Fedkiw 2001]. Attention has been given to the *visualization* of fluid flows and vector fields in general [Diewald et al. 2000]. These approaches to elasticity, fluid flow, and visualization are all based on PDE formulations; many of these approaches use finite-basis discretizations and thus easily accommodate basis refinement.

Animation We distinguish between *simulation*, which aims to *predict* physical behavior, and *animation*, which aims at *artistic control* of physical motion, often (but just as often not) with a secondary goal of physical plausibility. Animation problems are commonly formulated as PDEs. In his pioneering work on *spacetime constraints* Witkin [Witkin and Kass 1988] presented a variational formulation for constrained animation. These kinds of variational formulations benefit greatly (in performance and quality) by discretizing the spacetime ODE in multi-resolution and optimizing adaptively, as was demonstrated by [Liu et al. 1994] We return to this in our discussion of future research directions; see in particular the discussion of *multi-resolution model reduction* in Section 6.2. Recently, Capell et al. [2002a, 2002b] presented an interactive skeleton-based animation tool based on PDEs for linearized elasticity; they discuss their implementation of basis refinement, referring to our earlier work in this area [Grinspun et al. 2002].

Modeling An important class of shapes used in modeling are so-called *variationally optimal*, i.e., they minimize some energy functional corresponding to a variational formulation of some PDE. Welch and Witkin [1992] introduced variational formulations to the graphics modeling community; later Gortler and Cohen [1995] showed that such formulations can be solved efficiently using spline wavelet bases. In retrospect, we view Gortler and Cohen's work as a particular instance of basis refinement; our framework extends the implementation of these ideas, beyond the range of traditional wavelets, to arbitrary-topology surfaces based on multi-resolution subdivision discretizations. Geometric modeling continues to be an active research area [Terzopoulos et al. 1987b, Terzopoulos and Fleischer 1988, Celniker and Gossard 1991], [Metaxas and Terzopoulos 1992, Celniker and Welch 1992, Welch and Witkin 1992, Gortler and Cohen 1995, Kobbelt 2000b, Friedel et al. 2003].

Rendering In 1986 Kajia presented the *Rendering Equation* unifying the different models of rendering, e.g., ray-tracing [Foley et al. 1995], radiosity [Greenberg et al. 1986], etc. [Kajiya 1986]. Kajia showed that different discretizations of this integral equation lead to specific instances of earlier formulations, such as the *radiosity* equation which models radiative transport phenomena, leading under a finite-element discretization to a linear system. Gortler et al. [1993] demonstrated that wavelet discretizations of the radiosity equation enable adaptive solution radiosity. This earlier work fits neatly within our framework; with the tools presented herein we can generalize these ideas and their implementation from the setting of linear finite-elements and wavelets to the setting of high-genus surfaces with accompanying subdivision discretizations. Smoother discretizations offer potentially more compact representation of lighting; however, a critical issue to address would be sharp discontinuities in the radiance function. To that end, a complete treatment of adaptive radiosity computations based on subdivision discretizations should include a treatment of *edge tags* and their associated non-smooth basis functions [Biermann et al. 2000, DeRose et al. 1998, Zorin and Schröder 2000]; for further discussion of edge tags please see Section 6.2.

Surgery Problems in computational medicine are rapidly finding their way to the operating room. PDE formulations are everywhere. Consider, for example, PDE models of biological tissues such as the liver, blood, bones, tendons and grey matter [Roth et al. 1998, Cotin et al. 1996, Warfield et al. 2000, Wu et al. 2001]; PDE models of electromagnetism in the context of MRI (magnetic resonance imaging [Warfield et al. 2000]), CT (computerized tomography) [Christensen et al. 1997] and ECG (electrocardiogram [Johnson 2001]) scans. These models serve medical examination, training as well as surgery. Some examination applications analyze electromagnetic data obtained from ECG scans: this analysis may involve solving an inverse problem (see Section 5.5). Training and surgery applications are typically focused on modeling the deformation of body tissues due to external forces (see Section 5.3).

Biomechanics Similarly, engineers can consider interactions between their mechanical design and the body's tissues in a quantitative manner by measuring the stress and strain on the skeleton and muscles. This requires solving PDEs based on constitutive models of bones, muscles, ligaments and tendons. Applications include ergonomic design, construction of orthotics, design of sporting equipment, sports injury research.

Vision The field of computer vision is very broad; PDEs appear in problems from template matching to shape reconstruction. Most recently, Favaro presented a variational approach to *shape from defocus*, solving a PDE to globally reconstruct three-dimensional shape and radiance of a surface in space from images obtained with different focal settings [Jin and Favaro 2002].

We now turn to concrete examples, covering assorted application domains and discretizations, with: two- and three-dimensional domains; triangular, tetrahedral, and hexahedral tessellations; linear and trilinear finite-element as well as Loop subdivision bases; refinement by details as well as by substitution.

Together with Dr. Petr Krysl, we implemented these applications and demonstrated the efficacy of basis refinement. The author implemented and documented mainly the subdivision examples; Dr. Krysl implemented and documented mainly the finite-element examples; together both contributed to the software design and initial publications.

Although we have implemented these examples, our aim here is to provide a survey of the applications; to that end we have omitted those details which are best left to the original literature. The two-dimensional examples employ subdivision basis functions to simulate thin flexible structures including a balloon, a metallic cylinder, and a pillow. The three-dimensional 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.

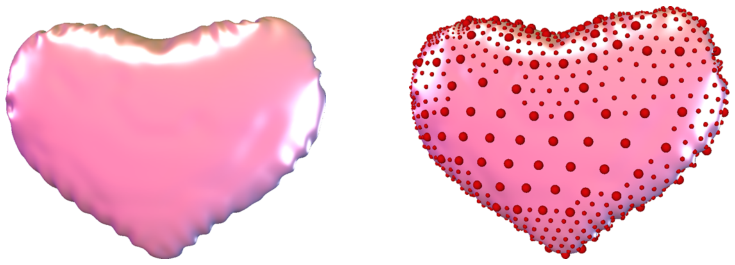


Figure 5.2: Thin shell simulation of inflating metal-foil balloon (left); red spheres represent active scaling functions (right). Note the concentration of finer functions near wrinkles and folds.

5.2 Non-Linear Mechanics of Thin Shells

The thin shell equations describe the behavior of thin flexible structures. Examples include aluminum 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. 5.1, 5.2 and 5.4).

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 a single-resolution Loop basis, the so-called *Subdivision Element Method* (SEM), and presented (non-adaptive) simulations supporting the claimed benefits of smooth subdivision discretizations [2000b, 2001].

Adaptivity is essential for efficiently modeling complex material phenomena such as wrinkling and buckling; such simulations were the original motivation behind the development of our framework. Here we present one static and two dynamic simulations that demonstrate the application of basis refinement to thin shells using Loop basis functions (please refer to our original publication [Grinspun et al. 2002] for videos of these simulations). Following our original publication of these results, Green [2002] demonstrated yet another benefit to using multi-resolution Loop discretizations of thin shell PDEs: *preconditioning* large systems, making tractable the simulation of highly complex models.

Inflating Balloon We simulated the dynamic behavior of a rapidly inflating metal-foil balloon (Fig. 5.2). 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 substitution refinement over the course of the 5ms simulated inflation.

Figure 5.3 shows the distribution of active nodes and elements 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 substitution as well as detail refinement to adapt the basis near the contact region.

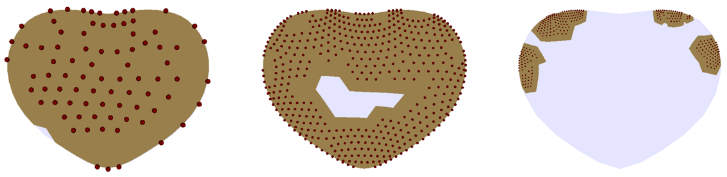


Figure 5.3: Visualization of the active nodes and elements at the end of inflation. The second through fourth levels of the six-level hierarchy are shown (left to right); the fourth and finer levels are sparsely populated.

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 aluminum cylinder under compression (Fig. 5.4). 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.

5.3 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 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 (single-resolution) 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. We extend this work by introducing a *multi*-resolution tetrahedral discretization and solving *adaptively* to maintain high accuracy.

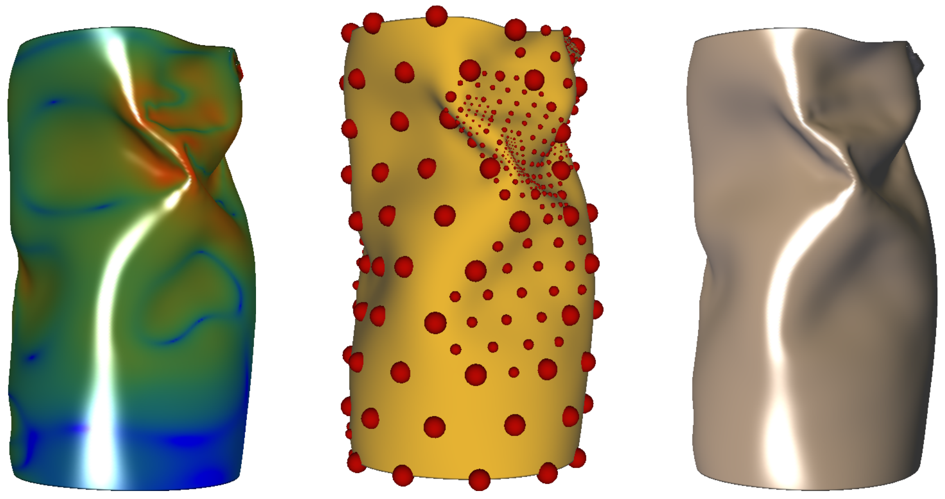


Figure 5.4: Thin shell simulation of a crushing cylinder. Refinement by substitution is used to introduce finer scaling functions: (left) regions with high bending force density are indicated in red; (middle) these regions have a high concentration of finer scaling functions, (right) consequently the animation captures the buckling mode and its sharp folds.

We modeled the volumetric deformation of the brain following the removal, or *resection*, 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.

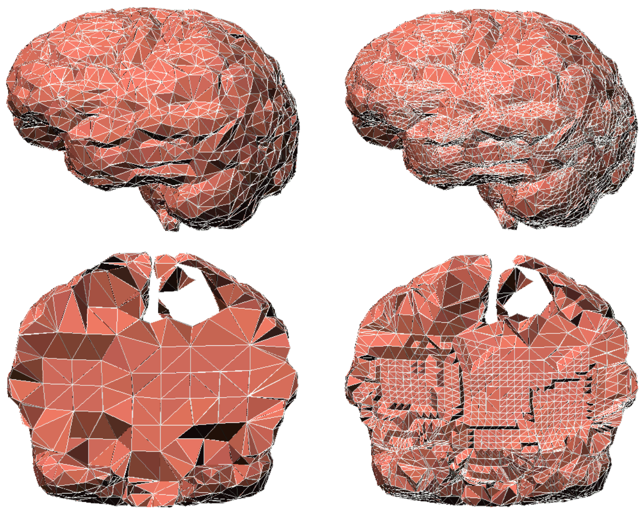


Figure 5.5: The initial (left) and refined (right) models of a brain after resection. Top row shows side view; bottom row shows dorsal view cross-section passing through the cavity of the resection.

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 by substitution to 64,905 DOFs, aiming for error equidistribution. Our error metric is the strain energy density. Figure 5.5 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 PI II laptop with 256MB: with a two- or four- CPU PC our simulation is fast enough for actual surgical interventions.

Figure 5.6 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

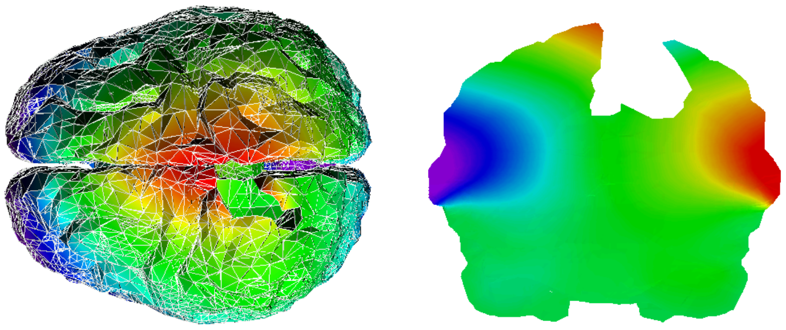


Figure 5.6: Refined FE model of the brain in a caudal view with color coded displacement amplitude (zero = purple; maximal = red). Notice the cavity resulting from the surgical resection of tissue in the left hemisphere. On the right color coded lateral displacement amplitude displayed on a dorsal cutting plane.

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.

5.4 Stress Distribution in Human Mandible

Numerical simulations are widely used in biomechanics to visualize response of skeletal structures to mechanical loads, as planning 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 this proof-of-concept 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 5.7 shows the original and refined FE model.

As our topological refinement operation we use 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 5.7, left). After refinement, placing details where strain-energy is high, the model captures the stress concentration immediately underneath the pressure point and in the thinner extremities of the mandible. It has approximately 4,200 DOFs (Fig. 5.7, right). We also ran this simulation with substitution-refinement with practically identical results. Figure 5.8 shows a close-up of the (detail-)refined 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.

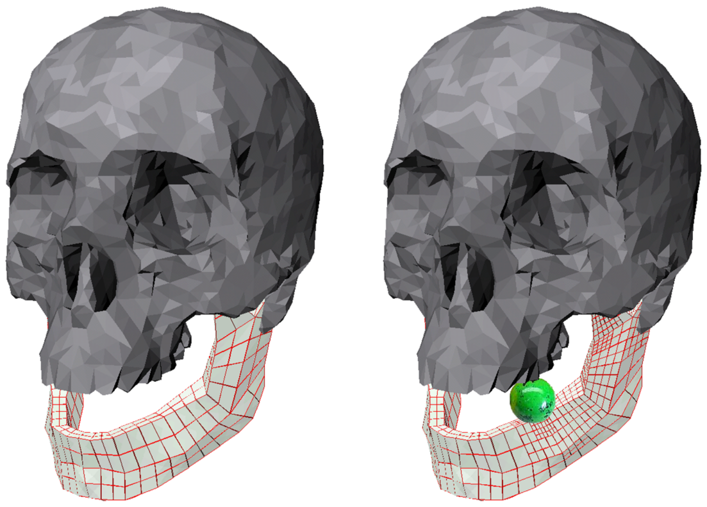


Figure 5.7: Adaptive refinement in response to stress. Unstressed mandible composited with skull (left); chewing on hard candy exerts a force on the jaw (right). The model (1,700 DOFs) is refined (4,200 DOFs) in the vicinity of the applied force as well as near the corner and attachment points.

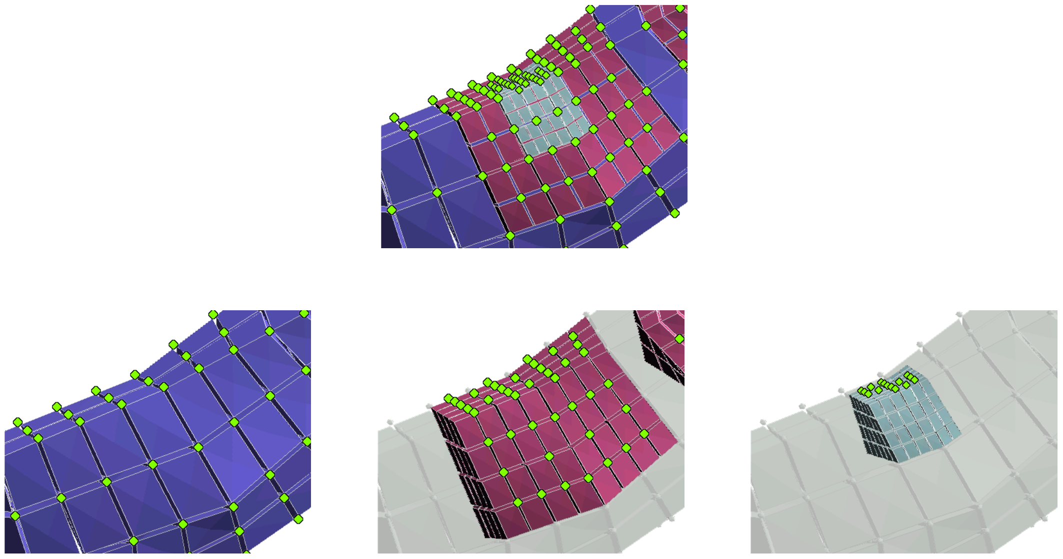


Figure 5.8: Mandible FE model with a multi-resolution basis of coarse-level scaling functions and finer-level details. Green dots indicate nodes associated with active basis functions. Top row: (integration) elements from all levels with active nodes; elements supporting detail functions on level 1 are colored blue. Bottom row: elements supporting detail functions on level 2 and 3 are respectively colored purple and tan. Note that elements at different levels overlap in space, and the detail functions active on the finer levels vanish along the internal boundaries of their supporting elements, thereby guaranteeing compatibility.

5.5 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 by Johnson [2001]. Here we show how basis refinement can be

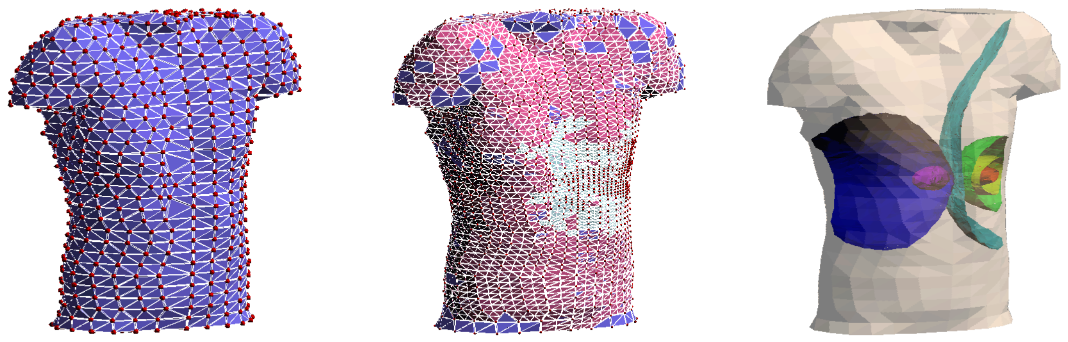


Figure 5.9: ECG field analysis example. On the left the initial grid; in the middle the refined discretization (four levels of substitution refinement). Red balls indicate active scaling functions, elements of different colors indicate the support of scaling functions on different levels. On the right, isocontours of the (adaptive) solution. Initial surface grid courtesy of the Center for Scientific Computing and Imaging, University of Utah.

applied to a subproblem of inverse ECG, the adaptive solution of the generalized Laplace equation,

$$\nabla \cdot \sigma \nabla \phi = 0 \,, \tag{5.1}$$

where σ is the conductivity tensor, with the boundary conditions

$$\phi = \phi_0 \ \text{ on } \ \Sigma \subset \Gamma_T \ \text{ and } \ \sigma \nabla \phi \cdot \mathbf{n} = 0 \ \text{ on } \ \Gamma_T \backslash \Sigma \,,$$

where Γ_T is the surface of the thorax, and ϕ_0 is given.

Figure 5.9 shows the initial grid with roughly 900 nodes and one of the refined grids with three levels of substitution refinement and a total of 8,500 nodes. The computed field is visualized through isopotential surfaces of a dipole located on the epicardial surface (assuming isotropic, homogeneous conductivity) in Figure 1.

Summary We demonstrated adaptive basis refinement using concrete example application drawn from engineering, surgical, and biomedical applications. For each example, we chose the multi-resolution discretization that best fit the problem; our choices spanned: two- and three-dimensional domains; triangular, tetrahedral, and hexahedral tesselations; linear and trilinear finite-element as well as Loop subdivision bases; refinement by details as well as by substitution.

Chapter 6

Conclusion

We have presented a natural method for adaptive refinement. Our mesh-based approach parallels mesh-less methods: the focus is the basis *not* tesselation, It unifies earlier instances of basis refinement, and generalizes these early works to the arbitrary topology, smoothness and accuracy setting. This chapter summarizes and further motivates our work, then describes exciting avenues for exploration, including links to asynchronous variational integration, model reduction, control systems, subdivision schemes with tags, novel error estimators, smoother unrefinement operators, traveling wavelets, hierarchical and multigrid preconditioners.

6.1 Summary and Closing Arguments

Many systems governed by PDEs have *singularities*, or nonsmooth solutions. For PDEs modeling physical phenomena this lack of smoothness is *inherently unavoidable*: we *wish* to capture phenomena such as wrinkles, buckles, boundary layers, shock waves and cracks. Two widely applicable classes of PDEs, elliptic equations on domains with re-entrant corners, and nonlinear hyperbolic systems of conservation laws, among others, in general have solutions exhibiting singularities.

Lack of smoothness is a severe obstacle to the convergence of numerical approximation methods, i.e., achieving a prescribed accuracy typically requires a finer discretization thus more computation in comparison with same same target accuracy solving for a smooth solution. In the context of wavelets, this observation has been made precise [Cohen 2003a]: a variety of smoothness classes can be characterized by the rate of decay of the multiscale approximation error:

$$\text{dist}(u(x), V^{(r)})_{L^p} \lesssim 2^{-sr} \ , \tag{6.1}$$

if and only if $u(x)$ has s derivatives in L^p, where p is the polynomial exactness (see Chapter 3) and $V^{(r)}$ is the level-r space of a multi-resolution wavelet discretization. We learn from this that the convergence of approximations to $u(x)$ at a prescribed rate, 2^{-sr}, as the discretization sampling rate *uniformly* increases, *requires* $u(x)$ to *everywhere* have smoothness $\geq s$. Convergence rate depends on smoothness; for uniform refinements, smoothness is measured by the lower bound on smoothness everywhere over the domain. For uniform refinement, a rough patch or singularity can be a *computational catastrophe.* For wavelet systems, DeVore has shown that for the same convergence rate, adaptive refinement (with appropriate error indicator) has a *significantly weaker smoothness requirement* [Cohen 2003a]. Thus we are motivated to implement *adaptive* solvers, which, *locally* refine the discretization, guided by a measurement of error. According to (6.1), error decays slowly in nonsmooth neighborhoods, consequently in striving to equidistribute error the solver naturally introduces a finer discretization in rough regions.

Existing adaptive approaches for mesh-based discretizations typically split mesh elements in isolation, violating compatibility conditions and then dealing with this problem using one of several techniques including constraints, Lagrange multipliers, or temporary irregular splitting of incident elements. These techniques must be specialized to different discretizations. This is in general *cumbersome* and in some cases *impossible*:

- For standard finite element basis functions, approaches to correcting incompatibility fail in higher dimensions. Consider refinement of a single element in a hexahedral mesh: it is impossible to split incident elements into finer hexahedra satisfying the compatibility conditions.

- Even when it is theoretically possible, element refinement can be so daunting that it is avoided albeit direly needed: consider, for example, refinement of four (or more!) dimensional spacetime discretizations, where one cannot easily visualize the (many!) permutations of incompatibilities that must be corrected.

- For discretizations carrying basis functions with support diameter greater than one-ring, among them the Subdivision Element Method, element refinement *does not lead to nested spaces* thus breaking refinement theory.

Mesh-less discretizations do not have such problems because there is no mesh thus no compatibility conditions! Without a mesh, the clear method for mesh-less (nested) refinement is to augment the spanning set. Bring this idea back to mesh-based discretizations: this is the key to avoiding the headaches associated with incompatibility.

Refining either a mesh-based or mesh-less discretization by augmenting the basis is, from a theorist's standpoint, an obvious idea. Starting with a finite-basis discretization, it is arguably the most direct means to nested spaces, and it does not require knowledge of whether the discretization is mesh-based or mesh-less. The question is how to carry this simplicity (and this enthusiasm!) to the *implementation* of adaptive *mesh-based* solvers.

Wavelets and hierarchical splines practitioners are familiar with building adaptive solvers that introduce basis functions. We view these earlier implementations as specific instances of basis refinement. Our contribution is a minimal structure which *unifies* spline, finite-element, (multi)wavelet, and subdivision discretizations among many others, laying out *general*, *provably-correct* algorithms that apply uniformly over any number of dimensions, for *mesh-based* discretizations with arbitrary smoothness, accuracy, tesselation, symmetry, etc.. Our algorithms maintain the set of basis functions which span (thus define) the approximation space, together with the associated structures required for numerical quadrature. For those applications which require a basis, we give additional structures which ensure linear independence of the spanning set.

The generality of the algorithms means easier implementation and debugging. While we were originally motivated by the need to find a refinement strategy for higher order basis functions, our method has significant advantages even when only piecewise linear basis functions are used. Building on his existing finite element solver software, Dr. Petr Krysl implemented and debugged our basic algorithms within a day. Due to the dimension independence of the algorithms, *extending the implementation to 2D and 3D problems took three hours each.* Although *mesh* refinement is popular for many kinds of finite elements, we learned that (a) finite-element *basis* refinement is very easy to implement, (b) existing finite element code is easily reused, and (c) spending time to implement basis refinement for a particular class of discretizations (e.g., finite elements) has the advantage that *once these data structures and algorithms are mastered they apply in a very broad setting.* The impact of basis refinement is conveyed in compelling applications spanning simulation of thin shells, biomechanical systems, and human tissue undergoing a surgical procedure. This forcefully attests to the simplicity and generality of the underlying framework.

6.2 Future Work

The work presented herein immediately invites exploration of:

Links to Asynchronous Variational Integration Explicit time-steppers for ODEs (with a time variable) and elliptic PDEs (with time and space variables) must respect a *stability limit* on the length of the time step; typically, this is the *CFL condition*, named after the three co-authors of [Courant et al. 1928], which checks that the temporal sampling frequency captures the highest frequencies of the discretized solution. For many spacetime PDEs, over *localized* regions of the domain the *spatial* frequency is proportional to the maximum local *temporal* frequency. Thus spatially adaptive discretizations do not in general have a spatially uniform stability limit. This motivates the *asynchronous* advancement of time over the domain, locally adopting shorter time intervals where the discretization is finer and larger time intervals where it is coarser.

Novel developments of *asynchronous variational integrators* (AVIs) are particularly promising [Lew et al. 2003a, Lew et al. 2003b]. This family of time-steppers improve both computational *performance* (via asynchrony) as well as *accuracy*: they are based on variational principles which guarantee that conserved quantities (e.g., energy, momentum) remain constant over the course of the simulation.

Explicit *synchronous* time-steppers for multi-resolution discretizations are limited by the highest spatial frequency: as refinement proceeds during the course of the simulation, this limit can be *globally* crippling. Fortunately, the adaptive discretizations produced by basis refinement are the ideal setting for demonstrating the power of AVIs. Preliminary investigations into, and discussions with the authors of, [Lew et al. 2003a] suggest that basis refinement and AVIs are *algorithmically and theoretically orthogonal*: we believe that these two techniques can (and should!) be used in conjunction.

Links to Model Reduction and Control Systems Control of physical systems is critical to many engineering and graphics applications [Dullerud and Paganini 2000, Hodgins et al. 1995]. In designing controllers for complex systems, the control problem is made tractable by *modeling* the target system at a sufficiently *coarse* granularity. *Model reduction* is the problem of capturing the salient, coarse features of a model faithfully, discarding the remainder. This problem has been studied extensively in the setting of *linear* input-output systems, where techniques such as *balanced truncation* (also known as *principle component analysis*) are employed [Moore 1981]. Recently, Lall and co-workers [2002] extended these techniques to *nonlinear* input-output systems. Their approach uses empirical (or simulated) data to identify the dynamic behavior most relevant to the input-output map of the system, i.e., they identify those states of the system which are not affected by actuators (i.e., feedback) and which most affect the sensors. They then apply a Galerkin projection to capture at a coarse grain the identified input-output map, producing a *nonlinear* reduced-order model with inputs and outputs especially suited for control applications.

Model reduction has exciting applications in many fields. For example, Popovic has demonstrated the use of variationally *optimized* controllers in animation. We believe that these approaches could benefit from basis refinement. The model reduction and control optimization stages could be repeated after an adaptive refinement of the Galerkin discretization, using the performance evaluation of the optimized controller to design an error indicator. This could potentially create multi-resolution reduced-models with very few inputs and outputs at the coarsest level.

Novel Error Estimators In our implementation we adopted well-established *a posteriori* error indicators. Since these give per-*element* errors, and we need to make (de)activation decisions per-*function*, we must convert: we *lump* the element error onto the basis functions, e.g., the error associated to $\phi_i(x)$ is $\int_\Omega \phi_i(x)\gamma(x)dx$, where $\gamma(x)$ is the piecewise constant *error density*, computed via equidistribution of elemental error over the elemental subdomain. This appears to be an effective technique, and it has the benefit that we may adopt any element-based error measure. However, we would be more satisfied with a posteriori error estimators which focus directly on the basis functions, i.e., estimate the change in error due to (de)activating a particular basis function. Later, it would be very desirable to build a mathematical framework for transforming per-element derivations into per-function derivations.

Links to Hierarchical and Multigrid Preconditioners Preconditioning techniques are critical for accelerating the convergence of iterative solvers. Recently Green demonstrated the preconditioning of problems which use Cirak's subdivision elements [Cirak et al. 2000a]; Green exploits the multi-resolution of subdivision discretizations to great effect, reporting encouraging results for the preconditioning of an elliptic PDE based on the Kirchoff-Love formulation of thin shells [Green et al. 2002]. Since multi-resolution discretizations lie at the heart of our method, the necessary structure exists to accommodate multigrid [Bank et al. 1988, Briggs et al. 2000] or wavelet [Cohen and Masson 1997] preconditioning following the example presented by Green.

Advances in Interpolated Unrefinement Unrefinement is inherently *lossy*. It requires projection onto a smaller approximation space. In general, this leads to undesirable artifacts in simulation applications, in particular temporal discontinuities in the configuration at the point in time that unrefinement occurs. One way to resolve this is to permit unrefinement only when the current approximated solution is contained in the smaller, unrefined trial space. This is lossless, but performance crippling —we might never be allowed to unrefine!— in general we expect there to be *some* (small but non-zero) error during refinement.

To remedy this, our implementation uses *interpolated unrefinement*. A basis function is *smoothly deactivated*: the deactivation of $\phi_i(x)$ is separated into two steps: (a) the solver no longer considers as unknown the associated coefficient, u_i, and (b) the coefficient is set to zero, i.e., $u_i \to 0$. In *immediate* (nonsmooth) deactivation, both of these steps are performed instantaneously. In smooth deactivation, (a) the unknown, u_i, is (as always) *immediately* removed from the solver's reach, i.e., it is considered a prescribed boundary condition, and (b) the value of the coefficient is prescribed by a *smoothly decaying* function over some finite time interval. e.g., $u_i(t) \to 0$ as $t \to t_0 + \Delta t_{\text{dd}}$, where t_0 is the time at which deactivation commences and

Δt_{dd} is the *deactivation delay*. In our implementation we chose a simple linear decay, leading to continuity over time of a configuration but *not* its velocity. This removed the visual "blips" from the animation; however keen observers note that the discontinuity in velocity is noticeable and undesirable—human observers are naturally trained to recognize sudden changes in *velocity* since these are associated with impulses. To that end, it may be desirable to use decay curves which have zero initial and final time derivatives, such that continuity of velocity is preserved during interpolated unrefinement. In contrast, we believe that continuity of acceleration is not equally important, since many physical systems have discontinuous accelerations.

Links to Advected Bases and Traveling Wavelets Adaptive basis refinement may be of benefit in the simulation of turbulent fluid flows. These problems are characterized by *advection* phenomena, i.e., the solution's features are moving over the domain with some velocity. Consider that an advection operator, to first order, locally *translates* pieces of the solution from one region of spatial domain to another. There is beauty in approximating the advected solution using *advected basis functions*, i.e., let the advection operator be a map from one approximation space *not* onto itself, rather to an *advected space*. This space is formed as follows. Place particles at the center of the original approximation space. Advect the particles, and build a set of advected basis functions centered at the advected particles. This approach is inspired by earlier work on *traveling wavelets* [Perrier and Basdevant 1991].

Links to Tagged Subdivision: Multi-nesting Relations An attractive attribute of subdivision schemes is their support for *tagged meshes*, which associate descriptive *tags* to each mesh entity, e.g., *sharp crease* edge, *corner* vertex, *flat* face (see, e.g., Biermann's use of tags with Loop subdivision [Biermann et al. 2000]). Tags modify the subdivision stencil *in the vicinity* of the tagged entity. For example, *sharp crease* tags modify the subdivision stencil to prevent smoothing (or otherwise propagating information) across the two surface patches incident to the tagged edge. This gives significant control over the shape of the limit function, most often over its *smoothness*. Sharp creases, for example, induce discontinuous derivatives across the associated patch boundaries. Recently DeRose introduced more general *semi-sharp creases*, which under the control of a real-valued *sharpness index* induce arbitrarily large but bounded derivatives across patch boundaries [DeRose et al. 1998].

The possibility of decorating a mesh with tags creates an expanded family of subdivision scaling functions associated to every (local) permutation of tags. With the introduction of parameterized tags (e.g., sharpness indices) an *infinite* set of scaling functions (corresponding to different values of the parameter) associates to a single mesh entity. Every space $V^{(p)}$ is now richer; subdivision theory guarantees that the nesting relation still holds, i.e., $V^{(p)} \subset V^{(p+1)}$. Furthermore, we may be able to generalize the nesting relation, defining a multi-dimensional nesting (*multi-nesting*) relation: let the space $V^{(p,s)}$ be spanned by all level-p scaling functions associated to meshes with arbitrary sharpness indices $< s$. We view this space as a point in a two dimensional space-of-spaces, i.e., $V^{(p,s)} \in \mathcal{V}$, with a discrete first dimension ($p \in \mathbb{Z}^*$) and continuous second dimension ($s \in \mathbb{R},\ s \geq 0$). Then $V^{(p,s)}$ is nested in its first dimension, i.e., $V^{(p,s)} \subset V^{(p+1,s)}$, as well as its second, i.e., $V^{(p,s)} \subset V^{(p,t)},\ s < t$. Refinement may naturally extend the current approximation space along any of the nesting directions, e.g., adding *finer* functions, or adding *sharper* functions.

Simulations of crushing, buckling and wrinkling benefit from adaptive approaches because they have phenomenological singularities. When the only approach to capturing these singularities is to make the discretization locally finer (i.e., so-called h-refinement [Eriksson et al. 1996]), the result is the introduction of numerous extremely fine scaling functions. While *adaptivity* avoids *globally* switching to a finer discretization, there is still a computational penalty near sharp (less smooth) features of the solution. Our hope is that the ability to introduce better-suited (e.g., sharper) scaling functions fitted to the features of the solution may significantly reduce the need for exceedingly-fine discretizations.

Links to Tagged Subdivision: Scaling Functionals Alternatively, mesh tag parameters (such as the sharpness index) may be viewed as *parameters* of a scaling functional, i.e., the trial space consists of all functions $\sum \phi_i(u_i, s_i)(x)$, where the scaling *functionals*, $\phi_i(u_i, s_i) \in H(\Omega)$, are linear in their first argument (e.g., *displacement*) but not their second (e.g., *sharpness*). In general, the functional may take several sharpness arguments, i.e., $\phi_i(u_i, s_{i,1}, s_{i,2}, \ldots, s_{i,N})$, corresponding to the N edges within its support. In this approach there are *multiple* coefficients associated to a single scaling functional. Of all formulations, it may be that the variational form is best-suited for dealing with this setting. It remains to be seen whether this is a useful construction; for both applications of tagged subdivision, we are inspired by the success of p-refinement [Eriksson et al. 1996], *ridgelets* [Candès 1998] and *curvelets* [Candès and Donoho 2000].

6.3 Conclusion

Our contributions are (a) a mathematical framework with (b) associated algorithms for basis refinement; furthermore, we (c) describe the mapping of popular methods —finite-elements, wavelets and multiwavelets, splines and subdivision schemes— onto this framework, and (d) we demonstrate working implementations of basis refinement with applications in graphics, engineering, and medicine, including in particular adaptive computation of thin shell problems using subdivision elements, for which classical finite element mesh refinement does not apply.

The core idea is to refine basis functions, not elements. This idea is made concrete by starting with a *hierarchy* of nested approximation spaces, equivalently a refinement relation. By construction, this idea leads to methods which are naturally *conforming*: unlike mesh refinement, basis refinement never creates incompatible meshes. Together, the hierarchical structure and natural compatibility give rise to simple *adaptive refinement* algorithms which make no assumptions as to (a) the dimension of the domain; (b) the tesselation of the domain; (c) the approximation smoothness or accuracy; and (d) the support diameter of the basis functions. This simple idea, basis refinement, is at the heart of a unifying, general class of *conforming, hierarchical, adaptive refinement methods*, briefly CHARMS.

Bibliography

[Alpert 1993] 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 et al. 2001] ARNOLD, D. N., MUKHERJEE, A., AND POULY, L. 2001. Locally Adapted Tetrahedral Meshes using Bisection. *SIAM Journal on Scientific Computing 22*, 2, 431–448.

[Atkinson 1989] ATKINSON, K. 1989. *An Introduction to Numerical Analysis*, 2nd edition ed. John Wiley & Sons, Inc.

[Azar et al. 2001] 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.

[Babuška et al. 1986] BABUŠKA, I., ZIENKIEWICZ, GAGO, J., AND OLIVEIRA, E. R. D., Eds. 1986. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations (Wiley Series in Numerical Methods in Engineering)*. John Wiley & Sons.

[Babuška et al. 2002] BABUŠKA, I., BANERJEE, U., AND OSBORN, J. 2002. Meshless and Generalized Finite Element Methods: A Survey of Some Major Results. Tech. Rep. 114 pp. TICAM Report 02-03, Texas Inst. for Comp. Eng. and Sci.

[Bajaj et al. 2002a] BAJAJ, C., SCHAEFER, S., WARREN, J., AND XU, G. 2002. A subdivision scheme for hexahedral meshes. *The Visual Computer 18*, 5/6, 343–356.

[Bajaj et al. 2002b] BAJAJ, C., WARREN, J., AND XU, G. 2002. A Smooth Subdivision Scheme for Hexahedral Meshes. *The Visual Computer*. to appear.

[Bank and Xu 1996] BANK, R., AND XU, J. 1996. An Algorithm for Coarsening Unstructured Meshes. *Numerische Mathematik 73*, 1, 1–36.

[Bank et al. 1988] BANK, R., DUPONT, T., AND YSERENTANT, H. 1988. The Hierarchical Basis Multigrid Method. *Numerische Mathematik 52*, 4, 427–458.

[Bernard and Wallace 2002] BERNARD, P. S., AND WALLACE, J. M. 2002. *Turbulent Flow: Analysis, Measurement and Prediction*. John Wiley & Sons.

[Bey 1995] Bey, J. 1995. Tetrahedral Grid Refinement. *Computing 55*, 355–378.

[Bey 2000] Bey, J. 2000. Simplicial Grid Refinement: On Freudenthal's Algorithm and the Optimal Number of Congruence Classes. *Numerische Mathematik 85*, 1–29.

[Biermann et al. 2000] Biermann, H., Levin, A., and Zorin, D. 2000. Piecewise Smooth Subdivision Surfaces with Normal Control. *Proceedings of ACM SIGGRAPH 2000*, 113–120.

[Brezzi and Fortin 1991] Brezzi, F., and Fortin, M. 1991. *Mixed and Hybrid Finite Element Methods*. Springer.

[Briggs et al. 2000] Briggs, W. L., Henson, V. E., and McCormick, S. F. 2000. *A Multigrid Tutorial*. SIAM.

[Buhmann 2003] Buhmann, M. D. 2003. *Radial Basis Functions*. Cambridge University Press.

[Candès and Donoho 2000] Candès, E. J., and Donoho, D. L. 2000. Curvelets, Multiresolution Representation, and Scaling Laws. In *Wavelet Applications in Signal and Image Processing VIII*, Proc. SPIE 4119, A. Aldroubi, A. F. Laine, and M. A. Unser, Eds.

[Candès 1998] Candès, E. J. 1998. *Ridgelets: Theory and Applications*. PhD thesis, Department of Statistics, Stanford University.

[Capell et al. 2002a] Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. 2002. Interactive Skeleton-Driven Dynamic Deformations. *ACM Transactions on Graphics 21*, 3, 586–593.

[Capell et al. 2002b] Capell, S., Green, S., Curless, B., Duchamp, T., and Popovic, Z., 2002. A Multiresolution Framework for Dynamic Deformations.

[Carey 1997] Carey, G. F. 1997. *Computational Grids: Generation, Adaptation and Solution Strategies*. Taylor & Francis.

[Catmull and Clark 1978] Catmull, E., and Clark, J. 1978. Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes. *Computer Aided Design 10*, 6, 350–355.

[Celniker and Gossard 1991] 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.

[Celniker and Welch 1992] Celniker, G., and Welch, W. 1992. Linear Constraints for Deformable B-spline Surfaces. *1992 Symposium on Interactive 3D Graphics 25*, 2, 165–170.

[Chen and Zeltzer 1992] Chen, D. T., and Zeltzer, D. 1992. Pump it Up: Computer Animation of a Biomechanically Based Model of Muscle Using the Finite Element Method. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2, 89–98.

[Christensen et al. 1997] CHRISTENSEN, G., JOSHI, S., AND MILLER, M., 1997. Volumetric transformation of brain anatomy.

[Cirak and Ortiz 2001] CIRAK, F., AND ORTIZ, M. 2001. Fully C^1-Conforming Subdivision Elements for Finite Deformation Thin-Shell Analysis. *Internat. J. Numer. Methods Engrg. 51*, 7, 813–833.

[Cirak et al. 2000a] CIRAK, F., ORTIZ, M., AND SCHRÖDER, P. 2000. Subdivision Surfaces: A New Paradigm for Thin-Shell Finite-Element Analysis. *Internat. J. Numer. Methods Engrg. 47*, 12, 2039–2072.

[Cirak et al. 2000b] CIRAK, F., ORTIZ, M., AND SCHRÖDER, P. 2000. Subdivision Surfaces: A New Paradigm for Thin-Shell Finite-Element Analysis. *Internat. J. Numer. Methods Engrg. 47*, 12, 2039–2072.

[Cohen and Masson 1997] COHEN, A., AND MASSON, R., 1997. Wavelet methods for second order elliptic problems—-preconditioning and adaptivity.

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

[Cohen 2003a] COHEN, A. 2003. *Numerical Analysis of Wavelet Methods*. Elsevier.

[Cohen 2003b] COHEN, A., 2003. Personal communications.

[Cotin et al. 1996] COTIN, S., DELINGETTE, H., AND AYACHE, N. 1996. Real Time Volumetric Deformable Models for Surgery Simulation. In *VBC*, 535–540.

[Courant et al. 1928] COURANT, R., FRIEDRICHS, K., AND LEWY, H. 1928. Über die partiellen differenzengleichungen der mathematischen physik. *Math. Ann.*, 100, 32.

[Crouzeix and Raviart 1973] CROUZEIX, M., AND RAVIART, P. 1973. Conforming and nonconforming finite element methods for solving the stationary stokes equations, part I. *RAIRO*, R-3, 33–76.

[Cutler et al. 2002] CUTLER, B., DORSEY, J., MCMILLAN, L., MÜLLER, M., AND JAGNOW, R. 2002. A Procedural Approach to Authoring Solid Models. *ACM Transactions on Graphics 21*, 3, 302–311.

[Daubechies 1992] DAUBECHIES, I. 1992. *Ten Lectures on Wavelets (Cbms-Nsf Regional Conference Series in Applied Mathematics, No 61)*. SIAM.

[de Boor et al. 1993] DE BOOR, C., HÖLLIG, K., AND RIEMENSCHNEIDER, S. 1993. *Box Splines*. Springer-Verlag New York Inc., New York, NY.

[Debunne et al. 2001] DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. ”Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling”. *Proceedings of ACM SIGGRAPH 2001*, 31–36.

[DeRose et al. 1998] DeRose, T., Kass, M., and Truong, T. 1998. Subdivision Surfaces in Character Animation. *Proceedings of SIGGRAPH 98*, 85–94.

[DeVore 1998] DeVore, R. 1998. Nonlinear Approximation. *Acta Numerica*, 7, 51–150.

[Diewald et al. 2000] Diewald, U., Preusser, T., and Rumpf, M. 2000. Anisotropic Diffusion in Vector Field Visualization on Euclidean Domains and Surfaces. *IEEE Transactions on Visualization and Computer Graphics 6*, 2, 139–149.

[Divo and Kassab 2003] Divo, E., and Kassab, A. J. 2003. *Boundary Element Methods for Heat Conduction with Applications in Non-Homogeneous Media*. WIT Press.

[Doo and Sabin 1978] Doo, D., and Sabin, M. 1978. Analysis of the Behaviour of Recursive Division Surfaces near Extraordinary Points. *Computer Aided Design 10*, 6, 356–360.

[Dullerud and Paganini 2000] Dullerud, G. E., and Paganini, F. 2000. *A Course in Robust Control Theory: A Convex Approach*. Texts in Applied Mathematics 36. Springer-Verlag, New York.

[Dyn and Levin 2002] Dyn, N., and Levin, D. 2002. Subdivision Schemes in Geometric Modelling. In *Acta Numerica*, A. Iseries, Ed., vol. 11. Cambridge University Press, ch. 2.

[Dyn et al. 1990] Dyn, N., Levin, D., and Gregory, J. A. 1990. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM Transactions on Graphics 9*, 2, 160–169.

[Eriksson et al. 1996] Eriksson, K., Estep, D., Hansbo, P., and Johnson, C. 1996. *Computational Differential Equations*. Cambridge University Press.

[Field 1995] Field, D. A. 1995. The legacy of automatic mesh generation from solid modeling. *Computer Aided Geometric Design 12*, 7, 651–673.

[Foley et al. 1995] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. 1995. *Computer Graphics: Principles and Practice in C*, 2nd ed. Addison-Wesley.

[Forsey and Bartels 1988] Forsey, D. R., and Bartels, R. H. 1988. Hierarchical B-Spline Refinement. *Computer Graphics (Proceedings of SIGGRAPH 88) 22*, 4, 205–212.

[Foster and Fedkiw 2001] Foster, N., and Fedkiw, R. 2001. ”Practical Animation of Liquids”. *Proceedings of SIGGRAPH 2001*, 23–30.

[Friedel et al. 2003] Friedel, I., Mullen, P., and Schröder, P. 2003. Data-Dependent Fairing of Subdivision Surfaces. In *Proceedings of the 8th ACM Symposium on Solid Modeling and Applications 2003*.

[Gortler and Cohen 1995] Gortler, S. J., and Cohen, M. F. 1995. Hierarchical and Variational Geometric Modeling with Wavelets. *1995 Symposium on Interactive 3D Graphics*, 35–42.

[Gortler et al. 1993] GORTLER, S. J., SCHRÖDER, P., COHEN, M. F., AND HANRAHAN, P. 1993. Wavelet Radiosity. *Proceedings of SIGGRAPH 93*, 221–230.

[Gourret et al. 1989] 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.

[Green et al. 2002] GREEN, S., TURKIYYAH, G., AND STORTI, D. 2002. Subdivision-Based Multilevel Methods for Large Scale Engineering Simulation of Thin Shells. In *Proceedings of ACM Solid Modeling*, 265–272.

[Greenberg et al. 1986] GREENBERG, D. P., COHEN, M., AND TORRANCE, K. E. 1986. Radiosity: A Method for Computing Global Illumination. *The Visual Computer 2*, 5, 291–297.

[Gries and Schneider 1993] GRIES, D., AND SCHNEIDER, F. B. 1993. *A Logical Approach to Discrete Math (Texts and Monographs in Computer Science)*, 3rd ed. Springer Verlag.

[Grinspun et al. 2002] GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Transactions on Graphics 21*, 3, 281–290.

[Grinspun et al. 2003] GRINSPUN, E., HIRANI, A., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete Shells. In *ACM SIGGRAPH Symposium on Computer Animation*. to appear.

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

[Harten 1993] HARTEN, A. 1993. Discrete Multiresolution and Generalized Wavelets. *J. Appl. Num. Math 12*, 153–193.

[Harten 1996] HARTEN, A. 1996. Multiresolution Representation of Data II: Generalized Framework. *SIAM J. Numer. Anal. 33*, 1205–1256.

[Hodgins et al. 1995] HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating Human Athletics. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 71–78.

[Hoppe 1996] HOPPE, H. 1996. Progressive Meshes. *Proceedings of SIGGRAPH 96*, 99–108.

[House and Breen 2000] HOUSE, D. H., AND BREEN, D. E., Eds. 2000. *Cloth Modeling and Animation*. A. K. Peters.

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

[Humphries 1997] HUMPHRIES, JR., S. 1997. *Field Solutions on Computers*. CRC Press.

[Jin and Favaro 2002] JIN, H., AND FAVARO, P. 2002. A Variational Approach to Shape from Defocus. In *7th European Conference on Computer Vision (ECCV 2002) (Part II)*, 18–30.

[Johnson 2001] 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 and Fischer 2000] KAGAN, P., AND FISCHER, A. 2000. Integrated mechanically based CAE system using B-Spline finite elements. *Computer Aided Design 32*, 8-9, 539–552.

[Kajiya 1986] KAJIYA, J. T. 1986. The Rendering Equation. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, vol. 4(20), 143–150.

[Keeve et al. 1996] KEEVE, E., GIROD, S., PFEIFLE, P., AND GIROD, B. 1996. Anatomy-Based Facial Tissue Modeling Using the Finite Element Method. *IEEE Visualization '96*, 21–28.

[Kobbelt 2000a] KOBBELT, L. 2000. $\sqrt{3}$ Subdivision. *Proceedings of SIGGRAPH 2000*, 103–112.

[Kobbelt 2000b] KOBBELT, L. P. 2000. Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer 16*, 3-4, 142–150.

[Kober and Muller-Hannemann 2000] 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*, 423–434.

[Koch et al. 1996] KOCH, R. M., GROSS, M. H., CARLS, F. R., VON BÜREN, D. F., FANKHAUSER, G., AND PARISH, Y. 1996. Simulating Facial Surgery Using Finite Element Methods. *Proceedings of SIGGRAPH 96*, 421–428.

[Kraft 1997] 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.

[Kry et al. 2002] KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. EigenSkin: Real Time Large Deformation Character Skinning in Hardware. In *ACM SIGGRAPH Symposium on Computer Animation*, 153–160.

[Krysl et al. 2003] KRYSL, P., GRINSPUN, E., AND SCHRÖDER, P. 2003. Natural Hierarchical Refinement for Finite Element Methods. *International Journal on Numerical Methods in Engineering 56*, 8, 1109–1124.

[Lamport 1993] LAMPORT, L. 1993. How to Write a Proof. http: //research.microsoft.com /users /lamport /proofs /proofs.html.

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

[Layton and van de Panne 2002] LAYTON, A. T., AND VAN DE PANNE, M. 2002. A Numerically Efficient and Stable Algorithm for Animating Water Waves. *The Visual Computer 18*, 1, 41–53.

[Lee et al. 1997] LEE, S., WOLBERG, G., AND SHIN, S. Y. 1997. Scattered Data Interpolation with Multilevel B-Splines. *IEEE Transactions on Visualization and Computer Graphics 3*, 3, 228–244.

[Lew et al. 2003a] LEW, A., MARSDEN, J., ORTIZ, M., AND WEST, M. 2003. Asynchronous variational integrators. *Archive for Rational Mechanics and Analysis 167*, 2, 85–146.

[Lew et al. 2003b] LEW, A., MARSDEN, J., ORTIZ, M., AND WEST, M. 2003. Variational time integrators. *International Journal for Numerical Methods in Engineering*. (to appear).

[Lewis et al. 1996] LEWIS, R. W., MORGAN, K., THOMAS, H. R., AND SEETHARAMU, K. N. 1996. *The Finite Element Method in Heat Transfer Analysis*. Wiley Europe.

[Liu and Joe 1995] LIU, A., AND JOE, B. 1995. Quality Local Refinement of Tetrahedral Meshes based on Bisection. *SIAM Journal on Scientific Computing 16*, 6, 1269–1291.

[Liu and Joe 1996] LIU, A., AND JOE, B. 1996. Quality Local Refinement of Tetrahedral Meshes based on 8-subtetrahedron Subdivision. *Mathematics of Computation 65*, 215, 1183–1200.

[Liu et al. 1994] LIU, Z., GORTLER, S. J., AND COHEN, M. F. 1994. Hierarchical Spacetime Control. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, 35–42.

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

[Lounsbery et al. 1997] LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. 1997. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *ACM Transactions on Graphics 16*, 1, 34–73.

[Malvern 1969] MALVERN, L. E. 1969. *Introduction to the Mechanics of a Continuous Medium*. Prentice-Hall, Englewood Cliffs, NJ.

[Mandal et al. 1997] MANDAL, C., QIN, H., AND VEMURI, B. C. 1997. Dynamic Smooth Subdivision Surfaces for Data Visualization. In *IEEE Visualization '97*, 371–378.

[Mandal et al. 2000] MANDAL, C., QIN, H., AND VEMURI, B. C. 2000. A novel FEM-based dynamic framework for subdivision surfaces. *Computer-Aided Design 32*, 8-9, 479–497.

[Marsden and Glavaški 2002] MARSDEN, S. L. J. E., AND GLAVAŠKI, S. 2002. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *Int. J. Robust Nonlinear Control*, 12, 519–535.

[Méhauté et al. 1997] MÉHAUTÉ, A. L., RABUT, C., AND SCHUMAKER, L. L., Eds. 1997. *Surface Fitting and Multiresolution Methods*, vol. 2. Vanderbilt University Press.

[Metaxas and Terzopoulos 1992] METAXAS, D., AND TERZOPOULOS, D. 1992. Dynamic deformation of solid primitives with constraints. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2, 309–312.

[Meyer et al. 2003] MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2003. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Visualization and Mathematics III*, H.-C. Hege and K. Polthier, Eds. Springer-Verlag, Heidelberg, 35–57.

[Mickens 2000] MICKENS, R. E., Ed. 2000. *Applications of Nonstandard Finite Difference Schemes*. World Scientific Pub. Co.

[Moore 1981] MOORE, B. C. 1981. Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1), 17–32.

[Müller et al. 2002] MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable Real-Time Deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, 49–54.

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

[O'Brien et al. 2001] O'BRIEN, J. F., COOK, P. R., AND ESSL, G. 2001. Synthesizing Sounds From Physically Based Motion. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 529–536.

[O'Brien et al. 2002] O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical Modeling and Animation of Ductile Fracture. *ACM Transactions on Graphics 21*, 3, 291–294.

[Perrier and Basdevant 1991] PERRIER, V., AND BASDEVANT, C. 1991. Travelling Wavelets Method. In *Proceedings of the US-French Wavelets and Turbulence Workshop*, Princeton.

[Plaza and Carey 2000] PLAZA, A., AND CAREY, G. 2000. Local Refinement of Simplicial Grids based on the Skeleton. *Applied Numerical Mathematics 32*, 2, 195–218.

[Prenter 1975] PRENTER, P. M. 1975. *Splines and Variational Methods*. John Wiley & Sons.

[Press et al. 1993] PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. 1993. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK.

[Qian and Dutta 2003] QIAN, X., AND DUTTA, D. 2003. Design of heterogeneous turbine blade. *Computer-Aided Design 35*, 3, 319–329.

[Ram-Mohan 2002] RAM-MOHAN, R. 2002. *Finite Element and Boundary Element Applications in Quantum Mechanics*. Oxford University Press.

[Rannacher and Turek 1992] Rannacher, R., and Turek, S. 1992. Simple nonconforming quadrilateral stokes element. *Num. Meth. PDE*, 8(2), 97–111.

[Raviart and Thomas 1977] Raviart, P., and Thomas, J.-M. 1977. A mixed finite element method for second order elliptic problems. In *Mathematical Aspects of the Finite Element Method*, I. Galligani and E. Magenes, Eds. Springer, 292–315.

[Richter-Dyn 1971] Richter-Dyn, N. 1971. Properties of Minimal Integration Rules. II. *SIAM Journal of Numerical Analysis 8*, 3, 497–508.

[Rivara and Inostroza 1997] Rivara, M., and Inostroza, P. 1997. Using Longest-side Bisection Techniques for the Automatic Refinement of Delaunay Triangulations. *International Journal for Numerical Methods in Engineering 40*, 4, 581–597.

[Rivara and Iribarren 1996] Rivara, M., and Iribarren, G. 1996. The 4-Triangles Longest-side Partition of Triangles and Linear Refinement Algorithms. *Mathematics of Computation 65*, 216, 1485–1502.

[Roth et al. 1998] Roth, S. H. M., Gross, M. H., Turello, S., and Carls, F. R. 1998. A Bernstein-Bézier Based Approach to Soft Tissue Simulation. *Computer Graphics Forum 17*, 3, 285–294.

[Snedeker et al. 2002] Snedeker, J. G., Bajka, M., Hug, J. M., Székely, G., and Niederer, P. 2002. The creation of a high-fidelity finite element model of the kidney for use in trauma research. *The Journal of Visualization and Computer Animation 13*, 1, 53–64.

[Stam 1998] Stam, J. 1998. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. *Proceedings of SIGGRAPH 98*, 395–404.

[Stam 1999] Stam, J. 1999. Stable Fluids. In *Proceedings of SIGGRAPH 99*, 121–128.

[Stam 2001] Stam, J. 2001. On Subdivision Schemes Generalizing Uniform B-Spline Surfaces of Arbitrary Degree. *Comput. Aided Geom. Des. 18*, 5, 383–396.

[Strang and Fix 1973] Strang, G., and Fix, G. 1973. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press.

[Strang and Nguyen 1996a] Strang, G., and Nguyen, T. 1996. *Wavelets and Filter Banks*. Wellesley-Cambridge Press.

[Strang and Nguyen 1996b] Strang, G., and Nguyen, T. 1996. *Wavelets and Filter Banks*. Wellesley-Cambridge Press.

[Strang 1989] Strang, G. 1989. Wavelets and dilation equations: A brief introduction. *SIAM Review 31*, 4, 614–627.

[Strela et al. 1999] STRELA, V., HELLER, P. N., STRANG, G., TOPIWALA, P., AND HEIL, C. 1999. The Application of Multiwavelet Filterbanks to Image Processing. *IEEE Transactions on Image Processing 8*, 4, 548–563.

[Szirmay-Kalos et al. 2001] SZIRMAY-KALOS, L., CSONKA, F., AND ANTAL, G. 2001. Global Illumination as a Combination of Continuous Random Walk and Finite-Element Based Iteration. *Computer Graphics Forum 20*, 3, 288–298.

[Terzopoulos and Fleischer 1988] TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture. *Computer Graphics (Proceedings of SIGGRAPH 88) 22*, 4, 269–278.

[Terzopoulos and Qin 1994] TERZOPOULOS, D., AND QIN, H. 1994. Dynamic NURBS with Geometric Constraints for Interactive Sculpting. *ACM Transactions on Graphics 13*, 2, 103–136.

[Terzopoulos et al. 1987a] TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically Deformable Models. In *Proceedings of SIGGRAPH*, 205–214.

[Terzopoulos et al. 1987b] TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically Deformable Models. *Computer Graphics (Proceedings of SIGGRAPH 87) 21*, 4, 205–214.

[Thompson et al. 1999] THOMPSON, J. F., SONI, B. K., AND WEATHERILL, N. P., Eds. 1999. *Handbook of Grid Generation*. CRC Press.

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

[Troutman and Max 1993] TROUTMAN, R., AND MAX, N. L. 1993. Radiosity Algorithms Using Higher Order Finite Element Methods. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, 209–212.

[Velho and Zorin 2001] VELHO, L., AND ZORIN, D. 2001. 4-8 Subdivision. *Computer-Aided Geometric Design 18*, 5, 397–427. Special Issue on Subdivision Techniques.

[Versteeg and Malalasekera 1996] VERSTEEG, H. K., AND MALALASEKERA, W. 1996. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Addison-Wesley Pub Co.

[Warfield et al. 2000] 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 and Witkin 1992] WELCH, W., AND WITKIN, A. 1992. Variational Surface Modeling. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2, 157–166.

[Wille 1992] WILLE, S. 1992. 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.

[Wille 1996] WILLE, S. 1996. 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.

[Witkin and Kass 1988] WITKIN, A., AND KASS, M. 1988. Spacetime Constraints. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, vol. 22(4), 159–168.

[Wu et al. 2001] 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 1986] YSERENTANT, H. 1986. On the Multilevel Splitting of Finite-Element Spaces. *Numerische Mathematik 49*, 4, 379–412.

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

[Zorin and Kristjansson 2002] ZORIN, D., AND KRISTJANSSON, D. 2002. Evaluation of Piecewise Smooth Subdivision Surfaces. *Visual Computer 18*, 5/6, 299–315.

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

[Zorin and Schröder 2001] ZORIN, D., AND SCHRÖDER, P. 2001. A Unified Framework for Primal/Dual Quadrilateral Subdivision Schemes. *Comput. Aided Geom. Des. 18*, 5, 429–454.

[Zorin et al. 1996] ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1996. Interpolating Subdivision for Meshes with Arbitrary Topology. *Proceedings of SIGGRAPH 96*, 189–192.

[Zorin 2000] ZORIN, D. 2000. Smoothness of Subdivision on Irregular Meshes. *Constructive Approximation 16*, 3, 359–397.