UNIVERSITY OF CALIFORNIA, SAN DIEGO

# An Adaptive Mesh Refinement Technique for Dynamics of Solids

A Dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Structural Engineering

by

Abhishek Trivedi

Committee in Charge:

Professor Petr Krysl, Chair
Professor David Benson
Professor Michael Holst
Professor Fransisco Lanza Di Scalea
Professor P. Benson Shiang

2007

**Dedicated to my belated uncle Mr. Nishkam Tripathi**

**and grand mother Mrs. Shri Devi Tripathi.**

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# PREFACE

This work is about adaptive mesh refinement and how it can be successfully utilized in dynamics problems. Adaptive mesh refinement technique focuses on refinement of the basis functions instead of subdivision of elements. Refinement done in this fashion produces compatible and hierarchical meshes [66], the overall algorithm is easy to implement [32], and it can be applied to a wide range of 1D, 2D, and 3D elements. We apply this technique to numerically explicit, dynamic problems and verify our solution strategy by validating it against NDE experiments performed for simulated guided wave problems [35].

There are already concrete, compelling applications of the current work [76]. A variety of complex simulation problems are likely to benefit from our adaptive framework [77-79].

# ACKNOWLEDGEMENTS

I would like to gratefully acknowledge funding from the Office of Naval Research for a part of this project.

I am very grateful for the guidance of Dr. Petr Krysl. Thanks for having faith on my ability to carry on this research work at some of the most difficult times. Without your mentoring and close collaboration this research work could not have been a reality. The lessons I learned from you will stay with me for lifetime.

I am very thankful to Dr. John McNamara for his support in development of verification problem with simulation of long range guided waves in rail road tracks. John thanks for being a great friend and fellow researcher.

I am thankful to Professor Francisco Lanza di Scalea for his encouragement and support on guided wave simulation project.

I am grateful that I had a chance to go through series of math courses taught by Dr. Michael Holst and series of FEA courses taught by Dr. David Benson. Lessons that I learned during these courses provided a solid foundation to carry on this research work.

I am thankful to Aaron Young for proof reading part of this work. Thanks for all that midnight oil. You have been a great friend.

I am thankful to Dr. Giridhar Chikkapalli and Dr. Amitav Majumdar at San Diego super computer center for there collaboration on brain deformation project. You guys had been a great help.

I am thankful to faculty and staff at structural engineering department. It was a pleasure and an honor to have a chance to work with so many highly regarded people in the field.

Fred, Bao Zhu, Bill, Daniel, Paul, Dawn, Sara, Sebastian, Howard:  Thanks for making my stay at UCSD so much fun.

# VITA

**Education**

- Ph.D. , University of California, San Diego, Adaptive mesh refinement for Dynamics of solids, Structural Engineering, University of California, San Diego
- M.S. , Structural Engineering, University of California, San Diego

**Employment History**

- Sept. 1999 – Sept. 2001 , Transoft International SA, Paris, France
- Feb 2005 – Present, Solidworks Corp., Santa Monica, CA, USA

**Publications**

- Krysl, P., Trivedi, A., and Zhu, B., 2004, “Object-oriented hierarchical refinement with CHARMS,” Int. J. Numer. Meth. Engng., **60**(8), pp. 1401-1424.

- Trivedi, A., Krysl, P., McNamara, J., and Lanza di Scalea, F., 2004, “Adaptive simulation of ultrasonic guided waves in railroad tracks using CHARMS,” Proc. 6th World Conference on Computational Mechanics , Springer - Verlag.

- Majumdar, A., Birnbaum, A., Choi, D. J., Trivedi, A., Warfield, S. K., Balridge, K., Krysl, P., 2005, “A Dynamic Data Driven Grid System for Intra-operative Image Guided Neurosurgery,” ICCS 2005, LNCS 3515 pp 672-679, Springer-Verlag

- Krysl, P., Damaser, M., Chukkapalli, G., Majumdar, A., Choi, D. J., Trivedi, A., Warfield, S. K., Hoyte L., 2006, “Computational model of levator ani muscle stretch during natural birth,” Int. J. of Biomechanics, To Appear

- Krysl, P., Trivedi, A., 2005, “Instructional Use of Matlab Software Components for Computational Structural Engineering Applications”, International Journal for Engineering Education, Vol 21 (5), 778 – 783,

# ABSTRACT OF THE DISSERTATION

An Adaptive Mesh Refinement Technique for Dynamics of Solids

by

Abhishek Trivedi

Doctor of Philosophy in Structural Engineering

University of California, San Diego, 2007

Professor Petr Krysl, Chair

Simulation of dynamics problems generally is heavily dependent on mesh size for convergence and accuracy. In many cases, the requirement of mesh size reaches to such proportions that the problem becomes unsolvable for the available computational resources. We perceive that such problems can be solved by refining and unrefining mesh adaptively during time stepping. Adaptive mesh refinement techniques available in popular literature are marred by various limitations, such as element-specific techniques, algorithm-specific techniques, etc. We utilize Conforming Hierarchical

Refinement Methods (CHARMS) [32] for our purpose since it relies on the refinement of finite element basis instead of geometric sub-division of elements. Refinement in this fashion produces conforming meshes during adaptive mesh refinement for a wide range of elements.  We improve CHARMS to incorporate time stepping problems. In order to show the effectiveness of such an algorithm, we modify an explicit Newmark solver to incorporate it within the adaptive mesh refinement framework. We perform the analysis of conservation properties of the modified algorithm to understand its limitations. Later, we demonstrate application of such development with a very practical example, simulation of an NDE experiment using ultrasonic guided waves. Details presented in [76] has shown that such a simulation would otherwise not have been possible. We further study the guided wave experiment and optimize various parameters of the algorithm to validate experimental results.

# OUTLINE AND CONTRIBUTIONS

## Chapter 2

This chapter establishes the foundation for the refinement of basis functions as suggested in [36] for discrete computational mechanics applications. Research such as this, where alteration in conventional numerical techniques is evident, requires development of software tools and implementation of ideas to study and demonstrate its impact. Although [32] highlights many of these ideas in detail, it was necessary to include the details of the modification in software framework for dynamic activation and deactivation during adaptive steps in the direct time integration scheme not published in [32]. Other major contributions in theoretical understanding presented in this chapter are attributed to the development of a procedure to properly interpolate data over refined mesh by Zhu and Krysl [34] and refinement of tetrahedral mesh by Enders and Krysl [33]. Following this development, many modifications in the data structure of our software were made based on object oriented adaptive framework. For example, a pointer to a container template class called “bucket” was introduced to automatically increase and decrease in size and reshuffle the basis function set efficiently during refinement and unrefinement. We used “in house” post processing tools to visualize the results; a graphics library called Elixir was utilized to create snapshots during time progression of the adaptive solver. Watch points were introduced to visualize XY plots during time marching. We also experimented and modified the math library PETSC for linear equation solvers and pre-conditioners, since dynamic problems can often be computationally very expensive. All the applications shown in this chapter were modified and tested using this new framework.

# Chapter 3

The general hyperbolic equation of the form $u,_{tt} = c^2 \Delta u$ has many applications in computational mechanics. For example, if we can assume a time harmonic motion, it leads to parabolic Helmholtz's equation, which is used for certain types of wave propagation problems. In discrete mechanics its weak form (or Galerkin formulation) leads to initial value problem for elastodynamics applications including, vibration and wave propagation problems. There are various ways to solve this problem in a time or frequency domain. Its solution in a frequency domain often relies on normalization of the mass and stiffness matrix with respect to natural modes. Participation of more natural frequencies corresponding to higher modal mass leads to a more accurate result. In a time domain it is solved using various direct time integration schemes. The Newmark family of solvers are industry wide and are the most established and most popular techniques to solve this problem. The Newmark family of solvers includes several implicit schemes and one explicit scheme, the central difference method. Implicit schemes such as the average acceleration method and the Fox-Goodwin method are computationally very expensive, unconditionally stable, and designed to conserve mass, momentum, and energy. On the other hand, the central difference method is relatively cheap computationally, conditionally stable, and designed to conserve only momentum. The explicit method imposes various restrictions on the solution process, time step, mesh size, etc., and its construction in certain fashions very often is reflected in the conservation properties. However its general economy for a wide variety of elastodynamics applications without the sacrifice of accuracy has made it ubiquitous in the commercial software, and much research effort has been spent to analyze its general behavior. In our case, we are utilizing it to demonstrate use of our adaptive scheme in dynamic solvers. Past work done on analysis of such schemes [81,88] relies on the fact that discrete space essentially does not change during the time marching algorithm. However, in our case discrete space is altered by our adaptive scheme. Therefore, various established assumptions and conservation properties must be reevaluated to account for such changes. We study the conservation properties of such a scheme, establish its limitations, and derive conclusions for some of the targeted applications, such as the one presented in chapter 5. An investigation of properties of integration, using such a scheme over refined mesh, brings our attention to several related issues such as: data transfer from coarse mesh to finer mesh and the necessity of imposing a partition of unity. Implementation issues related to such a solver are discussed to support the adaptive data structure discussed in chapter 4. An example from such implementation is discussed at the end of the chapter.

**Chapter 4**

A guided wave propagates in the axial direction of a layer, while behaving as a standing wave through the thickness of the layer. Long range ultrasonic guided waves are often preferred for their use in inspection of cracks in long slender geometries, such as railroad tracks. This is because conventional NDE techniques such as the pulse-echo method suffer from the problem of the complicated nature of reflected wave forms, making it very difficult to produce real time analysis. Apart from that, inspection of engineering structures using long-range guided waves is attractive, because it is possible to investigate complete material volumes in regions over 100 meters away from the point of measurement. Effective inspections require a good understanding of the different vibration modes. Finite element modeling is a powerful tool in assessing the feasibility of guided wave inspections.

Several conclusions are derived in chapter 3, and all point towards a robust application of the CHARMS-based adaptive dynamics solver for simulation of guided wave problems for many reasons. For one, solution of wave propagation problems leads to very fine mesh using a direct time integration scheme if we are looking for response of intermediate frequencies, as mesh size is related to wavelength and frequency. Very often this is the case in NDE applications. The problem becomes especially complex in practical NDE applications where predominant frequencies, which can provide useful information, are not know a priori. Analysis in this case cannot be performed in a frequency domain. We have all the ingredients to perform such an analysis: a lumped mass matrix which stays symmetric, positive definite with refinement, a momentum conserving, explicit adaptive Newmark solver where adaptivity does very little to kinetic energy (in small strain problems), and a mesh where refinement and unrefinement together are chasing the energy wave front. As a result, the effect of a moving and reflected wave front can be clearly seen with several-fold reduction in computational effort. Analysis of frequency content in the reflected wave front reveals startling similarity to experimental NDE results. This chapter describes this verification problem in great detail.

**Chapter 5**

Work done so far opens the door for solutions of various complicated engineering problems. Because of our immediate involvement in biomechanics [78] and NDE applications [76] during this research work, these areas have been the main beneficiaries of our research. We highlight similar developments in this chapter. This chapter also presents ideas for the potential improvement in the algorithm presented in this thesis, such as what can be done about a non-interpolating basis resulting from CHARMS, how lossy unrefinement may be handled, or what can be done to apply similar techniques in nonlinear dynamic problems. Some of these ideas are borrowed, while others are completely our own. The full range of such ideas is long, so we conclude this chapter with the most important, and hope that future research will shed more light on these topics.

# CHAPTER 1: BACKGROUND AND MOTIVATION

Necessity is the mother of all invention. The need for computational resources is as old as the invention of the computer itself. Numerical simulations of practical engineering problems especially consume a lot of computational effort, even with a small amount of complexity. Therefore, it is safe to assume that the idea of automatic increase or decrease in a computational grid to achieve better quality results using adaptive mesh refinement is as old as the notion of finite elements. Although a large volume of literature is available on this topic, adaptive solvers for general finite element problems have not been employed broadly because of the huge implementation effort. This chapter highlights current understanding of improvement of a solution over a discrete domain using adaptive mesh refinement. We present a brief review of existing techniques and argue in favor of a technique based on refinement of shape functions [31,36].

## 1.1 Introduction

In order to numerically simulate a physical problem represented by partial differential equations, the domain of interest must be discretized into a set of elements. A numerical simulation approximates the differential equations with algebraic equations, which in turn are solved to give an approximate solution for the field. In order to be a valid approximation, the discretization must conform to all boundary conditions of the original problem. Therefore, the mesh generation step is extremely important, since it dictates the convergence of the computational schemes as well as the accuracy of the numerical approximations of the solution.

There are many algorithms for creating meshes, ranging from geometry specific to general context techniques. Methods to create these meshes include user-driven, semiautomatic, and fully automatic methods which can create structured, unstructured, and mixed meshes. There is software [1] as well as a large group of literature [2,3] that deals with structured meshing ("grid generation"). A structured mesh is defined by all interior nodes of the mesh having equal numbers of adjacent elements. Commonly, the structured meshes generated are predominantly quadrilateral or hexahedral in nature. The meshing algorithms to generate structured meshes usually utilize complex iterative smoothing techniques in order to align elements with the boundary conditions of the problem. "Block-structured" techniques can be utilized to break up the domain into topological blocks where non-trivial boundary conditions exist.

Unstructured meshes differ from structured meshes in that the numbers of adjacent elements need not be equal. Triangle and tetrahedral meshing are the most commonly used forms of unstructured meshing; however, quadrilateral and hexahedral meshing may also be unstructured. As far as generation of the meshes, there is a great deal of overlap with regard to the technologies involved. The main difference is that structured mesh generators utilize iterative smoothing algorithms in the creation of the mesh.

Broadly, elements can be classified as structural elements and continuum elements. Whereas in the formulation of continuum elements the displacements u, v, and w are interpolated in terms of nodal point displacements of the same kind, in the formulation of structural elements, the displacements u, v, and w are interpolated in terms of mid-surface displacements and rotations. In addition to that, a major assumption is made that the stress normal to mid-surface is zero in formulation of structural elements. Examples of structural elements are beam, plate, and shell elements, while examples of continuum elements are triangle and tetrahedra elements.

Depending on the requirement of the problem, a combination of different types of elements can be used to solve a finite element problem to provide maximum accuracy with the least amount of computational effort. For example, mass elements can be used in problems such as frequency analysis of a PCB (Printed Circuit Board), with heavy components to simulate the mass of a component when the structural response of that component is of no interest. Spring elements can be used to simulate a flexible support condition of a part in shock isolators and flexible mounts. Gap elements can be used to simulate contact/separation between two disjointed bodies when closed, normal, and shear forces are transmitted. Truss and beam elements can be used to simulate space frame structures where the joint connections are strong. While beam elements can resist axial tension, compression, bending, and torsion, truss elements resist axial tension and compression only. For 2-dimensional analyses, 2D-planar and triangular elements can be used when all loads and displacements are in the analysis plane. Shell elements can be used to model thin members where the geometry or forces/displacements are not planar and the structure is so thin that meshing with solid elements leads to either poor aspect ratios or an impractical number of elements. Three dimensional elements such as tetrahedral and hexahedral elements are used for full-blown three dimensional analyses. While practically any solid model can be *automatically* meshed with tetrahedral elements, many models involving bricks must be meshed manually along with the geometry.

As the number of elements is increased, a valid finite element approximation must converge to the analytical solution of the differential equation. Usually, this convergence should show all the properties of the analytical scheme being approximated. This is true since the differential equations modeling the domain of interest show all of the necessary conditions that the solution must satisfy (e.g. stress, strain, displacement, etc.). While approximating the analytical solution, the convergence of the finite approximation must adhere to all physical constraints known to exist for the problem of interest. Depending on the finite elements used to analyze the model, the approximation may converge monotonically or non-monotonically to the exact solution as the number of elements is increased.

To converge monotonically, elements must be complete and compatible. Completeness is satisfied when the element's displacement functions are able to represent the rigid body displacement and constant strain states. Compatibility requires that displacement within and between the elements is continuous. In order to understand convergence, understanding of the following properties of finite elements is essential [4]:

1. If the exact solution of the governing partial differential equation is represented by u and the corresponding finite element solution using mesh by $u_h$ , the error $e_h$ has the following property:

$$e_h = -u_h + u$$

$$a(e_h , v_h) = 0 \qquad \text{for every } v_h \in V_h$$

Here a( . , .) is a symmetric bilinear form and $V_h$ is the space of finite element displacement functions which correspond to the displacement interpolations contained in all element displacement interpolation matrices. As the space $V_h$ increases, the solution accuracy will increase since large spaces consist of smaller spaces.

2. $a(u_h , u_h ) \leq a(u , u)$

   This indicates that the strain energy corresponding to finite elements is always less than or equal to the strain energy corresponding to the exact solution.

3. $a(e_h , e_h ) \leq a(u - v_h, u - v_h)$ for every $v_h \in V_h$

   This shows that the finite element solution $u_h$ is chosen such that the strain energy due to the difference in the actual and finite element solutions is the minimum.

The above properties provide clear evidence of the convergence of the solution as the mesh is refined. Furthermore, from properties 2 and 3 above, it is clear that the strain energy corresponding to the finite element will approach the exact strain energy as the mesh is refined.

The rate of convergence depends on the order of the polynomial used to construct the shape function. The following expression relates the error due to discretization to the order ($k$) of the approximation polynomial:

$$\|e_h\| = \|u - u_h\| \leq ch^k \tag{1-1}$$

Here constant $c$ is independent of mesh size $h$ but depends on material properties. In some cases when constant $c$ becomes very large, mesh size $h$ should be appropriately small to achieve convergence. Using the above relation, it can be shown that displacements converge one order higher than stresses. In the above convergence relationship, it is assumed that the solution of the partial differential equation is smooth. However, this is not the case in most of the practical problems. Sudden changes in geometry, material properties, loads, and boundary conditions produce a solution which is not smooth over the entire domain. Hence mesh needs to be fine in the areas where stresses or displacements are unusually high. The equation of convergence therefore should account for lack in smoothness and can be rewritten as:

$$\left\| u - u_h \right\|_1^2 \le c \sum_m h_m^{2k} \left\| u \right\|_{k+1,m}^2 \qquad (1\text{-}2)$$

Here m denotes the individual element, $h_m$ is the corresponding mesh size, and k is the degree to which the polynomial used in the displacement assumption is complete. In this sequence mesh size is uniformly reduced in the required areas. This approach is referred as the “H method of analysis”. Alternatively, solutions can be improved by increasing the order of approximation k. Such techniques are called P methods.

## 1.2 Mesh Improvement and Adaptive Mesh Refinement

In most practical finite element problems, the initial mesh does not provide an optimum solution of the governing differential equations. It needs to be improved to achieve a solution with a certain level of accuracy. After creation of the initial mesh, techniques such as smoothening or coarsening of mesh are employed to get elements with the correct size and aspect ratio. In the case of a static problem (after initial analysis is complete), the difference between the exact solution and the discrete solution provides a basis for mesh improvement. Typically remeshing and/or adaptive mesh refinement are used to improve the solution in consecutive iterations. Remeshing is typically done manually by regenerating the mesh with mesh controls and/or increasing the order of approximation in the areas where the solution is not sufficiently smooth or does not have the minimum required accuracy. The method of mesh improvement can be chosen based on either the experience of the analyst or using an error indicator algorithm.

Instead of using remeshing adaptive mesh refinement, an automatic mesh improvement process may be employed. The mesh improvement process is typically guided by an error indicator. Adaptive techniques can be divided into the following groups:

1. Specialized Adaptive Techniques
2. H Type Techniques
3. P Type Techniques
4. HP Type Techniques

### 1.2.1 Specialized Adaptive Techniques

The volume of literature published for adaptive techniques for specific types of problems is enormous. This type of literature can be broadly categorized as problem-specific, element-type, order-specific, or a combination of any or all of the three. During the first developmental stages, most adaptive techniques were developed for specific non-transient problems for specific types of elements. Works done by researchers such as Shephard, Babuška, and Berger [11,27,42,54,57,59] represent a chronology of developments in adaptive mesh refinement techniques. Probably the most interesting works in this category appeared in the early 1990s and later in the areas of highly nonlinear and dynamic problems. Here are the highlights of some of the papers.

To improve step size dependent approximations, Richardson's extrapolation is quite often utilized. With this, better approximation can be achieved for a polynomial $A$, where $A(h)$ depends on a positive step size $h$ by subtracting the largest term in the error, $O(h^{k_o})$. For better appoximations, this algorithm can be done repeatedly to remove further error terms.

One of the more widely used adaptive mesh refinement algorithms using Richardson's extrapolation was developed by Berger [11]. It starts with the entire computational domain overlayed with a coarsely resolved base-level regular cartesian grid. As the calculation progresses, individual cells within the grid are tagged for refinement using a criterion that can either be user-supplied or based on the Richardson extrapolation. For example, constant mass per cell for a criterion can be used since higher density regions are more highly resolved.

All tagged cells then proceed through refinement. This means that a finer grid is overlayed on top of the coarser grid. After refinement, individual grid patches on a single fixed level of refinement are integrated, advancing those cells in time. Finally, a correction procedure is implemented to fix transfers along coarse-fine grid interfaces. This is done to ensure that the amount of any conserved quantity transferred between bordering cells is exactly the same. At some point, if the level of refinement in a cell is greater than required, then the high resolution grid can be replaced with a coarser one.

Another interesting series of papers were published by Ortiz's group at the California Institute of Technology. One of the most broad-based efforts to create a general scheme for highly nonlinear and possibly dynamic problems was a paper published by Radovitzky and Ortiz in 1999 [12]. The idea utilized the fact that most nonlinear problems (nonlinear elastic, elasto-plastic, visco-plastic solids, compressible Newtonian fluids, etc.) follow a minimization principle which may be used as a basis for mesh refinement. For example, a stress update having the requisite potential structure can be procured by integrating the constitutive relationship along the path of minimization, i.e. along such deformation histories which minimize the incremental work of deformation. This minimization principle may be utiliized as a criterion for estimating error. Optimal distribution of mesh size then follows from an error indicator. One of the obstacles for mesh adoption in time-dependent problems is that mesh at time t(n+1) results from error indication at time t(n). However mesh fields differ at t(n+1) and t(n). This was solved by taking advantage of a piecewise constant representation for the state variables and the transfer operator that resulted. Whole collections of state variables are transferred as blocks from one quadrature point to another, which maintained all internal constraints and compatibility between the various state variables.  Another paper published by Molinari and Ortiz [13] made use of a similar technique for dynamic plasticity by creating an asymptotic error bound and mesh refinement by edge collapse.

Papers by Selman, Hinton, Bićanić [14], and others focused on developing reliable adaptive mesh refinement technique via remeshing. Adaptive mesh refinement use in plate bending problems with boundary layers, strain softening problems, and two-dimensional dynamic problems has been shown [14], and it benchmarks against various well-known applications in engineering.

### 1.2.2 H Type Techniques

The H type techniques can be broadly subdivided into four categories:

1. Techniques Based on Remeshing [8-10,14]
2. Fixed Order Mesh Modification Procedures [24-30]
3. Techniques Based on Element Splitting and Edge Collapse [16-20]
4. Techniques Based on Basis Refinement [15,31-35]

#### 1.2.2.1 Techniques Based on Remeshing

Adaptive remeshing methods regenerate the entire mesh using an automatic mesh generation algorithm, which uses element shape and size information for constraints. The INRIA group [38-41] applied adaptive remeshing in terms of adaptive Delaunay Kernal, and use of adaptive remeshing based on anisotropic advancing front mesh generation was done both by Peraire et al. [10] and Möller et al. [37]. Such advantages of adaptive remeshing include the ability of curved domains to be considered by mesh generation algorithms, and coarsening of mesh is taken into account. However, these methods prove very inefficient for the refining of only a few elements on an already refined mesh. Also, it tends to introduce additional complexities when solution fields are transferred between meshes.

#### 1.2.2.2 Fixed Order Mesh Modification Procedures

In this type of adaptive mesh refinement, the local mesh is modified in a certain order. For example, de Cougny and Shephard [27], de l'Isle and George [28], and Joe [29] have all improved the existing mesh quality using four different mesh modification procedures: swap, collapse, split, and relocation. For adaptivity in both two dimensions [24-26] and three dimensions [30], applications were made of these local mesh modification operations using desired mesh size and shape distribution as governing factors. Advantages of this type of method include: it is a local process for which curved geometry domain may be accounted [42], coarsening of collapse-based mesh may be applied [27], and as each modification is applied, solution field transfer can be done incrementally. Despite these benefits, the effectiveness of the local mesh modification procedure in three dimensions (as far as efficiency and quality are concerned) is largely related to how the various mesh modifications are taken into account.

#### 1.2.2.3 Techniques Based on Element Splitting and Edge Collapse

In these algorithms, the driving concept is that geometric division of finite elements is mandatory for refinement. However, the result of local refinement based on element splitting shows that this usually does not ensure that the modified (refined) mesh is compatible globally. Several methods currently used to resolve this issue include: (i) unknowns of incompatibly placed nodes are constrained with respect to other nodes to ensure that the resulting approximation is compatible, even though the mesh is not; (ii) penalty methods or Lagrangian multipliers to treat incompatibility; (iii) additional elements are split until the mesh is globally compatible. Several specialized mesh refinement techniques have been devised for an array of practically important cases: for triangular and quadrilateral meshes in two dimensions [36,42-44] and tetrahedral [15,20,21,45-47] or hexahedral meshes in three dimensions [48].

#### 1.2.2.4 Techniques Based on Basis Refinement

These techniques depart from earlier mentioned element-by-element splitting methods. In place of geometrically dividing elements, these techniques rely upon division of approximation space with additional finer basis functions. Performance of this style of refinement is accomplished by adding a dilated basis function in the middle of an element in order to affect the same space as element bisection. The solution stays the same if the coefficient of the introduced function is zero. Bases constructed in this manner are precisely the classical hierarchical bases of the FE (finite element) literature [51]. This approach forms the basis of work done in this thesis and will be explained in great detail in the next chapter.

### 1.2.3 P Type Techniques

P adaptive techniques, unlike H adaptive techniques, depend upon refinement of finite element space by increasing the order of approximation on a fixed mesh. Broadly, P adaptive refinement can be classified as either isotropic or orthotropic. Orthotropic refinement is useful in situations where the solution to a boundary value problem has a very strong gradient in one direction but is relatively flat in other directions. In similar situations, refinement is performed by choosing different polynomial orders for each enrichment direction. Construction of P orthotropic spaces and their verification with the boundary layer problem of Reissner-Mindlin plates has been demonstrated [52]. The P version converges at least as fast the H version with a quasi-uniform mesh, and it converges twice as fast as the H version with a solution having $r^{\gamma}$ type singularity [54,55]. The convergence rate also depends on basis functions chosen. [56] provides insight on selection of approximation functions for various problems. Although significant progress has been made since the conception of P adaptive techniques in the early 1980s, various important issues remain, such as: 1) lack of an effective a posteriori error estimation, 2) adaptive selection of piecewise shape functions, and 3) an inverse approximation theorem derivation.

### 1.2.4 HP Adaptive Techniques

It has been observed that P refinement is more efficient in areas where solutions are smooth and H refinement is more efficient in regions near singularities. In order to take advantage of the benefits of both H and P refinements, HP adaptive schemes have been devised. Babuška and others [57,59] have shown that finite element discretizations, where element size H and locally variable polynomial orders P are optimally distributed, deliver exponential convergence rates in terms of error versus degrees-of-freedom (d.o.f.). However, designing algorithms that deliver such a sequence of optimally refined HP meshes along with exponential convergence in a fully automatic mode (with no user interaction) has been the ‘holy grail’ of HP computations for a decade and a half [60]. It has been proven that HP mesh optimization is based on minimization of the projection-based interpolation error [62]. A comparison of various a posteriori error estimators has also been presented [63]. Despite various qualities such as exponential convergence, HP adaptive schemes are still in their infancy and unpopular in commercial codes because of the enormous effort to implement even the simplest of problems.

## 1.3 Adaptive Mesh Refinement in Commercial Software

Seamless integration of discrete multi-physics simulations with a CAD system is both lucrative and profitable, since it allows mechanical designers to perform complicated numerical analyses with relatively little effort and a fairly shallow learning curve. Therefore, an increasing number of corporations involved in the production of commercial analysis software are investing in tight integration of their software with popular geometrical modeling tools. However, difficulties with respect to the proper representation of CAD geometry (such as issues related to defeaturing, tolerance, and dirty geometry) present a major bottleneck in mesh generation and the proper discrete representation of the problem. In these cases, mesh refinement provides for the ability to capture the geometry with a considerably smaller number of elements and grid points. In addition, it can help to improve mesh quality in areas where rapid geometric change is observed. Adaptive mesh refinement that is used to support proper discrete representation of geometry is known as geometry based mesh adaptivity. This functionality involves both the ability of the adapted mesh to adhere to the original geometry and mandates that access to the original geometry be present. Mesh adaptivity that does not remain true to the original geometry is severely limited by initial mesh geometric approximations. In fact, the sensitivity of the results to the local geometric shape is so high that if the mesh geometric approximation does not improve as the adaptive simulation process continues, the results would not just be a poor approximation; they would be meaningless. In many cases, mesh edges and faces are the same size as the small geometric features that are often critical to the analysis (as in the case of the accelerator cavity). Simply moving new nodes introduced during refinement to curved model surfaces can create invalid elements in these situations. Procedures needed to effectively deal with these types of problems must include both general mesh modification operations and a control algorithm that ensures the procedure is correctly proceeding [42].

Geometry based mesh adaptivity advantages include both the abilities to start with initial meshes that are coarser and to ensure that the resulting model remains within an appropriate level of accuracy for the design geometry. An additional benefit that may not be obvious is the ability to combine geometry based mesh refinement with small feature defeaturing as a function of target mesh size. This can create an adaptive geometry representation for mesh adaptivity where small features are ignored in the initial mesh, yet are accounted for as a function of target mesh size in the following stages of the mesh adaptivity process. This combined approach greatly reduces the defeaturing requirements associated with geometry access for mesh generation and allows for initial coarse meshes of detailed geometric models.

Adaptive mesh refinement as a tool to improve results of a numerical solution is still at a very basic stage of development in most of the analysis software. This is because of the lack of a universally applicable adaptive scheme. Most commercial software provides adaptive schemes with various restrictions on mesh type or solver type. H or P adaptive refinement schemes are most commonly implemented with Zienkiewicz and Zhu type error indicators for structural analysis problems. For example, in the case of Cosmos (an analysis package sold by Solidworks Corporation) it is applied to only solid elements. For solutions of fluid flow problems with finite volume methods, techniques developed by Berger [11] and others, which are based on Richardson's extrapolation, are more popular. This is true because these techniques work very well with structured meshes and parabolic systems of partial differential equations, which is very common in fluid flow analysis. In most of the black box software, details of the adaptive technique are typically not revealed to the public domain. Therefore, it is hard to provide further details about the adaptive solvers.

## 1.4 Case for a New Adaptive Technique

A good mesh refinement algorithm should be fast, robust, efficient, and applicable to a wide range of problems with minimal implementation effort. An additional benefit is if it can generate nested meshes within the refinement hierarchy, greatly simplifying the incorporation of multi-grid solvers [63,64]. The efficiency of an adaptive mesh refinement technique refers to the fact that it should not become a bottleneck of adaptive computations, and it should generate good quality geometric meshes, i.e. elements must remain well-shaped upon refinement and unrefinement. Robustness is usually expressed as the requirement of termination with a valid result in finite time.

H adaptive techniques based on element splitting, in general, do not ensure global compatibility. In order to enforce compatibility, various methods were used in the past. Such methods include enforcement of constraints, applying penalties to hanging nodes, red green blue triangulation, etc. Apart from that, they are generally formulated with one type of element in mind, and their application to different types of elements involves special tricks. The same problem has been noticed with techniques based on local remeshing [42]. Adaptive mesh refinement techniques need to be supplemented with automatic mesh generation techniques, which is a big problem for structured meshes. Despite the fact that these techniques may look simple enough, their implementation is a formidable task.

Conforming hierarchical refinement methods (CHARMS) [32] is designed to incorporate the above mentioned needs. In [33], it is clear that this technique is both much simpler and more general than current techniques. CHARMS exploits the refinement of basis functions rather than elements. [33] demonstrates that this leads to both the applicability of the algorithm to any number of spatial dimensions and a greater variety of structured and unstructured element types than any standard mesh refinement algorithm to date.

However, work done so far [32-36] limits the use of this technique to non-transient solvers. This thesis further enhances the technique by expanding its use to transient solvers and both demonstrates and verifies its application in an explicit wave propagation solver. An object-oriented code called FAMULS, developed by Petr Krysl, was enhanced to incorporate the demand of dynamic solvers.

# CHAPTER 2 : CONFORMING HIERARCHICAL ADAPTIVE REFINEMENT METHODS (CHARMS)

## 2.1 Nested Spaces

Given a set of linearly independent scalar basis functions supported on finite elements such that the finite element space V has the following property:

$$V = \left\{ v(x) : v(x) = \sum_i \phi_i(x) v_i \right\}, \tag{2-1}$$

where $x \in R^n$, $v(x) \in R^m$, $v_i \in R^m$ $(n \geq 1, m \geq 1)$ and $\phi_i(x) \in R$. It is possible to construct a refined mesh $M'$ consisting of the set of functions $\phi'_{i(x)}$ such that using the *hierarchy* of approximation spaces, $V^{(j)}$ can be obtained from the coarsest space in the hierarchy, $V^{(0),}$. Hence an infinite sequence of nested spaces can be constructed as follows:

$$V^{(0)} \subset V^{(1)} \subset V^{(2)} \subset \ldots\ldots\ldots\ldots \subset V^{(m)}; \tag{2-2}$$

## 2.2 Refinement Equation

From the above nesting relationship, it is clear that $V^j \subset V^{j+1}$ . Hence $\phi_i^{(j)}(x)$ on level *j* may be exactly resolved in basis with finer resolution $\{ \phi_i^{(j+1)}(x) \}$:

$$\phi_i^{(j)}(x) = \sum_k \beta_{ik}^{j+1} \phi_k^{j+1}(x), \tag{2-3}$$

where $\beta_{ik}^{j+1} \neq 0$ are the coefficients of the linear combination.

## 2.3 Construction of Meshes of Higher Resolution

Let us assume that the *initial* mesh consists of an arbitrary number of element types (for instance, triangles mixed together with quadrilaterals and tetrahedra in a non-manifold geometry), but is *compatible*. Furthermore, we shall consider only *nodal basis functions*, i.e. the basis functions are associated with nodes, and are composed of pieces defined over the incident finite elements. On each level $j$ of the nesting hierarchy, it is assumed that the basis functions $\phi_i^{(j)}$ verify the Kronecker delta property, i.e.,

$$\phi_i^{(j)}\left(x_k\right) = \delta_{ik} \,, \tag{2-4}$$

where $x_k$ is the location of the node associated with function $\phi_k^{(j)}$. In other words, we assume that an approximation built of basis functions from any *single* level *interpolates* the nodal values.

Refining a basis function using Equation (2-3) will mean that basis functions with higher resolution need to be constructed from pieces over “finer” finite elements. Therefore, as one of the steps in our algorithm, we shall need to geometrically divide coarse finite elements into finer ones. Our mesh hierarchy will be built up in such a way that each element is divided into elements of the same type (triangles into triangles, quadrilaterals into quadrilaterals, etc.), and elements at each level will be divided using the same pattern (for instance, one triangle into four smaller triangles, no matter which level in the hierarchy the parent triangle is).

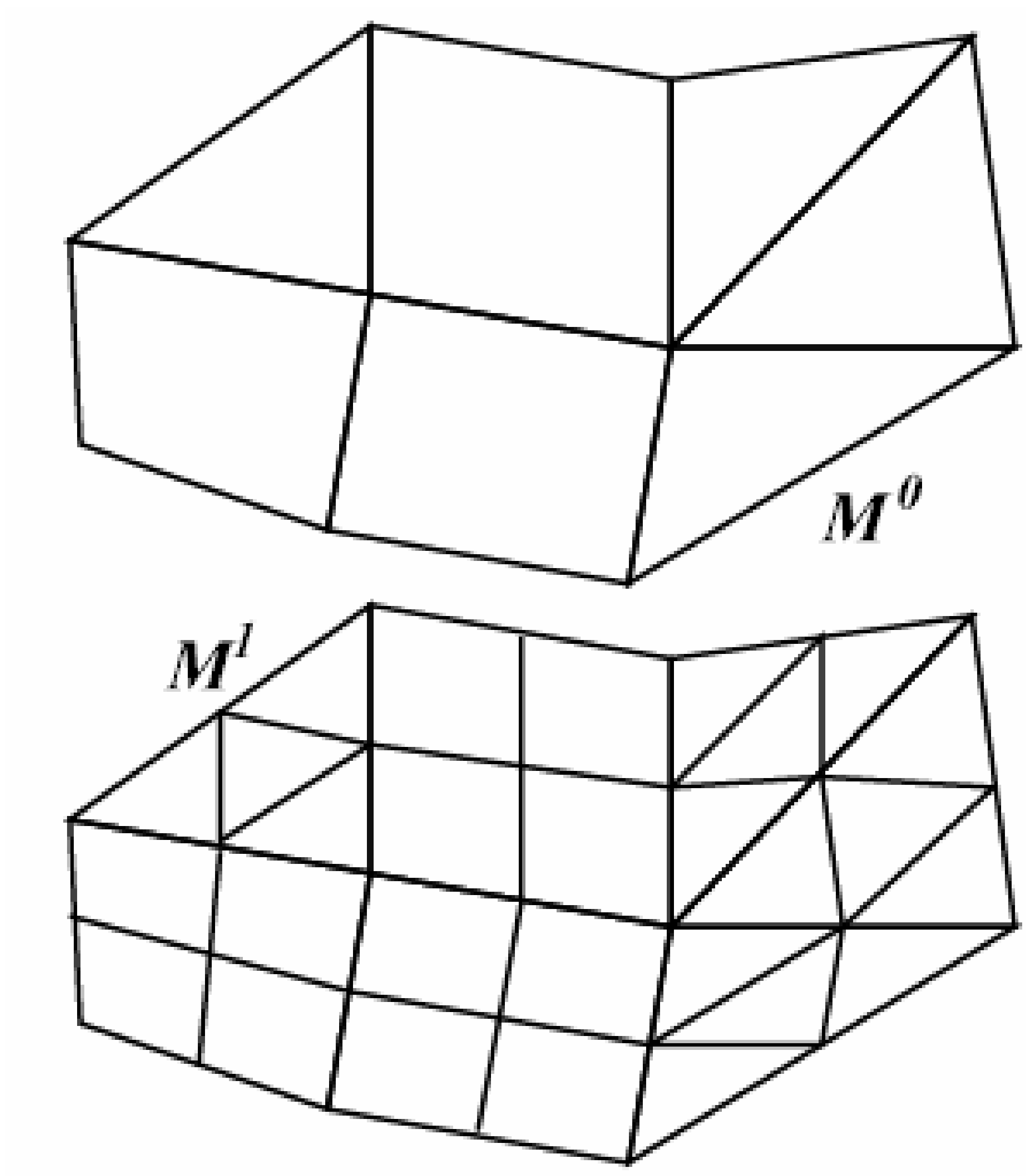


**Figure 2-1: Uniform division of mesh $M^0$ that results in a compatible mesh $M^1$.**

Furthermore, we shall assume that refining uniformly one particular mesh in the mesh hierarchy results in a compatible mesh one level higher; compare with Figure 2-1. Therefore, if for instance two finite elements in the mesh $M^j$ share an edge, we shall use a division pattern that divides the shared edge in the same way in both elements. The same principle applies to faces shared by three-dimensional elements, or in general to $d$-dimensional faces shared by ($d$ + 1)-dimensional elements. To specialize the above to some familiar element types: two-node line elements will be bisected, three-node triangles and four-node quadrilaterals will be quadrisected (which bisects their edges, making them compatible upon refinement with line elements attached along the edges of the quadrilateral or triangle), eight-node hexahedra will be octasected (again, their edges are bisected, and the divided faces are compatible with the refinement of a four-node quadrilateral).

The same principle may be applied unchanged in higher-dimensional elements of the Lagrange or Hermite type. For instance, if the basis function pieces over finite elements are constructed through the master element in the parametric space, which is then mapped to the physical space, the natural choice for refinement is uniform division in the parametric space. As an example, consider the quadratic 8-node quadrilateral element in Figure 2-2, where the isoparametric mappings are indicated by (I), and the refinement process is indicated by (R).

With this strategy there can be two possible ways to enrich the approximation space. One way is to add a dilated shape function in the middle of an element. If the coefficient of the introduced function is zero, the solution will remain unchanged. However, in this setup there may be entries in the stiffness matrix corresponding to basis functions with quite different refinement levels. We will refer to refinement in this fashion as true hierarchical refinement.

Alternatively, shape functions can be substituted with their dilated version. If coefficients of the substitution are correctly chosen, the solution again will remain unchanged. We will refer to this alternate technique as quasi-hierarchical refinement.

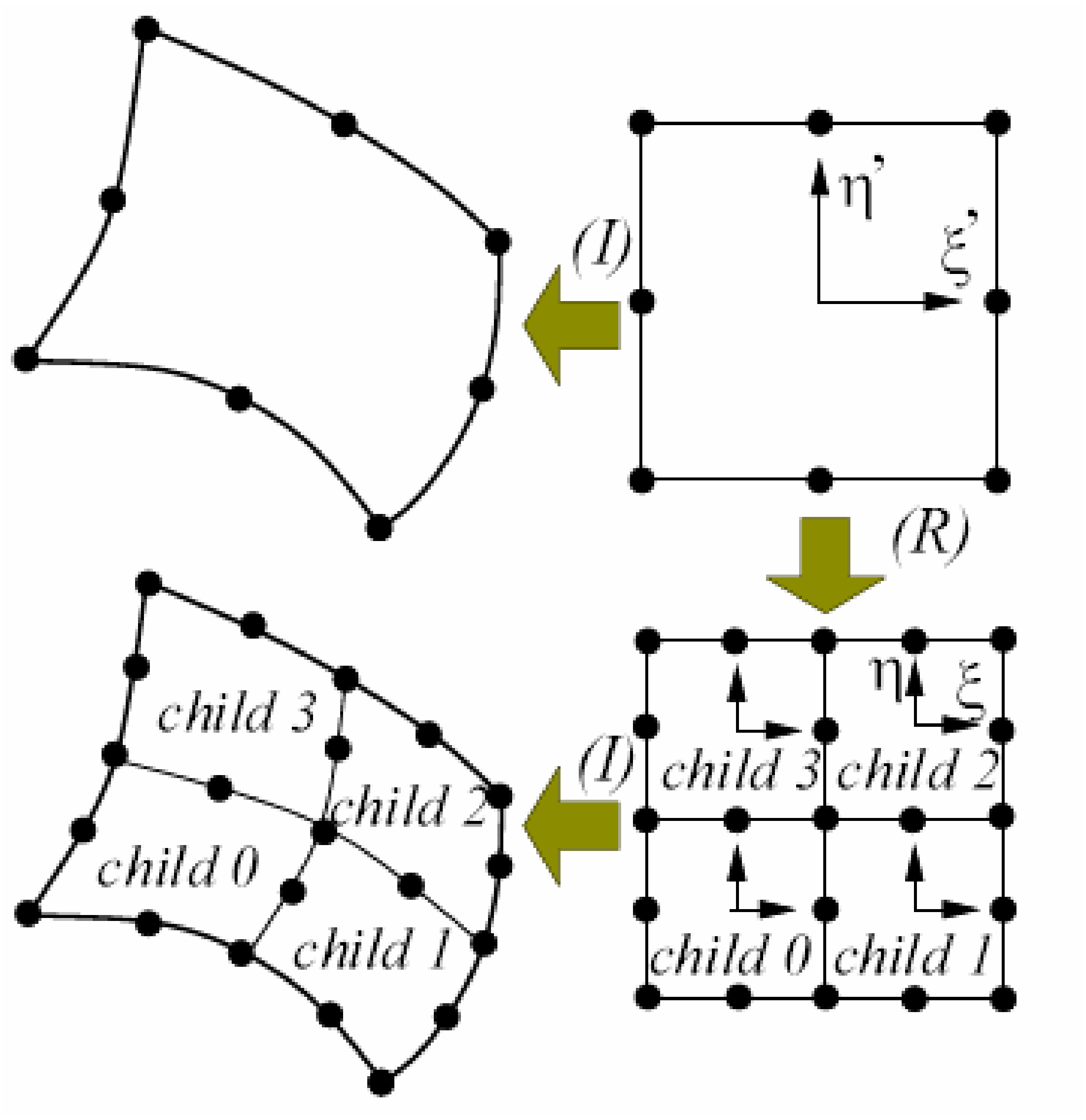


**Figure 2-2:** **Refinement of 8-node quadrilateral finite elements: (I) stands for isoparametric mapping; (R) stands for refinement.**

### 2.3.1 True Hierarchical Basis

Equation (2-3) may be viewed also as a statement of equivalence: the left hand side (the coarse function) is equivalent to the right hand side (the set of finer functions) in the sense that if both sides of Equation (2-3) are combined, a linearly-dependent set is obtained. Furthermore, it may be noted that there is just one function too many in the combined set of the left and right hand side functions. Therefore, all but one of the right hand side (finer) functions could be moved to the left hand side:

$$\phi_i^{(j)}(x) - \sum_{k \neq m} \beta_{ik}^{(j+1)} \phi_k^{(j+1)}(x) = \beta_{im}^{(j+1)} \phi_m^{(j+1)}(x) \qquad \text{(2-5)}$$

This is a statement of the same kind as before, an equivalence, but now the function that is being "reproduced" is on the finer level. Symmetry considerations dictate that the function to leave on the right hand side is the one whose node "stems" from the node of the coarse function. For instance, Figure 2-3 shows the patch (one triangle and three quadrilaterals) that support the function indicated by the filled circle (left). In the right hand part is shown the refined patch, where the filled circles indicate functions on level j + 1 on the left hand side of Equation (2-5), and the empty circle stands for the single function on the right hand side, $\phi_m^{(j+1)}(x)$; in other words, the filled circles indicate the linearly independent functions, the empty circle represents the one finer function that is dependent upon the rest. We shall call the functions on level j + 1 that were moved to the left hand side of Equation (2-5) the ***detail functions*** of $\phi_i^{(j)}(x)$.

The above discussion may suggest to the reader a way of constructing the defined approximation spaces: To refine a function from the coarse space *B*, add its detail functions to $B$ to obtain $B'$. The resulting approximation will be dubbed the ***true hierarchical*** basis, and, indeed it is not a new concept in itself, since we get the well-known hierarchical basis discussed, for example, by [52]. However, our use of this concept for selective refinement is novel.

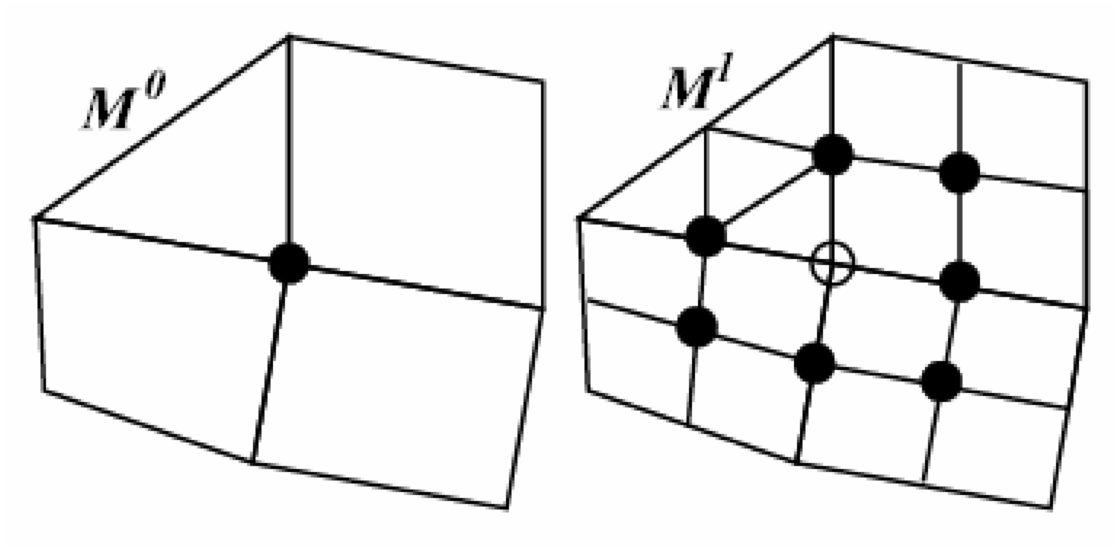


**Figure 2-3: Illustration of Equation (2-6)**

## 2.3.2 Quasi-Hierarchical Basis

To formalize somewhat, we shall use the term ***active function*** for functions selected from a given basis set, and the symbol $\hat{B}^{(k)}$ will be used for the set of active functions from basis set $B^{(k)}$. Now, the approximation basis function set $B'$ may be written as

$$B' = \bigcup_{j=0}^{\infty} \hat{B}^{(j)}, \qquad (2\text{-}6)$$

that is as a union of active sets from all mesh levels. For the true hierarchical basis described above, the active set on level j = 0 includes all the basis functions defined on the initial mesh, and each active set on level j > 0 includes only the detail functions. Now we shall describe an alternative refinement strategy, the quasi-hierarchical basis.

As before, we proceed from the refinement equation (2-3). As opposed to the true hierarchical approximation, we shall interpret the refinement equation as a recipe for *replacing* coarse functions (left hand side) by “finer” (higher-resolution) functions (right hand side). Thus, the refinement will delete (deactivate) coarse functions, replacing them in a lossless manner by finer functions, and unrefinement will delete (deactivate) fine functions and activate coarse functions. Clearly, if this type of refinement is applied globally, all coarse functions are replaced by fine functions and the interpolating partition of unity form of finite element approximation is recovered; on the other hand, if the refinement is graded, transition needs to be made from coarse to fine functions, and in the transition regions the resulting approximation resembles the true hierarchical basis in that the basis functions do not necessarily add up to unity. This is illustrated in Figure 2-4, where on the left we show the true hierarchical basis, compared to the quasi-hierarchical basis on the right. Even though we replace coarse functions instead of augmenting them with finer functions, the resulting approximation basis is still written in the form of equation (2-6), the only change being that the active sets $\hat{\beta}^{(j)}$ may now include not only the detail functions, but rather all the functions from the right hand side of the refinement equation (2-3).

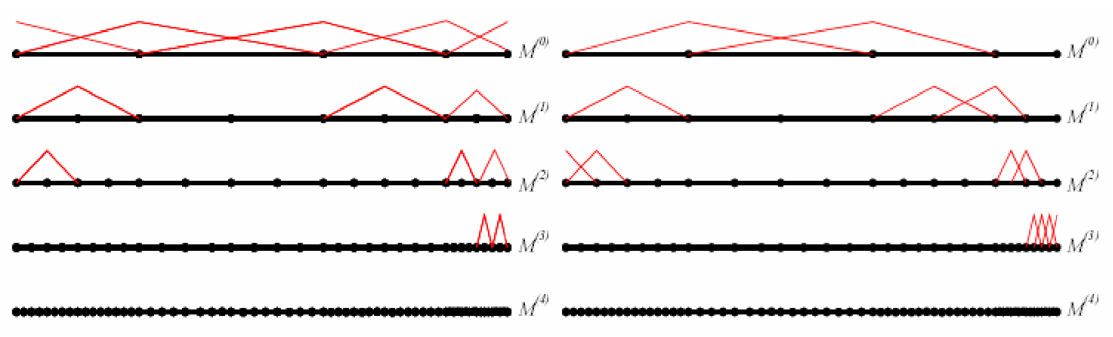


**Figure 2-4: Comparison of true hierarchical basis (left) with quasi-hierarchical basis (right).**

## 2.4 Approximation on CHARMS-Refined Meshes

Both the quasi-hierarchical and the true hierarchical approximation may be expressed in exactly the same way, and in fact often it is possible to devise refinement sequences that yield precisely the same spans for both basis types. This is not surprising, since both basis types follow from the refinement equation, and as shown in the next section, the construction of both types of bases may be expressed as the activation or deactivation of basis functions from the conceptual hierarchy of nested meshes. The equivalence of the quasi-hierarchical and the hierarchical basis allows us to write the finite element approximation in the unified way

$$u_h(x) = \sum_{j:\hat{B}^{(j)} \neq 0} \left( \sum_{i:\phi_i^{(j)}(x) \in \hat{B}^{(j)}} \phi_i^{(j)} u_i^{(j)} \right), \tag{2-7}$$

where $\hat{B}^{(j)}$ is the set of active functions on level $j$, and $u_i^{(j)}$ are the nodal parameters. Evidently, it would be possible to express the finite element approximation with a single sum, as usual, but the above form makes it clear that the basis functions "live" on different refinement levels.

The approximation properties of the CHARMS-adapted basis set are identical for the discretization error to the classical finite element basis using the same type of element. On the other hand, the hierarchical character of the basis may lead to deterioration in the conditioning of the system matrices [65]. This will be more pronounced for the true hierarchical basis than for the quasi-hierarchical basis. For a small number of overlaps (up to about five levels of refined basis functions interacting in the transition regions) the effect seems minor. However, so far this aspect has not been studied in sufficient detail.

It remains to formulate algorithms for the construction of the active sets, and that is the goal of the next section.

## 2.5 Formal Statement of the Refinement and Unrefinement Algorithms

It is advantageous for applications when the active functions constitute a *basis*, i.e. when they are linearly independent. In order to make the present chapter self-contained, we rephrase here the algorithms enunciated in [66], but without proofs. These algorithms activate and deactivate functions from the nesting hierarchy, and guarantee the linear independence of the active function set.

For the finite element meshes we consider in this chapter, no basis function support is entirely enclosed by the support of another basis function, i.e.,

$$\sup\left[\phi_k^{(j)}\right] \not\subset \sup\left[\phi_i^{(j)}\right] \text{ for } k \neq i \text{ and } \forall j \geq 0 . \qquad (2\text{-}8)$$

This is an important tool in our proofs [66]: Consider two functions on the same level; because of the above property, the refinement set of either must contain at least one function not present in the refinement set of the other.

There are many possible algorithms for building adapted bases, and here we present algorithms based upon three **rules**:

1. The refining/unrefining of a function on level $j$ may affect that function or any of its children on level $j + 1$; no other function may be involved.
2. A function on level $j + 1$ ($j + 1 \geq 1$) may be refined only when all its parents on level $j$ have been refined.
3. A function on level $j$ may become unrefined only if (a) it is currently refined and (b) none of its children on level $j + 1$ are refined.

If the basis functions are completely supported by one ring of elements around a node, then rules 2 and 3 enforce the common rule of one-level-difference refinement of neighbors; this rule has been applied to finite element meshes, as well as to spatial data structures such as quadtrees and octrees in various contexts (graphics, mesh generation, spatial searches, etc.) [67,68].

We now show that if the (un)refinement is applied atomically, i.e. it is either entirely executed or not at all, then linear independence of the adapted basis is guaranteed. Furthermore, our algorithms ensure that the refinement step is *lossless*; the span of the resulting set includes the span of the original set. (Unrefinement is not lossless, in general: some information is always going to be lost, since the goal of unrefinement is to decrease the span of the approximation space.)

### 2.5.1 Quasi-Hierarchical Basis

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

#### 2.5.1.1 Refinement

The ***refinement set*** of the coarser function $\phi_i^{(j)}$ supported by the mesh on level $j$+1 is denoted by $R^{(j+1)}\left[\phi_i^{(j)}\right]$, and contains exactly those basis functions that contribute to the right-hand side of the refinement equation with a non-zero coefficient:

$$R^{(j+1)}\left[\phi_i^{(j)}\right]=\left\{\phi_k^{(j+1)}\middle|\beta_{ik}^{(j+1)}\neq 0\right\}. \tag{2-9}$$

If $\phi_k^{(j+1)}$ belongs to $R^{(j+1)}\left[\phi_i^{(j)}\right]$, we say that "$\phi_i^{(j)}$ is a ***parent*** of $\phi_k^{(j+1)}$," and "$\phi_k^{(j+1)}$ is a ***child*** of $\phi_i^{(j)}$."

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

We claim that quasi-hierarchical refinement (a) preserves the linear independence requirement and (b) is lossless. See reference [66] for proofs.

### 2.5.1.2 Unrefinement

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

$$\left\{\phi_m^{(j+1)} \in R^{(j+1)}\left[\phi_i^{(j)}\right] \wedge \ \forall\ r \neq i \quad \phi_m^{(j+1)} \in R^{(j+1)}\left[\phi_r^{(j)}\right] \rightarrow \phi_r^{(j)} \in \hat{B}^{(j)}\right\}. \qquad \text{(2-10)}$$

We refer to this algorithm, which maps *B* to $B'$, as *quasi-hierarchical unrefinement*. We claim that quasi-hierarchical unrefinement preserves the linear independence requirement. (See reference [66] for proofs)

## 2.5.2 Hierarchical Basis

We have completed the algorithm of quasi-hierarchical refinement, which treats refinement as the *replacement* of coarse-level functions by finer-level functions. Let us turn to the alternative strategy for constructing adapted bases: hierarchical refinement treats refinement as the *addition* of finer-level "detail functions" to an unchanged set of coarse-level functions. Before we continue, let us formalize the concepts of a detail function and a detail (function) set that had been introduced above.

**Definition**: Given a function $\phi_i^{(j)}$, construct the set of all functions $\phi_k^{(j+1)} \in R^{(j+1)}\left[\phi_i^{(j)}\right]$ such that they vanish at the location $x_i$ of node *i*

$$D^{(j+1)}\left[\phi_i^{(j)}\right] = \left\{\phi_k^{(j+1)} \middle| \phi_k^{(j+1)} \in R^{(j+1)}\left[\phi_i^{(j)}\right] and\, \phi_k^{(j+1)}(x_i) = 0\right\}. \qquad \text{(2-11)}$$

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

Note that our definition of the detail set guarantees that there is *precisely one* fine function $\phi_i^{(j)}$ such that $R^{(j+1)}\left[\phi_i^{(j)}\right] = D^{(j+1)}\left[\phi_i^{(j)}\right] \cup \phi_i^{(j+1)}$.

### 2.5.2.1 Refinement

Given an initial basis function set, *B*, which contains $\phi_i^{(j)}$, and satisfies the linear independence requirement, we choose to produce a refined set $B'$ by activating $D^{(j+1)}\left[\phi_i^{(j)}\right]$. We refer to this algorithm, which maps $B$ to $B'$, as *hierarchical refinement*. As proved in reference [66], the hierarchical refinement (a) preserves the linear independence requirement and (b) is lossless.

### 2.5.2.2 Unrefinement

Given an initial basis function set, *B*, that satisfies the linear independence requirement, and given a previously refined $\phi_i^{(j)}$, also in *B* and eligible for unrefinement (rule 3), we choose to produce another function set $B'$ by deactivating functions $\phi_m^{(j+1)} \in D^{(j+1)}\left[\phi_i^{(j)}\right]$ that are absent from all the refinement sets of currently refined functions on level *j*. We refer to this algorithm, which maps $B$ to $B'$, as *hierarchical unrefinement*. It is easy to show that hierarchical unrefinement preserves the linear independence requirement [66].

## 2.6 Design of an Adaptive Solver

In this section we will pursue the discussion of the CHARMS technology into the design phase, which will be illustrated with UML diagrams [70]. Figure legends use slanted type to indicate abstract classes; the inheritance and delegation relationships are included with multiplicities.

### 2.6.1 Geometric Cell (GCELL)

The computational domain is assumed to be a non-manifold object embedded in a *d*-dimensional Euclidean space. The constituent manifolds are assumed to be locally coordinatized by $m$ Cartesian coordinates, $m \le d$, depending on the manifold dimension.

The domain is discretized into geometric cells. The manifold dimension of these cells corresponds to the manifold being discretized. Geometric cell (GCELL) is one of the basic classes in our implementation. However, GCELL is an abstract class, and only its specializations are being instantiated. The specialization branches in the direction of manifold dimension: thus there are geometric cells of manifold dimension zero (0) (GCELL_POINT_P1), of one (1) (e.g. GCELL_LINE_L2), two (2) (for instance GCELL_SURF_Q4, or GCELL_SURF_T3), and three (3) (GCELL_SOLID_H8, or GCELL_SOLID_T4). Figure 2-5 shows the class diagram for the quadratic six-node triangle (GCELL_SURF_T6).

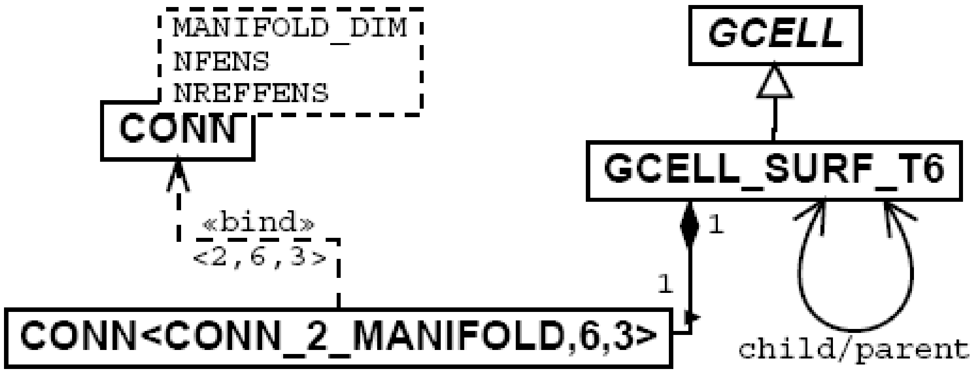


**Figure 2-5: Class diagram for a GCELL representing a six-node (quadratic) triangular finite element. Note the use of the templated CONN class for the specification of the connectivity.**

It is assumed that the cells in the physical space are images of “master” cells in the parametric domain. The parametric coordinates constitute the charts for the pieces of the manifolds embedded in the *d*-dimensional Euclidean space to which the master cell maps. Each cell may be mapped into a different Cartesian coordinate system, but for simplicity we shall assume in this chapter that all cells map to a single, global Cartesian coordinate system.

Geometric cells (GCELL's) provide a number of services to the adaptive code. Thus, they are the means of defining pieces of basis functions (in the master parametric domain), but they are also carriers of the topological/geometrical refinement hierarchy. In other words, the *h*-refinement involves division of a GCELL into finer, nested GCELL’s. It is assumed here that this division results in GCELL’s of the same type. The topological/geometrical refinement is applied recursively, resulting in a tree of GCELL’s, the root being represented by the coarsest GCELL, and the leaves being cells without children. To give an example, Figure 2-6 shows the hierarchy of

GCELL's stemming from the root at the bottom, with the finest cells as the leaves at the top, for a four-node quadrilateral.

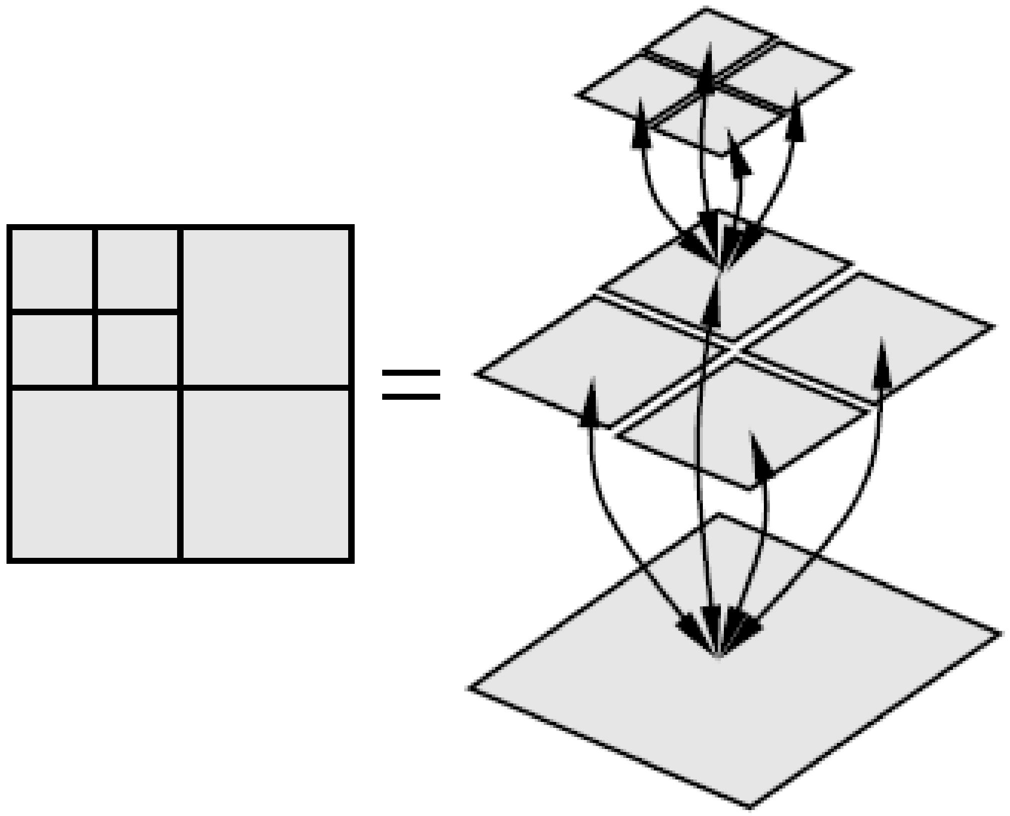


**Figure 2-6: Schematic refinement tree for a quadrilateral GCELL. Upward pointing arrows symbolize the "child" relationship. Downward arrows point at the parent.**

### 2.6.2 Connectivity and Refinement Nodes

Connectivity CONN is a template class, parameterized with the manifold dimension, number of connected nodes, and number of refinement nodes (Figure 2-5). This allows us to distinguish between a line with 3 connected nodes, and a 3-node triangle, etc. From a refinement point of view, a very important connectivity type is the vertex (CONN<CONN_0_MANIFOLD,1,1>). All refinement nodes topologically located at the nodes of the coarser levels are classified on the connectivity CONN<CONN_0_MANIFOLD,1,1>, which encodes a topological division rule. In order to maximize the number of element types that may be mixed in a single finite element mesh, we adopt the strategy that views elements of manifold dimension *d* as boundaries of elements of manifold dimension *d* + 1. This information is encoded in the instantiations of the template class CONN. Thus, vertices of cells are topological entities CONN<CONN_0_MANIFOLD,1,1>; that is their manifold dimension is zero, they connect a single node, and they are refined with one node. The edges of cells are cells in their own right, and vertices are their boundaries. For instance, CONN<CONN_1_MANIFOLD,2,1> is a two-node line segment, which is refined with a single node at the midpoint (origin of the parametric coordinates). Three-node triangles (CONN<CONN_2_MANIFOLD,3,0>) and four-node quadrilaterals (CONN<CONN_2_MANIFOLD,4,1>) are bounded by connectivities CONN<CONN_1_MANIFOLD,2,1>, which makes it possible to mix those two in a single mesh. Three-node triangles have no refinement nodes of their own (all refinement nodes are located at the edges, none in the interior), whereas the four-node quadrilaterals have one interior refinement node. As the last illustration, consider the quadratic isoparametric triangle. The class diagram for the quadratic six-node triangle in Figure 2-5 refers to the connectivity type CONN<CONN_2_MANIFOLD,6,3>, that is a 2-manifold that connects 6 nodes, and has 3 internal refinement nodes. The boundaries of this cell are of type CONN<CONN 1 MANIFOLD,3,2>, and the connectivity of the quadratic triangle is compatible with the refinement of the 10-node tetrahedron.

The topological division starts with the connectivities of the lowest manifold dimension; that is zero (vertices). Then edges are divided, followed by faces, and finally volumes. Thus, an eight-node hexahedron would first divide its vertices, then its edges, faces, and finally it would conclude the process by generating one refinement node at its barycenter.

### 2.6.3 Field

The class FIELD is a fundamental concept in our software framework. The term "field" implies spatial variation [71], and in the present context, field *u* is represented by the sum

$$u_h(x) = \sum_j \phi_j(x) u_j \,, \tag{2-12}$$

where $u_h(x)$ is the usual finite element expansion of function $u(x)$, $\phi_j(x)$ are the finite element basis functions, and $u_j$ are the nodal parameters. The approximated function may be a scalar, vector, or tensor function of a vector argument, e.g. the displacement field over the domain (vector function of a vector argument) or the temperature field (scalar function of a vector argument). Each term in the sum of equation (2-12) is expressed through an object, the FIELD_PAIR. It relates two objects: the basis function, BFUN, and the degree-of-freedom parameter (DOFPARAM for short). The value of the DOFPARAM object may change freely, or some or all of its components might be prescribed as an expression of an essential boundary condition. FIELD_PAIR is a class parameterized with the number of components in the DOFPARAM object, and in that way the current implementation is able to represent scalar degrees of freedom (one component), or vectors in 3D space (three components), or an array of an arbitrary number of components.

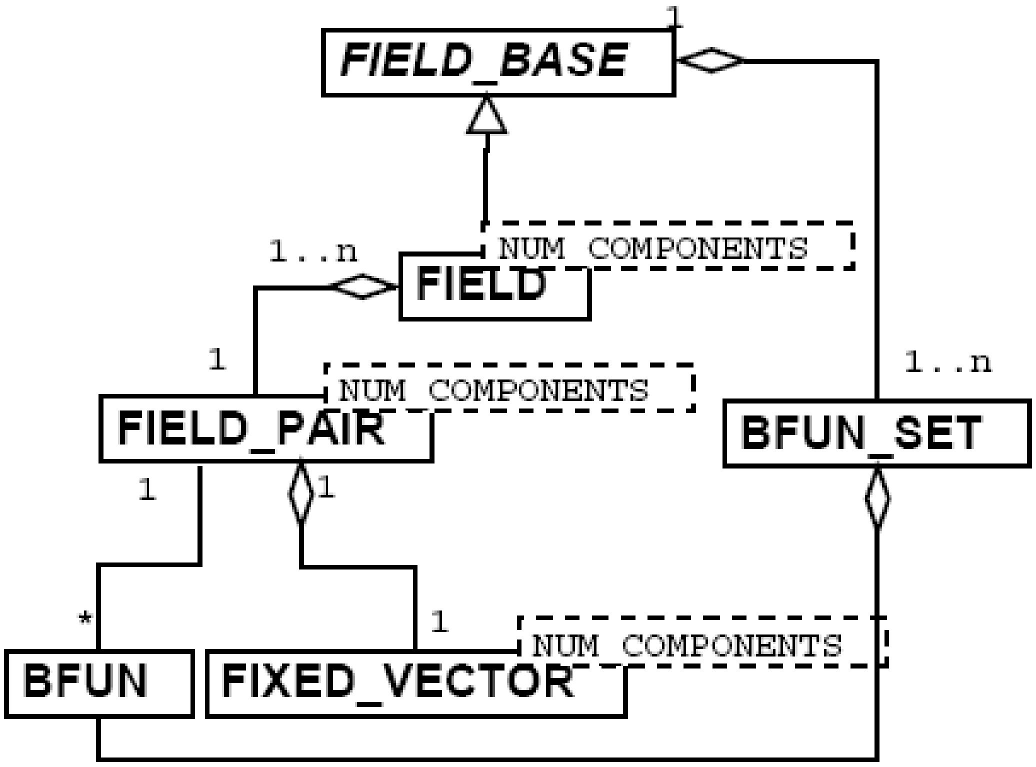


**Figure 2-7: Class diagram for the field pair and field.**

### 2.6.4 Basis Function (BFUN)

Figure 2-8 shows the class diagram for the basis function and basis function set types. BFUN serves as the abstract base class. BFUN_FE builds up the first concrete class upon BFUN and provides access to the standard finite element basis functions. BFUN_FE maintains a list of all the GCELL's over which the individual pieces of the given basis function are defined. As an example, the basis function associated with the node indicated by the black dot in the coarse mesh in Figure 2-3 would include one triangle (GCELL_SURF_T3) and three quadrilaterals (GCELL_SURF_Q4).

### 2.6.5 Basis Function Set (BFUN_SET)

The basis function set (BFUN_SET) collects basis functions defined on a particular geometric mesh. Any particular BFUN_SET maybe associated with any number of fields. The BFUN_SET provides the field with a very important service: The BFUN_SET generates an opaque identifier (BFUN_DOFPARAM_PAIR_ID) which may be used to access a field pair in constant time (it is in fact implemented as an array access). The BFUN_SET generates this identifier based on the pointer to an opaque, system-wide unique identifier, which is carried by a finite element node. That in turn implies that a one-to-one link exists between the unique identifier (finite element node) and a basis function. For example, the BFUN_DOFPARAM_PAIR_ID identifier is used by solvers to distribute computed solutions to the field pairs.

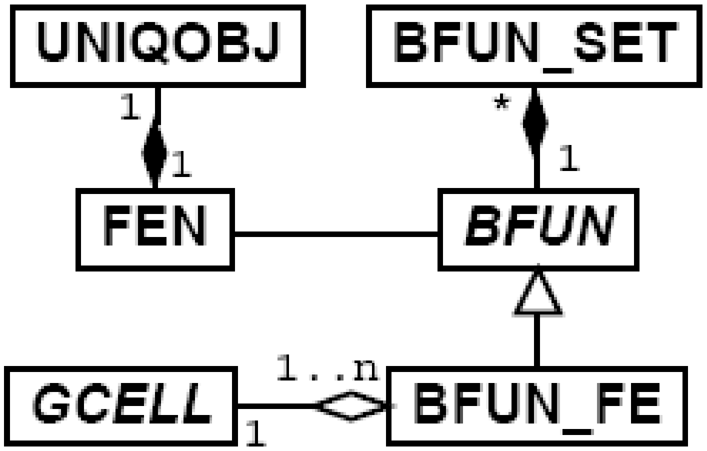


**Figure 2-8: Class diagram for the basis function and basis function set.**

## 2.7 Implementation of Refinement/Unrefinement

The implementation of the refinement and unrefinement algorithms as enunciated earlier is straightforward. However, it seems worthwhile to point out that the refinement of a single basis function involves the generation of child GCELL's, which in turn requires the addition of refinement nodes. Some of these nodes are shared by two or more GCELL's, and care must be taken to introduce only a single copy of each shared refinement node so that compatibility of the refinement functions is ensured.

To enable the sharing of refinement nodes during the creation of the child geometric cells, we use the concept of a refinement context. It acts as the dispenser of refinement nodes. The present code labels refinement nodes with an instance of the connectivity object on which they are classified. For instance, for an eight-node hexahedron, the refinement nodes are either located at the vertex nodes, at the mid-points of the edges, at the barycenters of the faces, or at the barycenter of the volume: compare with Table 2-1. The request to supply a refinement node on a given connectivity object (for instance, on an edge connecting nodes *I* and *J*) is processed by the refinement context: If the refinement node had not been created for the connectivity object before, it is created, and the reference to the connectivity is remembered for future lookups. Each GCELL may thus proceed with the refinement independently of its neighbors; the sharing of refinement nodes is managed transparently.

**Table 2-1: Refinement Nodes of an 8-node Hexahedron**

| Location of refinement node | Classified at (`CONN<MANIFOLD_DIM,NFENS,NREFFENS>`) |
|---|---|
| Vertex | `CONN<CONN_0_MANIFOLD,1,1>` |
| Edge | `CONN<CONN_1_MANIFOLD,2,1>` |
| Face | `CONN<CONN_2_MANIFOLD,4,1>` |
| Volume | `CONN<CONN_3_MANIFOLD,8,1>` |

### 2.7.1 Example: Equation of Steady Diffusion

To illustrate the adaptive code, we shall consider a simplified version of the linear partial differential equation of steady diffusion. In strong form:

Given $f:\Omega \leftarrow R, g:\Gamma_g \leftarrow R$, and $h:\Gamma_h \leftarrow R$, find $u:\overline{\Omega} \leftarrow R$

$$\begin{aligned} -\kappa\Delta u &= f \quad \text{in } \Omega \\ u &= g \quad \text{in } \Gamma_g \\ u,_i n_i &= h \quad \text{in } \Gamma_h , \end{aligned}$$

where $\Omega$ is the domain (two- or three-dimensional), $\Gamma_h$ and $\Gamma_g$ are disjoint parts of the boundary, $\Gamma = \Gamma_h \cup \Gamma_g, \Delta$ is the Laplace operator, ",*i*" in the subscript means differentiation with respect to the *i*-th Cartesian coordinate, $\kappa$ is the conductivity (assumed to be constant in $\Omega$), and the functions *g* and *h* are the prescribed boundary values. A weak form of equation (2-13) is derived in a standard way [69] as

$$a(w,u) = (w,f) + (w,h)_{\Gamma_h}, \qquad \text{(2-13)}$$

where

$$a(w,u) \quad = \quad \int_\Omega \kappa w_{,i} u_{,i} \quad d\Omega, \qquad \text{(2-14)}$$

$$(w, f) = \int_{\Omega} wf \quad d\Omega, \tag{2-15}$$

$$(w,h)_{\Gamma_h} = \int_{\Gamma_h} wh \quad d\Gamma_h . \tag{2-16}$$

The functions $u$ and $w$ are the trial and test functions respectively, $u \in S$ and $w \in V$, where $S$ is the trial space and $V$ is the test space.

Next, we choose finite dimensional approximations of the trial and test spaces as the span of some suitable finite element basis functions. Adopting a Galerkin formulation, the test functions $v^h \in V^h$ will all vanish on the boundary $\Gamma_g$, and the trial functions $u^h$ will be decomposed as

$$u^h = v^h + g^h, \tag{2-17}$$

where $g^h$ will satisfy (approximately) the boundary condition $u = g$ on $\Gamma_g$, and $v^h \in V^h$.

The homogeneous-part of the trial functions is written as

$$v^h = \sum_B \phi_B d_B ,$$

which leads to the discrete system of linear equations

$$\sum_{B \notin \Gamma_h} a(\phi_A, \phi_B) d_B = (\phi_A, f) + (\phi_A, h)_\Gamma - \sum_{B \in \Gamma_h} a(\phi_A, \phi_B) g_B , \; A \notin \Gamma_h . \tag{2-18}$$

On the left-hand side of (2-18) is the product of the conductivity matrix with the vector of unknowns, and the source terms are all on the right-hand side.

### 2.7.1.1 Hexahedral Discretization in 3D

Hexahedral finite elements are often preferred to tetrahedra because of their slightly better performance in a number of applications. However, refining hexahedra by introducing compatible edges, or by using a technique similar to the mesh refinement of tetrahedra by bisection is a tough problem, since the element quality tends to deteriorate very quickly and without bounds. Due to the regular division of the elements by octasection, CHARMS has no problem with shape deterioration. The implementation of mesh refinement with CHARMS also proves extremely easy, and in fact it took the first author just a couple of hours to add the hexahedral refinement to the CHARMS framework once the code had been debugged in one dimension. Most of the implementation effort goes into the coding of the connectivity of the parent and its children: the information about which nodes are connected by a given child needs to be recorded.

Figure 2-9 shows the mesh and the results of a sample simulation for a steady diffusion problem solved on a polyhedral domain. The initial grid and a refined grid are compared side-by-side. The integration cells are shown in Figure 2-9 and the reader should take care to realize the apparently hanging nodes are a visualization artifact: no active basis functions are associated with the incompatibly located nodes.

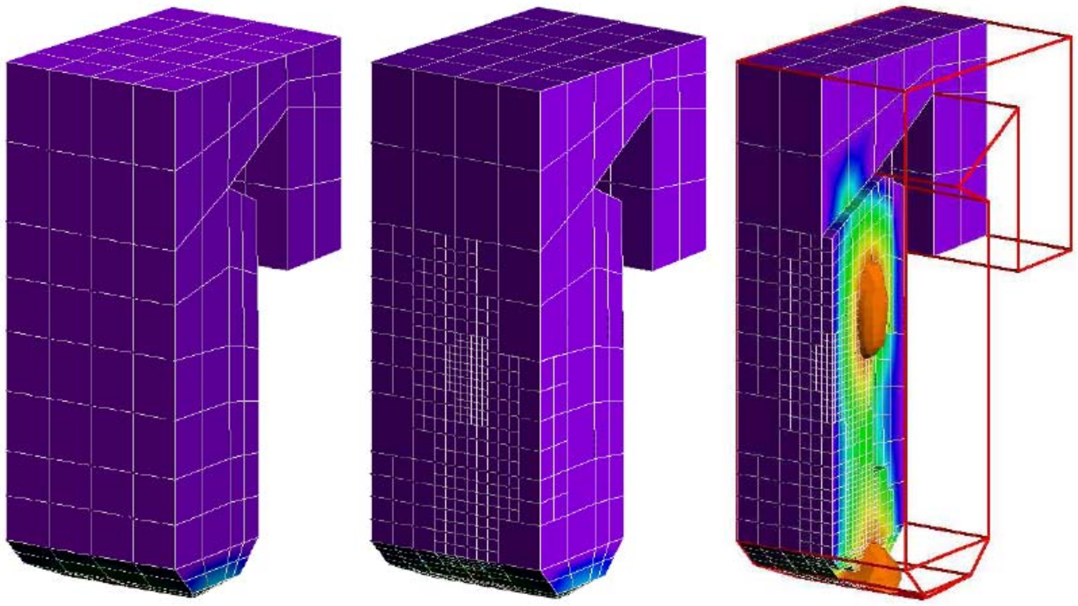


**Figure 2-9: Steady diffusion equation solved in a three-dimensional domain. Left: initial grid; middle: step after two adaptations; right: cut-off grid with a solution contour. The integration cells are shown, with color coding corresponding to the temperature distribution.**

### 2.7.1.2 Refinement of Triangulations with Quadratic Triangles

As an example of mesh refinement for higher-order approximations we discuss here an implementation for the quadratic (6-node) triangles. Quadrisection is used to divide each parent triangle into four children of the same type in the parametric space: see Figure 2-10. The refinement nodes are numbered from 0 to 14. Refinement nodes 0 to 5 are classified at the nodes of the parent, nodes 6 to 11 are classified on the 3-node edges of the parent, and the last three nodes (12, 13, and 14) are classified in the interior of the parent element. An important piece of information that the refinement algorithm needs is which functions at the nodes of the child refine a particular function of the parent. For instance, function at node 0 of the parent is refined by functions at refinement nodes 0, 6, 7, 13, 14, 10, 11 of the children (all but the function at node 0 are detail functions); function at node 4 of the parent is refined by functions at nodes 4, 8, 9, 12, 13, 14 of the children (all but the first are detail functions), etc. Figure 2-11 shows the contours of the solution of the Poisson equation for a "dipole" source function. The results are displayed on the integration cells. The visualization approximates the smooth quadratic variation by piecewise linear polygons. Figure 2-12 illustrates the active basis function set for the refined mesh in the upper-left corner of the domain from Figure 2-11 by showing the active functions as red balls, and the geometric cells that support the active functions are filled triangles. Note that the active functions on level $j$+1 vanish along the boundaries of the geometric cells of level $j$. That is a visual confirmation of the preserved compatibility.

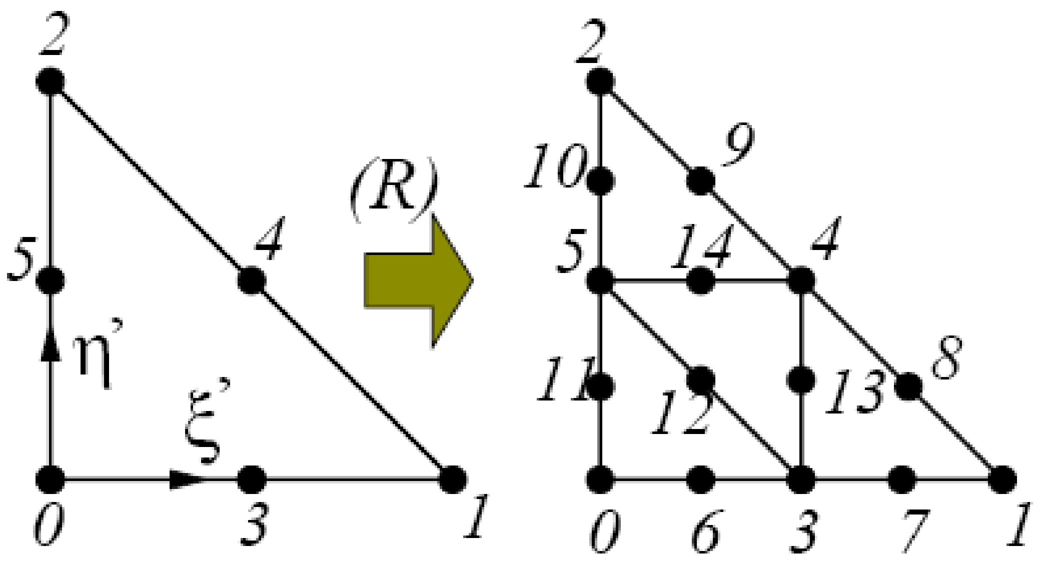


**Figure 2-10: Refinement of the 6-node triangle**

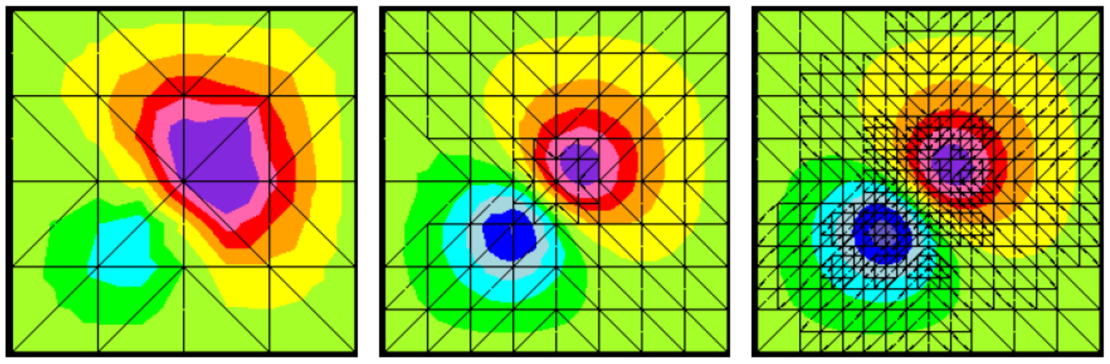


**Figure 2-11: Dipole equation with homogeneous boundary conditions solved on a triangulation of square domain with 6-node quadratic triangles. Color coding of the field on three refined grids.**

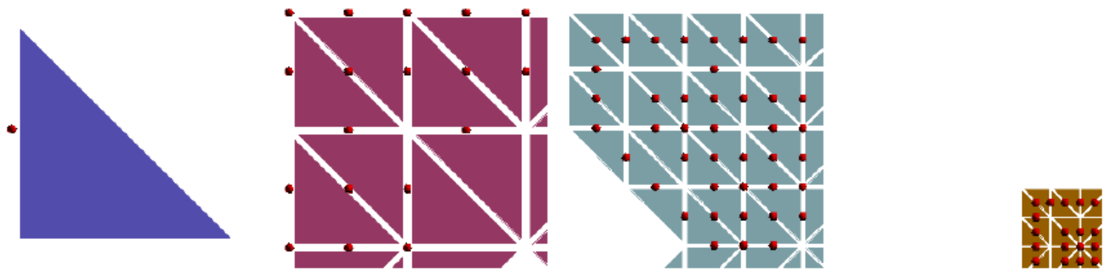


**Figure 2-12: Dipole equation, 6-node quadratic triangles. The geometric cells from the upper-left corner of the computational domain that support basis functions on different levels: left to right, level 1, 2, 3, 4. The red balls indicate the active basis functions.**

### 2.7.2 Example: Equation of linear elasticity

In strong form, the equation of linear elasticity can be written as following: Given $f:\Omega \leftarrow R, g:\Gamma_g \leftarrow R$, and $h:\Gamma_h \leftarrow R$, find $u:\overline{\Omega} \leftarrow R$

$$\begin{array}{lcll} \sigma_{ij,j} + f_i & = & 0 & \text{in } \Omega \\ u_i & = & g_i & \text{in } \Gamma_{g\,i} \\ \sigma_{ij} n_j & = & h_i & \text{in } \Gamma_{h\,i}, \end{array}$$

Here $\Omega$, $\Gamma_h$, $\Gamma_g$ ,*i*, *g* and *h* are the same as defined in the section 2.2.3. A weak form of equation (2-19) is derived in a standard way [69] as

$$a(w,u) = (w, f) + (w,h)_{\Gamma_h}, \tag{2-19}$$

where

$$a(w,u) \quad = \quad \int_\Omega w_{(i,j)} c_{ijkl} u_{,(k,l)} d\Omega, \tag{2-20}$$

$$(w, f) \quad = \quad \int_\Omega w_i f_i \quad d\Omega, \tag{2-21}$$

$$(w,h)_{\Gamma_h} \quad = \quad \sum_{i=1}^{n_{sd}} \left( \int_{\Gamma_h} w_i h_i d\Gamma_h \right). \tag{2-22}$$

The functions *u* and *w* are the trial and test functions respectively, $u \in S$ and $w \in V$, where *S* is the trial space and *V* is the test space.

Next, we choose finite dimensional approximations of the trial and test spaces as the span of some suitable finite element basis functions. Adopting a Galerkin formulation, the test functions $v^h \in V^h$ will all vanish on the boundary $\Gamma_g$, and the trial functions $u^h$ will be decomposed as

$$u^h = v^h + g^h, \tag{2-23}$$

where $g^h$ will satisfy (approximately) the boundary condition $u = g$ on $\Gamma_g$, and $v^h \in V^h$.

The homogeneous-part of the trial functions is written as

$$v^h = \sum_B \phi_B d_B,$$

which leads to the discrete system of linear equations

$$\sum_{B \notin \Gamma_h} a(\phi_A, \phi_B) d_B = (\phi_A, f) + (\phi_A, h)_\Gamma - \sum_{B \in \Gamma_h} a(\phi_A, \phi_B) g_B, \ A \notin \Gamma_h. \tag{2-24}$$

On the left-hand side of (2-24) is the product of the stiffness matrix with the vector of unknowns, and the source terms are all on the right-hand side.

#### 2.7.2.1 Refinement of Tetrahedra

State of the art techniques refine tetrahedra through bisection or octasection. However, the bisection pattern on a face shared by two tetrahedra can have three cuts. Hence it is not unique. That would make it a necessity to communicate the refinement information among neighbors. Moreover, bisection has the potential to severely degrade the shape quality measure of the finer functions since the bisected element is not similar to its offspring. Although octasection of tetrahedra avoids these problems, the issue with the degrading shape quality is something that currently known octasection based approaches are struggling with. However, since the use of CHARMS ensures the construction of a conforming mesh a priori, shape quality measure does not degrade even though the geometrical mesh (i.e. supports of the basis functions) *looks* non-conforming. CHARMS uses Ong's theory to produce an adaptive strategy that guarantees shape quality without necessitating the restoration of compatibility or other complications. Further details of this technique can be found in [33].

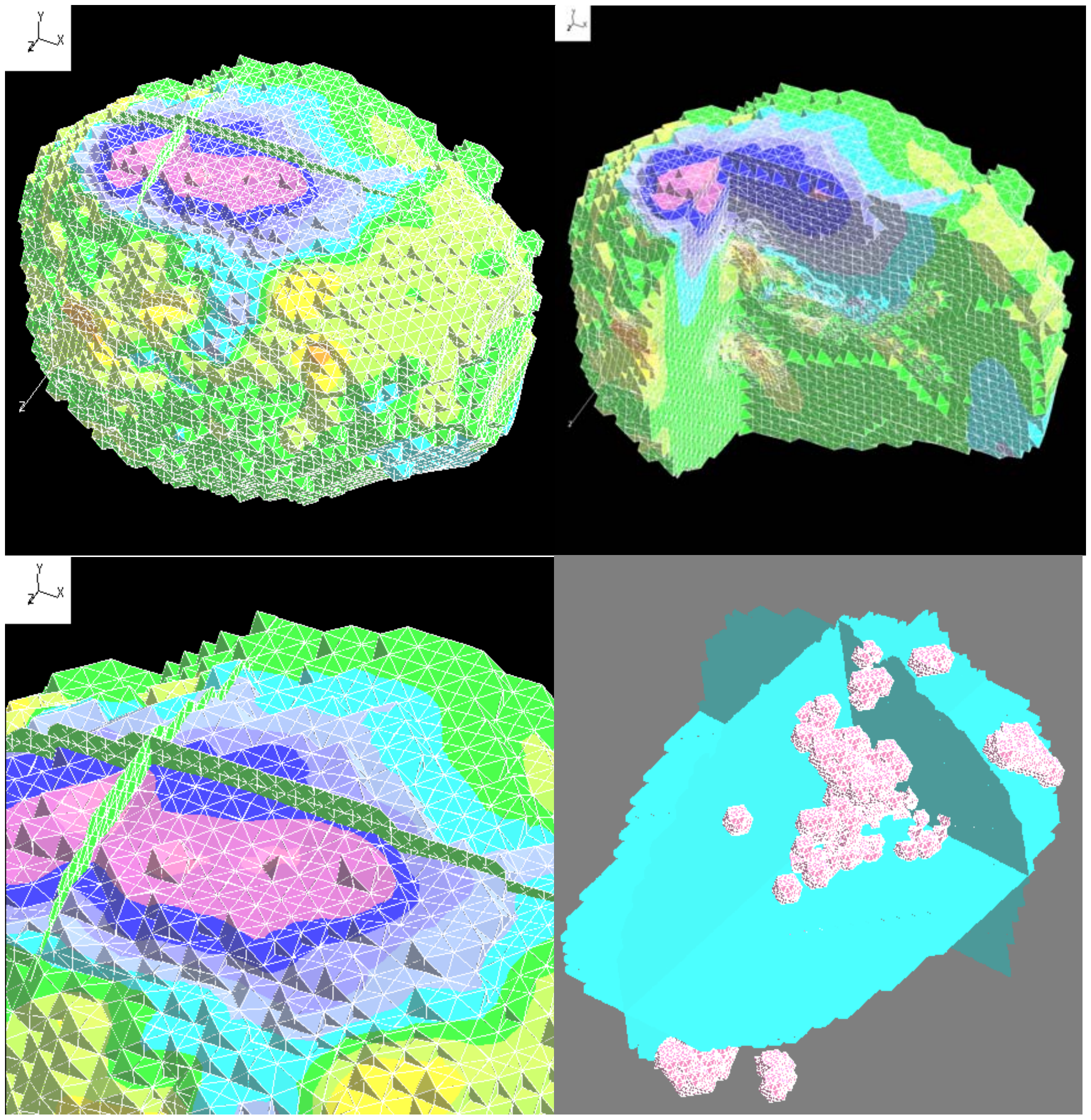


**Figure 2-13:** **Linear elastic model of human brain using tetrahedral mesh refinement. Color-coded area shows distribution of Von Mises stresses.**

# CHAPTER 3 : ADAPTIVE SIMULATION OF ELASTIC WAVE PROPAGATION IN SOLIDS

## 3.1 Equation of Elastic Wave Propagation

In strong form, the equation of elastodynamics can be written as:

Given $f:\Omega\times]0,T[\to R, g:\Gamma_g\times]0,T[\to R$, and $h:\Gamma_h\times]0,T[\to R$, find $u:\overline{\Omega}\times]0,T[\to R$

$$\sigma_{ij,j}+f_i = \rho u_{i,tt} \quad \text{in } \Omega\times]0,T[ \qquad (3\text{-}1)$$

$$u_i = g_i \quad \text{in } \Gamma_{g\,i}\times]0,T[ \qquad (3\text{-}2)$$

$$\sigma_{ij}n_j = h_i \quad \text{in } \Gamma_{h\,i}\times]0,T[ \qquad (3\text{-}3)$$

$$u_i(x,0) = u_{0i}(x) \quad x\in\Omega \qquad (3\text{-}4)$$

$$u_{i,t}(x,0) = \dot{u}_{0i}(x) \quad x\in\Omega \qquad (3\text{-}5)$$

Here $\Omega$, $\Gamma_h$, $\Gamma_g$ ,$i$, $g$ and $h$ are the same as defined in section 2.2.3. $u_{0i}:\Omega\to R$ and $\dot{u}_{0i}:\Omega\to R$ are initial conditions of displacement and velocity as described in equations 3-4 and 3-5, and $\sigma_{ijkl}=C_{ijkl}\varepsilon_{kl}$. Here $\varepsilon$ is the strain tensor. For isotropic materials, the material response tensor $C$ may be written:

$$C_{ijkl}=\lambda\delta_{ij}\delta_{kl}+\mu(\delta_{ik}\delta_{jl}+\delta_{il}\delta_{jk}).$$

A weak form of equation (3-1) is derived in a standard way [69]:

Given $f,g,h,u_0,\dot{u}_0$, find $u(t),t\in[0,t]$ such that for every $w\in V$

$$(w,\rho\ddot{u})+a(w,u)=(w,f)+(w,h)_{\Gamma_h}, \qquad (3\text{-}6)$$

$$(w^h, \rho u(0)) = (w, \rho u_0) \qquad (3\text{-}7)$$

$$(w^h, \rho \dot{u}(0)) = (w, \rho \dot{u}_0) \qquad (3\text{-}8)$$

The functions $u$ and $w$ are the trial and test functions respectively; $u \in S$, $S$ is the trial space, and $V$ is the test space.

Next, we choose finite dimensional approximations of the trial and test spaces as the span of some suitable finite element basis functions. Adopting a Galerkin formulation, the test functions $v^h \in V^h$ will all vanish on the boundary $\Gamma_g$, and the trial functions $u^h$ will be decomposed as

$$u^h = v^h + g^h, \qquad (3\text{-}9)$$

where $g^h$ will satisfy (approximately) the boundary condition $u = g$ on $\Gamma_g$, and $v^h \in V^h$. Therefore, the semi-discrete Galerkin formulation can be written:

Given $f, g, h, u_0, \dot{u}_0$, find $u^h = v^h + g^h$, $u^h(t) \in \Omega$ such that for every $w^h \in V^h$

$$(w^h, \rho \ddot{v}^h) + a(w^h, v^h) = (w^h, f) + (w^h, f)_\Gamma - (w^h, \rho \ddot{g}^h) - a(w^h, g^h) \quad (3\text{-}10)$$

$$(w^h, \rho v^h(0)) = (w^h, \rho u_0) - (w^h, \rho g^h(0)) \qquad (3\text{-}11)$$

$$(w^h, \rho \dot{v}^h(0)) = (w^h, \rho \dot{u}_0) - (w^h, \rho \dot{g}^h(0)) \qquad (3\text{-}12)$$

$v^h$ and $g^h$ can be written as:

$$v_i^h(x,t) = \sum_{A \notin \tau_k} N_A(x) d_{ia}(t)$$

$$g_i^h(x,t) = \sum_{A \in \tau_k} N_A(x) g_{ia}(t)$$

This leads to the matrix form of the equation:

Given $F:]0,T[\rightarrow \Re^{n_{eq}}$, find $d:]0,T[\rightarrow \Re^{n_{eq}}$ such that

$$[M][\ddot{d}]+[K][d]=\{F\}. \quad t\in]0,T[ \tag{3-13}$$

$$d(0)=d_0 \tag{3-14}$$

$$\dot{d}(0)=\dot{d}_0 \tag{3-15}$$

where

$$[M] = \text{ mass matrix} = \mathop{\mathrm{A}}_{e=1}^{n_{el}}(m^e),\ m^e=[m^e_{pq}]=\delta_{ij}\int_{\Omega_e} N_a\rho N_b d\Omega \tag{3-16}$$

$$[K] = \text{ stiffness matrix} = \mathop{\mathrm{A}}_{e=1}^{n_{el}}(k^e),\ k^e=[k^e_{pq}]=e_i^T\int_{\Omega_e} B_a^T DB_b d\Omega\ e_j \tag{3-17}$$

$$F(\mathrm{t}) = F_{nodal}(t)+\mathop{\mathrm{A}}_{e=1}^{n_{el}}(f^e(t)),$$

$$f^e=\{k^e_p\}=\int_{\Omega} N_a f_i d\Omega+\int_{\Gamma^e_{h_i}} N_a h_i d\Gamma-\sum_{q=1}^{n_{el}}(k^e_{pq}g^q_e+m^e_{pq}\ddot{g}^q_e) \tag{3-18}$$

If we account for viscous damping in the system equation (3-3) can be written as:

$$[M][\ddot{d}]+[C][\dot{d}]+[K][d]=\{F\} \tag{3-19}$$

where $C$ is the damping matrix.

## 3.2 Solution with Adaptive Scheme

In our case, there are three essential steps for an adaptive solver: predict, adapt and solve. Our algorithm first divides the overall time interval into a number of subintervals, depending upon the number of adaptive steps necessary. At the end of each subinterval, the solver forces the mesh to adapt to the solution process by refining and unrefining in the areas of higher and lower levels of strain energy (or using some other error indicator). The solver than restarts from initial conditions at the end of the previous subinterval and computes a little further in the time domain for a pre-specified time. We call this procedure the "Look Ahead" algorithm and the pre-specified time is called the back track time. The "Look Ahead" algorithm, combined with the use of a partition of a unity basis function set, provides us a way to deal with the loss of conservation involved in the addition and removal of basis functions from a non-interpolating basis function set.

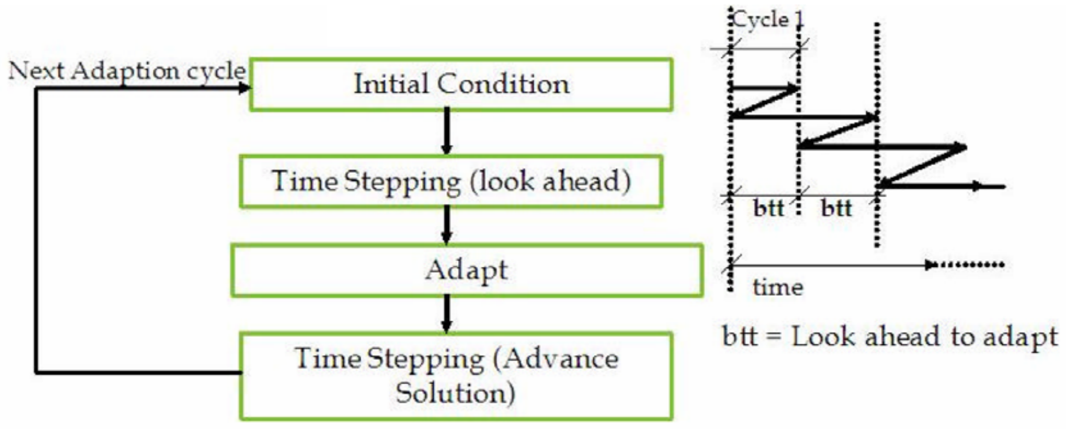

Next Adaption cycle
Initial Condition
Time Stepping (look ahead)
Adapt
Time Stepping (Advance Solution)
Cycle 1
btt
btt
time
btt = Look ahead to adapt

**Figure 3-1:** **Progress of solver.**

Given initial velocity $v_0$ and initial displacement $u_0$

Calculate Effective Force: $f_0 = -Ku_0 + R_0$ (3-20)

For $n$ = 1, 2,

Update $u$: $u_n = u_{n-1} + \Delta t v_{n-1} + \Delta t^2 M^{-1} f_{n-1}$ (3-21)

Calculate effective force: $f_n = -Ku + R$ (3-22)

Update $V$: $v_n = v_{n-1} + 0.5\Delta t M^{-1}\left(f_n + f_{n-1}\right)$ (3-23)

End

## 3.3 Required Resolution for Wave Propagation Problems

In order to adequately model elastic waves, the spatial sampling rate must be fine enough to provide adequate resolution. What “adequate resolution” may mean is defined by the requirement of the physical experiment being investigated. In general, researchers have developed ‘rules of thumb’ based on their experiences that indicate appropriate ranges for both spatial and temporal sampling rates. Some have proposed that the temporal sampling rate should be high enough to sample 20 points per cycle of the highest frequency of interest, but may require being increased to 180 points per cycle depending on the application [73]. The spatial sampling rate is determined by the size of the elements being used (which is a function of the mesh size). Some have proposed that 10 nodes per wavelength is enough for accurate results [74], while others agree that 20 elements per wavelength would provide better results [75]. In the end, the spatial and temporal sampling rates can only be justified by the results that they produce. While these ‘rules of thumb’ for sampling rates do not guarantee accurate results, they are a good starting point.

As mentioned earlier, temporal resolution is dictated by spatial resolution. Reducing the minimum mesh size will reduce the time step, causing the computational cost to increase. Therefore, spatial resolution should be chosen wisely to produce accurate results with minimum computational effort. This chapter presents an efficient solution for this problem by refining mesh in the areas indicated by the error indicator and unrefining in the areas where error in the strain energy norm falls below a particular threshold (using CHARMS).

## 3.4 Adaptive Time Stepping

Explicit solvers are generally conditionally stable. Time step is governed by a sufficient condition for stability obtained from estimates of maximum eigenvalues of individual elements. Using this condition, the stable time step for the central difference method can be derived as

$$\Delta t \leq \frac{2}{\omega_{\max}^{h}} = \frac{h}{c} \tag{3-24}$$

Here $h$ is smallest mesh size and $c$ is highest wave velocity. In the case of guided waves, spherical waves generally travel fastest, and their velocity, given by $\sqrt{\kappa/\rho}$ in elastic isotropic media (where $\kappa$ is Bulk modulus and $\rho$ is density of material), can be used to calculate a stable time step.

For the adaptive mesh refinement scheme, the time step has to be calculated during each adaptive cycle, since mesh size keeps changing due to both refinement and coarsening of mesh.

## 3.5 Error Indicator

Error in the strain energy norm is calculated at the end of each time subinterval as the wave front proceeds in the time domain. Error is then distributed over the basis functions on which the finite domain is supported. Dimensions of an ideal cube that could hold this strain energy density are calculated. The approximate minimum mesh size required to reduce this error below the prescribed limit is then calculated and compared with the characteristic dimension of the basis function. Here the characteristic dimension of a basis function refers to the maximum span of basis functions over the elements on which it is supported. The prescribed limit is controlled by parameters defined in the input files H_OVER_HBAR_REF (for refinement) and H_OVER_HBAR_UNREF (for unrefinement). If the size of the characteristic dimension of the basis function exceeds the approximate mesh size achieved from the error associated with the basis function, the basis function refines itself. The reverse is true for the unrefinement process.

$$SE = \{u\}^T [K] \{u\} \qquad (3\text{-}25)$$

$$\overline{V_i} = \frac{\overline{SE}}{SE_i} \times V_i \qquad (3\text{-}26)$$

$$\overline{SE} = \frac{\sum SE_i}{n} \qquad (3\text{-}27)$$

Here $SE$ is the strain energy associated with the basis function and $\overline{SE}$ is the amount of strain energy that a basis function can ideally hold within a cube equal to the characteristic dimension of the basis function. $V_i$ and $\overline{V_i}$ are volumes of cubes made of the ideal characteristic dimension and the characteristic dimension, respectively.

## 3.6 Partition of Unity basis

Partition of unity on a given support in the refined mesh is defined as the following:

Let $\{N_i\}$ be a collection of functions spanning over a given support. Then normalization $\phi_i = \frac{N_i}{\sum_j N_i}$ yields a partition of unity subordinate to cover the support. In classical finite element analysis, shape functions are chosen such that they form a partition of unity. Although this not a necessary requirement, construction of shape functions in this fashion simplifies the problem considerably in many instances. For example, consider the construction of a lumped mass matrix using the row sum technique:

$$m_{pq}^{e} = \sum_{b=1}^{n_{en}} \int_{\Omega_e} N_a \rho N_b d\Omega$$

If the basis function set follows a partition of unity, then $\sum_{b=1}^{n_{en}} N_b = 1$; leading to

$$\sum_{b=1}^{n_{en}} \int_{\Omega_e} N_a \rho N_b d\Omega = \int_{\Omega_e} N_a \rho d\Omega$$

In simplified form, this can be written:

$$m_{pq}^{e} = \begin{Bmatrix} \delta_{ij} \int_{\Omega_e} \rho N_a d\Omega, & a = b \\ 0, & a \neq b \end{Bmatrix} \tag{3-28}$$

Recent work done by Melenk and Babuska [86] on the 'partition of unity finite element method' further highlights the fact that when any arbitrary finite element is constructed from piecewise polynomial basis functions (if basis functions constitute a partition of unity on each support), the displacement will converge towards the exact value for elliptic boundary value problems as the mesh is refined.

Refined meshes produced by CHARMS are not necessarily partition of unity by nature because of their non-interpolating nature. Consider the refinement of a simplistic one-dimensional mesh as shown in the Figure 3-2:

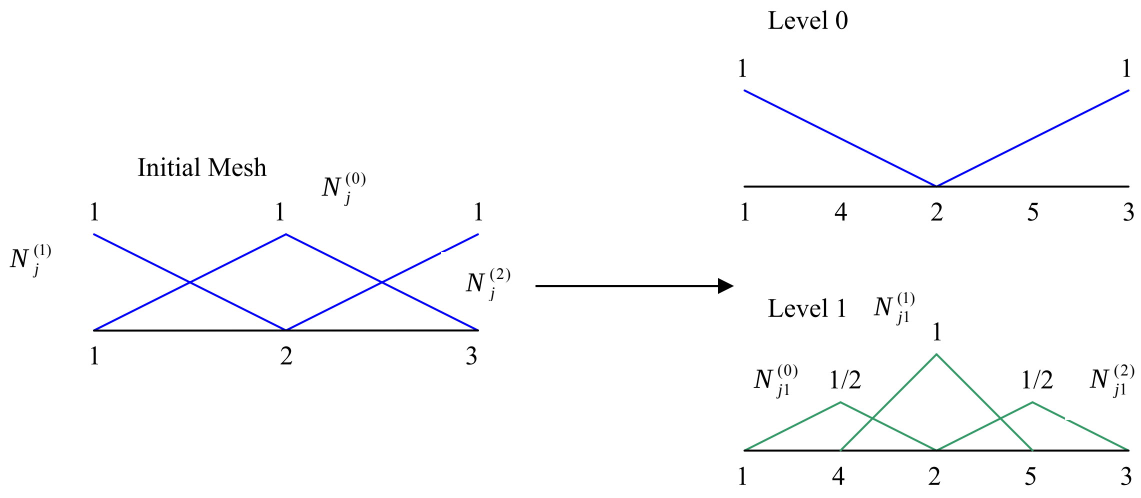


**Figure 3-2: Quasi-hierarchical refinement of shape function $N_j^{(0)}$ with partition of unity**

Here a coarse basis function $N_j^{(0)}$ is replaced with three finer basis functions $N_{j1}^{(0)}$, $N_{j1}^{(1)}$, and $N_{j1}^{(2)}$. Suppose we assume these functions to be simple hat functions as classical finite element theory suggests. Summation of these basis functions over node number 4 will be equal to:

$$\sum_{b=1}^{n_{en}} N_b = \frac{1}{2} + 1 = \frac{3}{2} \qquad (3\text{-}31)$$

Here a contribution of 0.5 is coming from $N_j^{(1)}$. This is contrary to the requirement of the row-sum technique. While using CHARMS for mesh refinement, activation and deactivation of basis functions make the finite element basis non-interpolating [36]. Therefore, partition of unity is required to be enforced each time that mesh refinement takes place. The simplest way to enforce this requirement in this case, while preserving the requirement of linear independence, will be to modify the coefficient of hat functions spanning over node number 4. Therefore, as shown in figure 3-2, the summation of adjusted coefficients of hat functions spanning over node number 4 will be:

$$\sum_{b=1}^{n_{en}} N_b = \frac{1}{2} + \frac{1}{2} = 1 \qquad (3\text{-}32)$$

While using CHARMS for mesh refinement, this condition is necessary to avoid major computational complexity at a later stage.

## 3.7 Implementation of a wave propagation solver

### 3.7.1 Adaptive Algorithm

Refinement or unrefinement follows an estimate of the error using an error indicator. Our code uses residual-based estimates to compute per cell errors, which are then processed to yield per-basis-function errors. As an example of such a procedure we have used the error density, which is obtained as the ratio of the summed error from all GCELL's supporting the given basis function to their volume (area, in two dimensions). Basis functions with high error density are candidates for refinement; basis functions with low error density are considered for unrefinement.

CHARMS make it very easy to reuse the solution from the coarse mesh in the process of solving the fine mesh solution. The solution field from the coarse mesh is transferred to the fine mesh (see next section), which is then passed along to the linear equation solver as the initial guess of the fine mesh solution. Iterative solution methods then can make use of this initial guess to converge to the fine mesh solution more efficiently.

Refinement is done by activation or deactivation of basis functions using the refinement rules mentioned earlier. BFUN_SET stores active and inactive basis functions in a data structure pointer called **BUCKET**. The top of the data structure consists of active basis functions while the bottom consists of inactive basis functions, according to their IDs (known as DOFPARAM_IDs). When a function is activated, it is moved to top of the data structure and receives a new ID. Similarly, when a function is deactivated, it is moved to the bottom of the data structure and receives a new ID.

In order to control the extent of the refinement, several parameters are used. **REF_FRACTION,** or refinement fraction, controls the percentage of eligible basis functions that actually go through the refinement. **MAX_REF_LEVEL** decides the maximum refinement levels that the refinement algorithm is allowed to refine. **H_OVER_HBAR_REF** and **H_OVER_HBAR_UNREF** define the fraction of errors per basis function; above or below this error range, basis functions are eligible for refinement/unrefinement.

### 3.7.2 Field Transfer

To satisfy the needs of an adaptive procedure, ***field transfer*** from one field (the source) to another (the destination) is required. Data should be transferred from coarser mesh to the refined mesh in a lossless fashion. Therefore, depending upon the construction of the algorithm, different mesh refinement algorithms follow a unique data transfer algorithm. For example, [86] devises a field transfer operator by rendering the potential energy on a discrete basis for highly nonlinear and dynamic problems. The transfer operator devised in this fashion preserves all the internal constraints and compatibility between different state variables. Operators devised in this fashion transfer data from quadrature point to quadrature point. [16,19,20,89] present other examples of data transfer on integration points between source and target fields. Strategies based on transfer of data over integration points make more sense for algorithms based on physical bisection of elements. In our case, since we rely on refinement of shape functions instead of elements, data can be directly transferred from nodal points at the coarse level to nodal points at the finer level. The two fields are defined on two distinct refined meshes, resulting from the refinement of the same initial mesh. The field transfer operation is in general formulated as the computation of the nodal parameters of the destination field from the condition that the destination field be somehow “close” to the source field. Since our refinement algorithm is lossless, we can transfer a field from a coarse mesh to a destination field on a fine mesh exactly. However, if the destination field is defined fully or partially on a mesh coarser than the source field, the transfer will involve some loss of information.

Approximations generated by CHARMS are in general non-interpolating. This makes it seemingly difficult to come up with an efficient computation of the nodal parameters. A function given on the source field

$$\tilde{u}(x) = \sum_{k} \tilde{\phi}_k(x)\tilde{u}_k \, , \tag{3-33}$$

where $\tilde{\phi}_k \in \tilde{B}$ and $\tilde{u}_k$ are the basis functions and nodal parameters of the source field, is to be approximated by (transferred to) the destination field

$$u(x) = \sum_{j} \phi_j(x)u_j \, , \tag{3-34}$$

where $\phi_j \in B$ and $u_j$ are the basis functions and nodal parameters of the destination field. The parameters $\tilde{u}_k$ are known, but parameters $u_j$ are all unknown. Thus we have a classical function approximation problem, and a number of schemes may be used to solve for the unknown parameters; interpolation, and least square fitting being perhaps the most often used. Instead of striving for the most general solution, we take advantage of the special nature of the nested finite element spaces and use an interpolating prolongation when transferring from coarse to fine; we drop details during restriction in the opposite direction. The refinement equation (2-3), in combination with the refinement and unrefinement algorithms of section 2.1.3, yields an efficient algorithm for the computation of the nodal parameters in the destination field as described previously. Therefore using the equation 2-8 system of linear equations for evaluation at node k, which is active on level l, can be written as:

$$u_h\left(x_k^{(l)}\right) = \sum_{i \le l : \hat{B}(j) \ne 0} \left( \sum_{i : \phi_i^{(j)} \in B(i)} \left( \phi_i^{(j)} x_k^{(l)} u_i^{(j)} \right) \right) \quad (3\text{-}35)$$

This can be further simplified by using the Kronecker delta property described in 2-5:

$$u^{(l)}{}_h(x) = u_h\left(x_k^{(l)}\right) - \sum_{j : \hat{B}^{(j)} \ne 0} \left( \sum_{i : \phi_i^{(j)}(x) \in \hat{B}^{(j)}} \phi_i^{(j)} \left(x_k^{(l)}\right) u_i^{(j)} \right) \quad (3\text{-}36)$$

This shows that computation of the nodal parameter of node k on level l only requires the information of nodal parameters on levels lower than the current level. The computations of the nodal parameters of nodes on the same level are independent of each other. Therefore, in order to solve for the nodal parameters, one can start with the coarsest level, i.e. level 0, followed by sequentially higher levels in the hierarchy [34].

```
algorithm field_transfer (source field, destination field)
    1. Loop over refinement levels in destination field, and get the number of field pairs in the destination field.
    2. Loop over field pairs in the destination field.
    3. If the refinement level of the basis function corresponding to the current field pair is the same as the refinement level of the destination field, get the nodal parameter of the source field and calculate nodal parameters of the target field using equation 3-35.
    4. Go back to step number 2.
    5. Go back to step number 1.
```

### 3.7.3 Evaluation Cell (ECELL)

In our case, the mesh is needed for the evaluation of the integrals, which in turn involves the basis function derivatives. Note that there are clearly two separate and orthogonal operations here:

- Evaluate the basis functions (and their derivatives); and
- Use the computed basis function values or their derivatives to evaluate the integrals.

The second task will change with each new problem. The first task (the evaluation of the basis functions), on the other hand, will be independent of the problem to be solved. Our design therefore separates the evaluation of the basis functions into a task performed by the geometric cells, and all the other operations are factored into responsibilities for the so-called ***evaluation cells*** (ECELL's).

### 3.7.4 Protocols

The protocol PROTO_WAVE for the problem at hand consists of various functions: assemble_stiffness_matrix, assemble_mass_matrix, assemble_source_terms, and solve using time stepping as described in section 3.4. The role of the protocol is simply to bundle the functionality into a reusable component (creation of the evaluation cells, looping over those cells to assemble the contributions to the left- and right-hand sides of the linear dynamic system, and finally the solution of the system). The protocol enlists the services of a linear dynamic equation solver component.

### 3.7.5 Calculation of Basis Function Tables

When the basis function tables (the function value, and values of the derivatives with respect to the spatial coordinates) are needed at a particular quadrature point, they need to be computed for the hierarchy of overlapping basis functions. The geometric cells on which the basis functions are defined are related through the parent/child relationship.

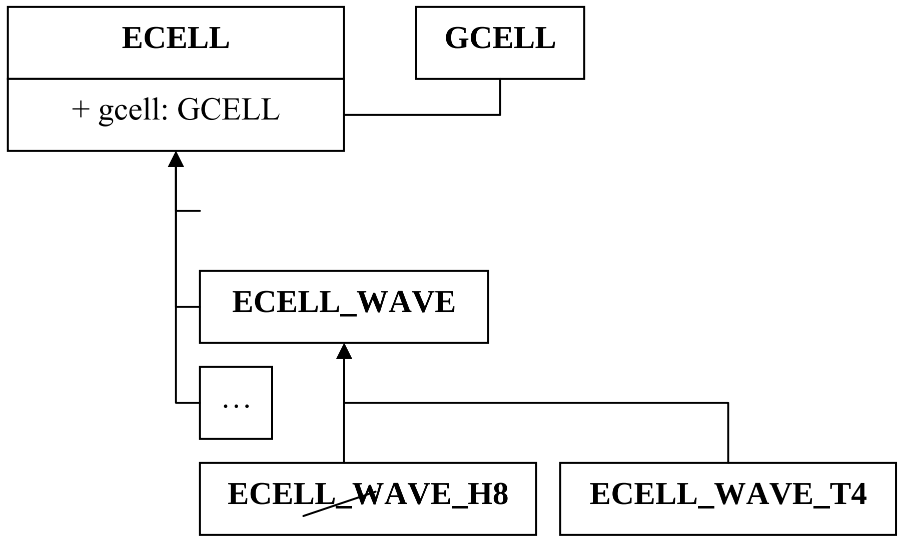


**Figure 3-3: Class diagram for wave propagation evaluation cell for 8-node hexahedron.**

To evaluate the spatial integrals, perform numerical quadrature on the geometric cells that are the leaves of the refinement hierarchy. In other words, the interaction of basis functions from different levels is evaluated over geometric cells that support the finest basis function. An alternative approach that evaluates interactions of pairs of basis functions over the support of the finer function had been proposed in [36].

Because of our chosen quadrature scheme (all integrals are evaluated over the leaf cells), we need to traverse the hierarchy from the leaf towards the root. Furthermore, since all the parent/child relationships are transitive, it is sufficient to consider in this discussion just two levels from the mesh hierarchy: the parent level and the child level.

Without loss of generality we may consider a particular element type, for instance the isoparametric eight-node (serendipity) quadrilateral, Figure 2-2, [72]. The basis functions of a single element are defined in the parametric coordinates in the biunit square $\left(-1 \le \xi \le +1, -1 \le \eta \le +1\right)$ as

$$\Phi^T(\xi,\eta) = \begin{bmatrix} -1/4(1-\xi)(1-\eta)(1+\xi+\eta) \\ -1/4(1+\xi)(1-\eta)(1-\xi+\eta) \\ -1/4(1+\xi)(1+\eta)(1-\xi-\eta) \\ -1/4(1-\xi)(1+\eta)(1+\xi-\eta) \\ 1/2(1-\xi)(1+\xi)(1-\eta) \\ 1/2(1-\eta)(1+\eta)(1+\xi) \\ 1/2(1-\xi)(1+\xi)(1+\eta) \\ 1/2(1-\eta)(1+\eta)(1-\xi) \end{bmatrix}, \qquad (3\text{-}37)$$

where the superscript $T$ indicates transpose. The derivatives of the basis functions with respect to the spatial coordinates are evaluated as usual by the chain rule, which can be put in matrix form as

$$\begin{bmatrix} \dfrac{\partial \Phi(\xi,\eta)}{\partial x} \\ \dfrac{\partial \Phi(\xi,\eta)}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \dfrac{\partial \Phi(\xi,\eta)}{\partial \xi} \\ \dfrac{\partial \Phi(\xi,\eta)}{\partial \eta} \end{bmatrix}, \qquad (3\text{-}38)$$

with the Jacobian matrix defined as

$$\mathbf{J} \quad = \quad \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}. \qquad (3\text{-}39)$$

The elements of the Jacobian matrix are computed from the expansions of *x* and *y* in terms of the basis functions

$$x = \sum_i \Phi_i(\xi,\eta) x_i, \quad y = \sum_i \Phi_i(\xi,\eta) y_i\,, \qquad (3\text{-}40)$$

where $x_i$ and $y_i$ are the nodal geometry parameters.

If we now consider approximation on a mesh with basis functions at several levels, we may still keep equation (3-39), but the elements of the $\Phi$ matrix are now functions $\phi_i^{(j)}$ at various levels *j*. Similarly, the derivatives with respect to spatial derivatives may be computed from equation (3-40), but it must be realized that the geometric cells in the mesh hierarchy have their own parametric coordinates (i.e. they are all mapped from the master element). To compute the Jacobian matrix, equation (3-6), all the derivatives $\partial \Phi_i / \partial \xi$, etc. need to be computed in a *single* coordinate system. In our implementation, we choose to compute all the derivatives in the parametric coordinates of the leaf geometric cell (i.e. the finest cell in the hierarchy). We use the notation $\xi'$ and $\eta'$ for the parametric coordinates of the parent, and $\xi$ and $\eta$ for the parametric coordinates of the child, and $\phi_i^{(j)}$ for the basis function defined on the parent, and $\phi_i^{(j+1)}$ for the basis functions defined on the child. The parent (top row in Figure 2-2) is divided into four children (bottom row) in its parametric space, and all elements, parent and the children, are mapped to the physical space using the standard isoparametric mapping in their own coordinates. We express the derivatives of the basis functions on the parent cell with respect to the parametric coordinates of the child needed in equation (3-39) through the chain rule

$$\frac{\partial \phi_i^{(j)}}{\partial \xi} = \frac{\partial \phi_i^{(j)}}{\partial \xi'} \frac{\partial \xi'}{\partial \xi}, \tag{3-41}$$

and similarly for the relationship between $\eta$ and $\eta'$. For our particular example, the mapping from the parent to the child is simple. For instance, for child 0 (compare with Figure 2-2):

$$\begin{aligned} \xi' &= 1/2(\xi - 1) \\ \eta' &= 1/2(\eta - 1), \end{aligned} \tag{3-42}$$

and we get $\partial \xi' / \partial \xi = 1/2$ and $\partial \eta' / \partial \eta = 1/2$; it is easy to show that this relationship holds for all four children.

We are ready to state the algorithm eval_bfun_set for the computation of the values and derivatives with respect to the spatial coordinates of the basis functions active at a particular quadrature point: The inputs are the leaf cell and the parametric coordinates of a given quadrature point in that cell. Note that the tree needs to descend from the leaf cell towards the root, and the algorithm may proceed by recursion or by iteration.

**algorithm** eval_bfun_set

```
1. Descend the refinement tree and find $\phi_i^{(j)}$, $\partial\phi_i^{(j)}/\partial\xi$,… for all active
        functions, $j = 0,1,…$from (Eq. 3-40).
2. Compute the Jacobian matrix $\mathbf{J}$ of Equation (3-39), and then invert
        to obtain $\mathbf{J}^{-1}$
3. Using $\partial\phi_i^{(j)}/\partial\xi$,. . . from step 1, and $\mathbf{J}$ from step 2, compute
        $\partial\phi_i^{(j)}/\partial x$, $j = 0,1,…$, from Equation (3-38)
```

Note that the task of evaluating the individual basis functions and their derivatives in the parametric coordinates is performed by the GCELL's, as usual in standard finite element codes. The only difference is that the child's coordinates need to be related to the parent's coordinates through the chain rule (the factor 1/2 in the example discussed above).

### 3.7.6 Geometry

A remark is in order concerning the definition of the geometry of the domain. An isoparametric description of the finite element fields is used, and Equation (3-40) makes it clear that in order to evaluate the geometry, i.e. the location of a point given by its natural, parametric coordinates, the nodal parameters for the geometry are needed. Our adaptive framework represents the geometry as an ordinary field. In other words, the code makes no distinction between the unknown fields and the geometry. However, one difference needs to be noted: the geometry needs to be initialized at the beginning of the computation. Fortunately, that is an easy task, since the basis functions on the initial (input) mesh interpolate. Therefore, the nodal parameters of the initial geometry field are simply the coordinates of the nodes. However, the geometry fields for the refined approximations need to be computed by field transfer, since the approximation becomes non-interpolating upon refinement. If the initial mesh describes the domain exactly, the geometry is also described exactly on any refined mesh, because of the lossless character of field transfers from coarse to fine meshes. On the other hand, if the geometry of the domain is only approximated by the mesh, the nodal geometry parameters may be computed to improve the shape approximation by fitting curved boundaries [34].

## 3.8 Conservation Laws

In Newtonian mechanics success of a time-stepping scheme very often is closely related to its capacity to preserve mass, momentum, and energy. Most of the iterative schemes for time-dependent problems either preserve these quantities exactly, as is the case of many implicit schemes, or these quantities are bounded. Therefore, much research in the past went towards understanding the conservation behavior of popular schemes such as Newmark's algorithm and Wilson's theta method [81,87]. In our case, we are introducing adaptive mesh refinement within the framework of the explicit Newmark's algorithm. Conservation behavior for such an overall scheme is demonstrated with the help of first a simple example and then with a numerical experiment. A generalized proof is highly desirable, but left for future research in this area.

### 3.8.1 Conservation of Mass and Mass Matrix during Refinement

The development of the mass matrix dates back to work done by Duncan and Collar [90,91], where they showed a 3x3 diagonal "inertia matrix" for a triple pendulum. Mass matrices can be broadly classified as lumped, consistent, and template (lumped-consistent) mass matrices. In the early '60s, the lumped mass matrix was developed not just for simplicity and overall computational cost effectiveness, but because of its diagonal nature, it was the obvious choice to account for nonstructural masses. A lumped mass matrix can be constructed in various ways. Nodal quadrature, row-sum technique, and HRZ lumping are some of the most popular techniques.

While nodal quadrature has the tendency to generate negative and zero masses, the row-sum technique does not produce zero masses. However, the row-sum technique can generate negative masses in some cases, such as in the corner node for the eight node serendipity element. HRZ lumping overcomes both of these problems; however, more research work is required to prove its correctness for general elements. Archer [92,93] first pointed towards the correctness of consistent mass, taking clues from the Lagrange dynamics equation. Melosh [94] established the connection between the Rayleigh-Ritz method and FEM and showed uses of the mass matrix in similar analysis. Melosh's work brought attention towards some of the problems with the consistent mass matrix: namely, (1) it was prohibitively expensive and inefficient for some solution processes. For example, in the case of explicit dynamics, accelerations are computed at the global level by multiplying the inverse of the mass matrix with effective dynamic force. This leads to a trivial solution of system of equations. If a non-diagonal consistent mass matrix is used, the solution of the system of equations will be nontrivial; (2) Non-structural masses were not automatically accounted for. In fact, in many applications such as the analysis of aircraft and ships, structural masses account for only 20-30% of total mass; (3) Other alternatives might provide better results. If a stiffness matrix results from conforming displacement interpolation, pairing it with a consistent mass matrix guarantees providing upper bounds on natural frequency. This may or may not be a good thing. In practice, it is observed that errors increase rapidly as one moves up the frequency spectrum. For wave propagation problems, where response is strongly driven by intermediate and high frequencies, the consistent mass matrix may give very poor results. There are several qualities a mass matrix should have in order to be successfully used in engineering problems:

(1) It should be symmetric, and element symmetries should be reflected in the mass matrix. Although this is not a necessary requirement as seen in the Petrov-Galerkin procedure, computationally an unsymmetric mass matrix will be very expensive and practically impossible to use in large engineering simulations.

(2) Another important property of the mass matrix is that in semi-discrete equations (ordinary differential equations), it complies with all conservation laws. One consequence of this is that the sum of the elements of the mass matrix gives the total mass.

(3) Usually if the mass matrix is at least positive semi-definite, the numerical problem becomes a lot easier to solve; i.e., for any non-zero velocity field v,

$\{v\}^T [M] \{v\} \geq 0$. A positive definite mass matrix will lead to positive eigenvalues. A negative eigenvalue means that the corresponding natural frequency of the structure is complex (not real). A mass matrix is usually positive definite or positive semi-definite, but in some applications, e.g., in buckling, the mass matrix is only nonnegative, or even indefinite.

In this section, we are going to study the construction of mass matrices using various techniques. The accuracy of eigenvalues and eigenfunctions are measures of the quality of both the stiffness and mass matrices. Although we already pointed out that one of our targeted applications is wave propagation problems and that a consistent mass matrix is not appropriate for wave propagation problems where response of medium and high frequencies is desired, we are going to nevertheless include a consistent mass matrix in our analysis. In our convergence analysis, we will use a uniformly refined mesh as a benchmark to validate CHARMS results.

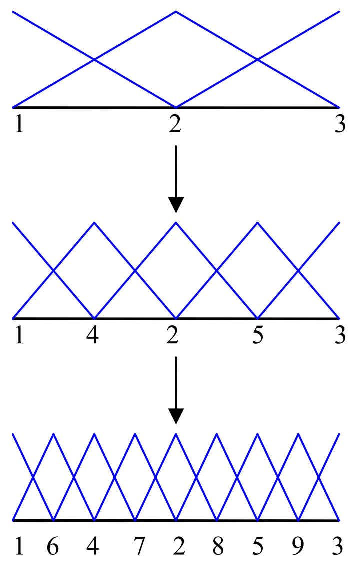


**Figure 3-4 Uniformly Refined Mesh**

### 3.8.1.1 Uniformly Refined Mesh

#### 3.8.1.1.1 Using Lumped Mass (Row-sum technique)

Consider the initial mesh: Shape functions for the coarse mesh can be written as:

**Table 3-1: Linear hat functions and their derivatives of coarse mesh.**

| Range | Title | Shape Function | $\frac{\partial N}{\partial x}$ |
|---|---|---|---|
| $0 < x < l$ | $N_1$ | $1-\frac{x}{l}$ | $-\frac{1}{l}$ |
| $0 < x < l$<br><br>$l < x < 2l$ | $N_2$ | $\frac{x}{l}$<br><br>$2\left(1-\frac{x}{2l}\right)$ | $\frac{1}{l}$<br><br>$-\frac{1}{l}$ |
| $l < x < 2l$ | $N_3$ | $\frac{x}{l}-1$ | $\frac{1}{l}$ |

The lumped mass matrix can be derived from the formulation shown in section 3.6. The mass matrix for elements 12 and 23 on the initial mesh can be written as:

$$m_{12} = \frac{\rho l}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad m_{23} = \frac{\rho l}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Therefore, the global lumped mass matrix for the initial mesh will be:

$$M = \frac{\rho l}{2}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Similarly, the stiffness matrix can be assembled from the pieces of the finite element (considering unit cross-sectional area):

$$K = \sum_{j=1}^{n_{el}} \int_0^l [B]^T E[B] dx$$

$$k_{12} = k_{23} = \frac{E}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$K = \frac{E}{l}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

Therefore eigenvalues can be written as $\lambda = [0,2,4]$. Here 0 frequency arises from the singular stiffness matrix due to the presence of a rigid body mode.

Now, uniformly refine the mesh by subdividing all the elements,

$$K = \frac{E}{L}\begin{bmatrix} 2 & -2 & 0 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 \\ 0 & -2 & 4 & -2 & 0 \\ 0 & 0 & -2 & 4 & -2 \\ 0 & 0 & 0 & -2 & 2 \end{bmatrix}, \text{ and } M = \rho l \begin{bmatrix} 1/4 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/4 \end{bmatrix}$$

Eigenvalues can be written as $\lambda = [0, 2.3431, 8, 13.66, 16]$ $\lambda = [0, 2, 4]$

Refined once again:

$$K = \frac{E}{L}\begin{bmatrix} 4 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & 8 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & 8 & -4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & 8 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 8 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 8 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & 8 & -4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -4 & 8 & -4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 4 \end{bmatrix}$$

Diagonal components of M are $M = [1/8, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/8]$

Eigenvalues can be written as:

$$\lambda = [0 \quad 2.4 \quad 9.3 \quad 19.75 \quad 32 \quad 44 \quad 54 \quad 61 \quad 64]$$

### 3.8.1.1.2 Using Consistent Mass

The consistent mass matrix can be derived by taking the *kinetic energy* as part of the governing functional. The kinetic energy of an element of mass density $\rho$ that occupies the domain $\Omega^e$ and moves with velocity field $\vec{v}^e$ is

$$T^e = 0.5 \int_{\Omega^e} \rho (\vec{v}^e)^T (\vec{v}^e) d\Omega$$

Following the FEM philosophy, the element velocity field is interpolated by shape functions: $\vec{v}^e = N\dot{u}^e$, where $\dot{u}^e$ are nodal velocities, and N is a shape function matrix. Substituting this into the previous equation,

$$T^e = 0.5(\dot{u}^e)^T \int_{\Omega^e} \rho(N)^T (N) d\Omega \dot{u}^e = 0.5(\dot{u}^e)^T M^e \dot{u}^e$$

The element mass matrix follows as the Hessian of $T^e$:

$$M^e = \frac{\partial^2 T^e}{\partial v \partial v} = \int_{\Omega^e} \rho N^T N d\Omega$$

Therefore the consistent mass matrix for element 12 can be written:

$$m_{12} = \rho \int_0^L \begin{bmatrix} 1-\frac{x}{L} \\ \frac{x}{l} \end{bmatrix} \begin{bmatrix} 1-\frac{x}{L} & \frac{x}{l} \end{bmatrix} dx = \frac{\rho l}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

The overall consistent mass matrix can be written as:

$$M = \frac{\rho l}{6} \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

The stiffness matrix, as defined in the previous section, will remain unchanged. Therefore, eigenvalues can be written, $\lambda = [0,2,4]$, which are exactly the same values derived from the coarse mesh using a lumped mass. Upon refining once, the consistent mass matrix can be written as:

$$\mathrm{M} = \begin{bmatrix} 1/6 & 1/12 & 0 & 0 & 0 \\ 1/12 & 1/3 & 1/12 & 0 & 0 \\ 0 & 1/12 & 1/3 & 1/12 & 0 \\ 0 & 0 & 1/12 & 1/3 & 1/12 \\ 0 & 0 & 0 & 1/12 & 1/6 \end{bmatrix}, \text{ and } \lambda = [0,2.6,12,31.7,48]$$

Upon refining twice eigenvalues can be written:

$$\lambda = [0 \quad 2.45 \quad 9.21 \quad 19.45 \quad 32 \quad 44.1 \quad 53.5 \quad 60.8 \quad 66]$$

### 3.8.1.2 CHARMS Refined Mesh

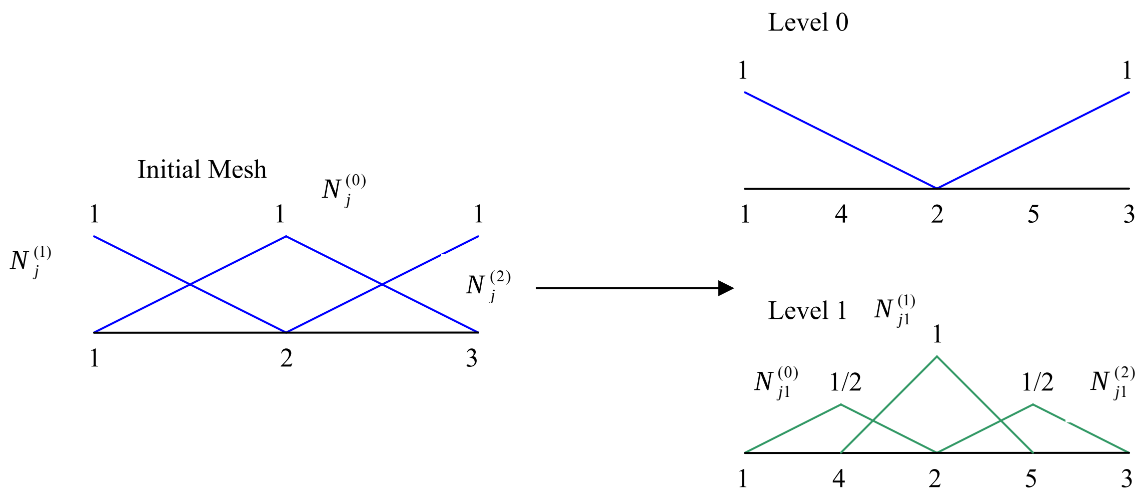


**Figure 3-5: Adaptive mesh refinement of shape function $N_j^{(0)}$ using CHARMS**

Shape function and their derivatives are as follows:

On coarse mesh, shape functions will be the same as the previous section.

On the finer mesh, shape functions can be written:

**Table 3-2: Linear hat functions and their derivatives of mesh refined using CHARMS.**

| **Range** | **Title** | **Shape Function** | $\frac{\partial N}{\partial x}$ |
|---|---|---|---|
| $0 < x < l$ | $N_1$ | $1-\frac{x}{l}$ | $-\frac{1}{l}$ |
| $l/2 < x < l$ <br> $l < x < 3l/2$ | $N_2$ | $\frac{2x}{l}-1$ <br> $\left(3-\frac{2x}{l}\right)$ | $\frac{2}{l}$ <br> $-\frac{2}{l}$ |
| $l < x < 2l$ | $N_3$ | $\frac{x}{l}-1$ | $\frac{1}{l}$ |
| $0 < x < l/2$ <br> $l/2 < x < l$ | $N_4$ | $\frac{x}{l}$ <br> $1-\frac{x}{l}$ | $\frac{1}{l}$ <br> $-\frac{1}{l}$ |
| $l < x < 3l/2$ <br> $3l/2 < x < 2l$ | $N_5$ | $\left(\frac{2x}{l}-2\right)$ <br> $\left(-\frac{2x}{l}+4\right)$ | $\frac{2}{l}$ <br> $\frac{-2}{l}$ |

Unlike classical finite element practices, in a CHARMS refined mesh multiple shape functions might be spanning over a given support, as discussed earlier. Therefore, one should carefully integrate by parts while calculating pieces of the stiffness and mass matrices. For example, various components of the consistent mass matrix can be written as:

$$\int_0^l N_1^2 = \int_0^l \left(1-\frac{x}{l}\right)^2 dx = l/3 \qquad \int_0^l N_4^2 = \int_0^{l/2} \left(\frac{x}{l}\right)^2 dx + \int_{l/2}^l \left(1-\frac{x}{l}\right)^2 dx = l/12$$

$$\int_{l/2}^{3l/2} N_2^2 = \int_{l/2}^l \left(\frac{2x}{l}-1\right)^2 dx + \int_l^{3l/2} \left(3-\frac{2x}{l}\right)^2 dx = \frac{l}{3}$$

$$\int_0^l N_4 N_1 = \int_0^{l/2} \left(1-\frac{x}{l}\right)\frac{x}{l} dx + \int_{l/2}^l \left(1-\frac{x}{l}\right)^2 dx = l/8$$

$$\int_{l/2}^l N_4 N_2 = \int_{l/2}^l \left(1-\frac{x}{l}\right)\left(\frac{2x}{l}-1\right) dx = l/24 \quad \int_{l/2}^l N_1 N_2 = \int_{l/2}^l \left(1-\frac{x}{l}\right)\left(\frac{2x}{l}-1\right) dx = l/24$$

The rest of the values can be obtained using symmetry. Different components of the stiffness matrix can be calculated similarly:

$$\int_0^l \frac{\partial N_1}{\partial x}\frac{\partial N_1}{\partial x} = \int_0^l \frac{1}{l^2} dx = \frac{1}{l} \qquad \int_0^l \frac{\partial N_1}{\partial x}\frac{\partial N_4}{\partial x} = \int_0^{l/2} -\frac{1}{l^2} dx + \int_{l/2}^l \frac{1}{l^2} dx = 0$$

$$\int_{l/2}^l \frac{\partial N_1}{\partial x}\frac{\partial N_2}{\partial x} = \int_{l/2}^l -\frac{2}{l^2} dx = \frac{-1}{l} \qquad \int_{l/2}^{3l/2} \frac{\partial N_2}{\partial x}\frac{\partial N_2}{\partial x} = \int_{l/2}^l \frac{4}{l^2} dx + \int_l^{3l/2} \frac{4}{l^2} dx = \frac{4}{l}$$

$$\int_0^l \frac{\partial N_4}{\partial x}\frac{\partial N_4}{\partial x} = \int_0^l \frac{1}{l^2} dx + \int_{l/2}^l \frac{1}{l^2} dx = \frac{1}{l} \quad \int_0^l \frac{\partial N_1}{\partial x}\frac{\partial N_2}{\partial x} = \int_{l/2}^l -\frac{2}{l^2} dx = \frac{-1}{l}$$

### 3.8.1.2.1 Row sum technique

The global stiffness and lumped mass matrices can now be assembled from the element stiffness and mass matrices, as shown in previous section. The global lumped mass, using the row-sum technique and the stiffness matrix can be written:

$$M = \frac{\rho l}{2}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathrm{K} = \frac{E}{L}\begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

Eigenvalues for the structure using the above stiffness and mass matrices can be written as $\lambda = [0, 2, 2.9, 4, 11]$. It is easy to see that the lumped mass matrix using the row-sum technique conserves total mass, $M = \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{2} + \frac{1}{4} + \frac{1}{2}\right)\rho l = 2\rho l$, and it is positive definite.

### 3.8.1.2.2 Consistent Mass

The global consistent mass matrix for mesh refined using CHARMS can be written:

$$M = \rho l \begin{bmatrix} 1/3 & 1/8 & 1/24 & 0 & 0 \\ 1/8 & 1/12 & 1/24 & 0 & 0 \\ 1/24 & 1/24 & 1/3 & 1/24 & 1/24 \\ 0 & 0 & 1/24 & 1/12 & 1/8 \\ 0 & 0 & 1/24 & 1/8 & 1/3 \end{bmatrix}$$

It is easy to see that the consistent mass matrix formed using CHARMS conserves total mass: $M = \left(2\left(\frac{4}{24}+\frac{2}{8}\right)+\frac{3}{3}+\frac{2}{12}\right)\rho l = 2\rho l$. Also, it can be verified that the consistent mass matrix is positive definite since eigenvalues of the consistent mass matrix are positive: $\lambda = [0.028, .0316, .2807, 0.3851, 0.4413]$. Using the stiffness matrix from previous calculations, eigenvalues of the structure are $\lambda = [0, 2.3, 8, 13, 66]$.

### 3.8.1.2.3 HRZ Mass Lumping

Unlike the row-sum technique, HRZ lumping developed by Hinton, Rock and Zinkiwicz [95] has been popular because it always produces positive lumped masses. In this technique, first diagonal terms of the consistent mass matrix are computed, then they are scaled so as to preserve total mass.

$$m_{pq}^{e} = \begin{Bmatrix} \alpha\delta_{ij} \int\limits_{\Omega_e} \rho N_a^2 d\Omega, & a = b \\ 0 & a \neq b \end{Bmatrix}$$

$$\alpha = \frac{\int\limits_{\Omega_e} \rho d\Omega}{\left( \sum\limits_{a=1}^{n_{en}} \int\limits_{\Omega_e} \rho N_a^2 d\Omega \right)}$$

Using the consistent mass matrix developed in the previous section, HRZ lumping for CHARMS refined mesh can be written:

$$M = \frac{\rho l}{7} \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix}$$

Using HRZ lumping, eigenvalues of the structure are $\lambda = [0, 1.75, 3.5, 7, 12.25]$. It is also obvious that the lumped mass matrix formed this way conserves total mass, $M = \left(\frac{4}{14} + \frac{1}{14} + \frac{4}{14} + \frac{1}{14} + \frac{4}{14}\right)\rho l = 2\rho l$, and it is positive definite for any given problem.

### 3.8.1.2.4 Template Mass Lumping

Template mass lumping is obtained by combining the lumped and consistent mass matrices:

$$M_{template} = (1-\beta)M_{lumped} + \beta M_{consistent}$$

Although it is observed that one might achieve superior accuracy by customizing a mass matrix in this fashion [96], and dispersion of low and intermediate frequencies can be controlled efficiently ($\beta$=1/2 will give optimal dispersion for low frequencies) [90], the derivation is computationally very expensive, even for a one dimensional system such as this. Therefore its use is impractical in any given algorithm.

### 3.8.1.3 Conclusion of Calculations

Eigenvalues achieved from each mass matrix are summarized as follows:

**Table 3-3: Comparison of eigenvalues**

| Uniform Refined Mesh | | | | | | | CHARMS Refined mesh | | |
|---|---|---|---|---|---|---|---|---|---|
| Unrefined | | Refined Once | | Refined Twice | | Mode Number | AMR | AMR | AMR |
| Lumped Masss | Consistent Mass | Lumped Mass | Consistent Mass | Lumped Mass | Consistent Mass | | Lumped Mass | Consistent Mass | HRZ Lumping |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2 | 2 | 2.34 | 2.6 | 2.4 | 2.5 | 2 | 2 | 2.3 | 1.75 |
| 4 | 4 | 8 | 12 | 9.37 | 10.38 | 3 | 2.9 | 8 | 3.5 |
| | | 13.66 | 31.7 | 19.75 | 24.87 | 4 | 4 | 13 | 7 |
| | | 16 | 48 | 32 | 48 | 5 | 11 | 16 | 12.25 |

The following conclusions can be derived from the above calculations:

1. Different types of mass matrices studied in this section are symmetric, produce positive definite mass matrices, and they conserve total mass with mesh refined using CHARMS.
2. Although, the consistent mass matrix shows the fastest convergence in eigenvalues, it is computationally very expensive in comparison to row-sum and HRZ mass lumping.

### 3.8.2 Conservation of Momentum

In order to arrive at a more conclusive point with our adaptive explicit algorithm and make more intelligent choices, it is very important to understand if and how it is modifying the underlying properties of the existing explicit algorithm. In this section, we will first notice the momentum conservation behavior of the explicit Newmark algorithm, and then we are going to study its behavior after introduction of refinement using the same one-dimensional example as in the previous section.

#### 3.8.2.1 Momentum Conservation with Central Difference Method

The key equations of the Newmark method are as follows:

$$\mathbf{d}_{n+1} = \mathbf{d}_n + \Delta t\mathbf{v}_n + 0.5\Delta t^2\left[(1-2\beta)\mathbf{a}_n + 2\beta\mathbf{a}_{n+1}\right] \qquad (3\text{-}43)$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t\left[(1-\gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+1}\right]$$

Using $\beta = 0$ and $\gamma = 0.5$ for the central difference method, these equations may be rewritten as follows:

$$\mathbf{d}_{n+1} = \mathbf{d}_n + \Delta t\mathbf{v}_n + 0.5\Delta t^2\mathbf{a}_n \qquad (3\text{-}44)$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + 0.5\Delta t(\mathbf{a}_n + \mathbf{a}_{n+1})\,. \qquad (3\text{-}45)$$

Upon equating forces,

$$\mathbf{M}\mathbf{a}_n = \mathbf{K}\mathbf{d}_n$$

$$\mathbf{a}_n = \mathbf{M}^{-1}\mathbf{K}\mathbf{d}_n\,.$$

Substituting into equation (3-45), we get

$$\mathbf{v}_{n+1} = \mathbf{v}_n + 0.5\Delta t\mathbf{M}^{-1}\mathbf{K}(\mathbf{d}_n + \mathbf{d}_{n+1})\,. \qquad (3\text{-}46)$$

Consider the scalar product where $\eta$ is a rigid body mode:

$$\eta \cdot \mathbf{M} \cdot \mathbf{v}_n = [\eta_1^t, \eta_2^t, \eta_3^t, \eta_4^t, \ldots \eta_N^t] \cdot \mathbf{M} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}_n = \sum_i \sum_j \eta_i M_{ij} v_j \qquad (3\text{-}47)$$

By multiplying the above scalar product on both sides of equation (3-46),

$$\eta \mathbf{M} \mathbf{v}_{n+1} = \eta \mathbf{M} \mathbf{v}_n + 0.5 \Delta t \eta \mathbf{K} (\mathbf{d}_n + \mathbf{d}_{n+1}) \qquad (3\text{-}48)$$

If there are no restraints, **K** will be singular because of presence of rigid body modes in the problem. Therefore, $(\eta, \mathbf{K})^T = \mathbf{K} \cdot \eta = 0$, since rigid body motion produces no force. Hence from equation (3-48),

$$\eta \cdot \mathbf{M} v_{(n+1)} = \eta \cdot \mathbf{M} v_n \qquad (3\text{-}49)$$

Therefore momentum will stay conserved. As a matter of fact, if all the rigid body modes are not allowed, momentum cannot be conserved by any time stepping algorithm exactly.

### 3.8.2.2 Momentum Conservation Property with Adaptive Mesh Refinement

Consider an arbitrary time interval during the solution of the equation of motion using the explicit Newmark method as explained in section 3.4. Suppose refinement takes place between the time steps n and n+1. At the end of step n, the error is calculated. Mesh is refined and the solution starts from a few time steps behind. Conservation of momentum while going from step n to n+1 using the same mesh has already been shown in previous section. Now let us look at the momentum conservation before and after adaptive mesh refinement.

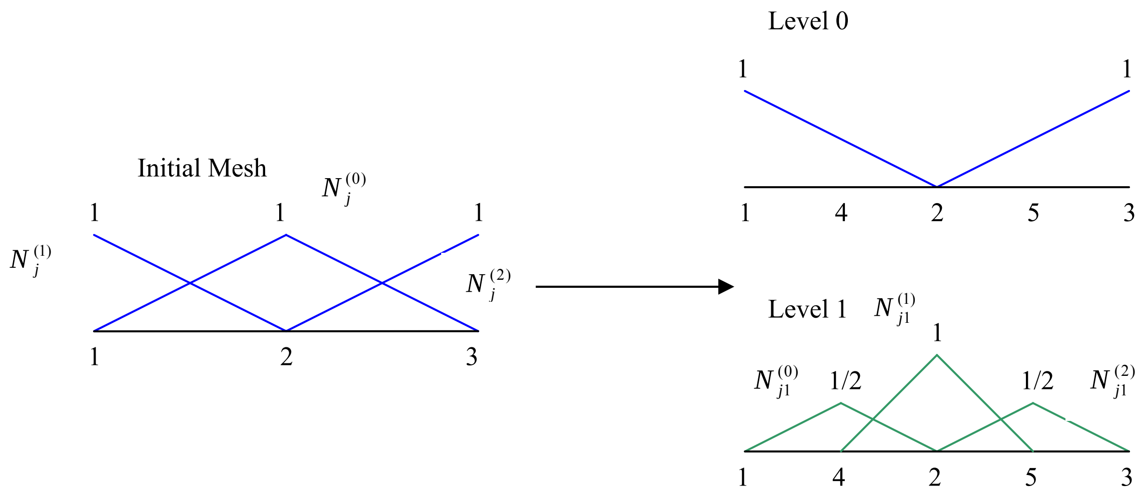


**Figure 3-6: Adaptive mesh refinement of shape function $N_j^{(0)}$ using CHARMS**

In order to compare momentum before and after refinement, we will have to calculate the velocity field at the refined mesh, given its values on the unrefined mesh. This can done using a field transfer procedure, as described previously:

$$u_h\left(x_k^{(l)}\right)=\sum_{i\le l:\hat{B}(j)\neq 0}\left(\sum_{i:\phi_i^{(j)}\in B(i)}\left(\phi_i^{(j)}x_k^{(l)}u_i^{(j)}\right)\right) \qquad (3\text{-}16)$$

Suppose the velocity field on the original mesh was: $v_1, v_2, v_3$

and the velocity field on the refined mesh was: $v_1', v_2', v_3', v_4', v_5'$

From equation (3-16):

$$\begin{aligned} v_4' &= 0.5v_1 + 0.5v_2 \\ v_5' &= 0.5v_2 + 0.5v_3 \\ v_1' &= v_1 \\ v_2' &= v_2 \\ v_3' &= v_3 \end{aligned}$$

Let *P* be momentum. Momentum of the different elements on the initial mesh will be:

$$P_{12} = m_{12}v_{12} = \frac{\rho l}{2}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}v_1\\v_2\end{bmatrix} = \begin{bmatrix}\rho l v_1/2\\ \rho l v_2/2\end{bmatrix} \qquad P_{23} = m_{23}v_{23} = \frac{\rho l}{2}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}v_2\\v_3\end{bmatrix} = \begin{bmatrix}\rho l v_2/2\\ \rho l v_3/2\end{bmatrix}$$

Therefore, momentum for the initial mesh:

$$P = \begin{bmatrix}\rho l v_1/2\\ \rho l v_2\\ \rho l v_3/2\end{bmatrix}$$

If velocity was uniform in magnitude and direction $\vec{v}_1 = \vec{v}_2 = \vec{v}_3 = \vec{v}$, then overall momentum of the structure using the initial mesh can be written:

$$\rho l v_1/2 + \rho l v_2 + \rho l v_3/2 = 2\rho l v\,.$$

### 3.8.2.2.1 Using Row-Sum Technique

Now let us consider the refined mesh. Momentum on the refined mesh can be calculated:

$$P_{12} = m_{12}v_{12} = \frac{\rho l}{2}\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}\begin{bmatrix}v_1\\v_2\end{bmatrix} = \begin{bmatrix}\rho l v_1/2\\ 0\end{bmatrix}$$

$$P_{23} = m_{23}v_{23} = \frac{\rho l}{2}\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}v_2\\v_3\end{bmatrix} = \begin{bmatrix}0\\ \rho l v_3/2\end{bmatrix}$$

$$P_{42} = m_{42}v_{42} = \frac{\rho l}{4}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}(v_1+v_2)/2\\ v_2\end{bmatrix} = \begin{bmatrix}\rho l(v_1+v_2)/8\\ \rho l v_2/4\end{bmatrix}$$

$$P_{25} = m_{25}v_{25} = \frac{\rho l}{4}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}v_2\\ (v_2+v_3)/2\end{bmatrix} = \begin{bmatrix}\rho l v_2/4\\ \rho l(v_2+v_3)/8\end{bmatrix}$$

Therefore, overall momentum for the refined mesh will be:

$$P = \begin{bmatrix} \rho l v_1 / 2 \\ \dfrac{\rho l (v_1 + v_2)}{8} \\ \dfrac{\rho l v_2}{4} + \dfrac{\rho l v_2}{4} \\ \dfrac{\rho l (v_2 + v_3)}{8} \\ \rho l v_3 / 2 \end{bmatrix}$$

Again, if velocity was uniform, $\vec{v}_1 = \vec{v}_2 = \vec{v}_3 = \vec{v}$, momentum of the structure on the initial mesh can be written as:

$$\rho l v / 2 + \frac{\rho l v}{4} + \rho l v / 2 + \frac{\rho l v}{4} + \rho l v / 2 = 2 \rho l v$$

As mentioned above, conservation of momentum upon application of uniform velocity also indicates the overall mass conservation property of the lumped mass matrix during refinement.

#### 3.8.2.2.2 Using Consistent Mass Matrix

The consistent mass matrix is calculated in previous section. Momentum using consistent mass matrix can be written:

$$P = \rho l \begin{bmatrix} 1/3 & 1/8 & 1/24 & 0 & 0 \\ 1/8 & 1/12 & 1/24 & 0 & 0 \\ 1/24 & 1/24 & 1/3 & 1/24 & 1/24 \\ 0 & 0 & 1/24 & 1/12 & 1/8 \\ 0 & 0 & 1/24 & 1/8 & 1/3 \end{bmatrix} \begin{bmatrix} v_1 \\ (v_2 + v_1)/2 \\ v_2 \\ (v_2 + v_3)/2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \dfrac{19 \rho l v_1}{48} + \dfrac{5 \rho l v_2}{48} \\ \dfrac{\rho l v_1}{6} + \dfrac{\rho l v_2}{12} \\ \dfrac{\rho l v_1}{16} + \dfrac{3 \rho l v_2}{8} + \dfrac{\rho l v_3}{16} \\ \dfrac{\rho l v_3}{6} + \dfrac{\rho l v_2}{12} \\ \dfrac{19 \rho l v_3}{48} + \dfrac{5 \rho l v_2}{48} \end{bmatrix}$$

If velocities are uniform, $\vec{v}_1 = \vec{v}_2 = \vec{v}_3 = \vec{v}$, momentum of the structure on the refined mesh can be written:

$$2\rho l\left(\frac{19v}{48}+\frac{5v}{48}+\frac{v}{6}+\frac{v}{12}\right)+\frac{\rho lv}{16}+\frac{3\rho lv}{8}+\frac{\rho lv}{16}=2\rho lv$$

Therefore momentum remains conserved using the consistent mass matrix upon application of uniform velocity.

### 3.8.2.2.3 Using HRZ Lumping

The HRZ lumped mass matrix is calculated in the previous section.

Momentum using HRZ lumped mass matrix can be written:

$$P=\frac{\rho l}{7}\begin{bmatrix}4&0&0&0&0\\0&1&0&0&0\\0&0&4&0&0\\0&0&0&1&0\\0&0&0&0&4\end{bmatrix}\begin{bmatrix}v_1\\(v_2+v_1)/2\\v_2\\(v_2+v_3)/2\\v_3\end{bmatrix}=\begin{bmatrix}\dfrac{4\rho lv_1}{7}\\\dfrac{\rho lv_1}{14}+\dfrac{\rho lv_2}{14}\\\dfrac{4\rho lv_2}{7}\\\dfrac{\rho lv_3}{14}+\dfrac{\rho lv_2}{14}\\\dfrac{4\rho lv_3}{7}\end{bmatrix}$$

If velocities are uniform, $\vec{v}_1 = \vec{v}_2 = \vec{v}_3 = \vec{v}$, momentum of the structure on the refined mesh can be written as:

$$2\rho l\left(\frac{4v}{7}+\frac{v}{14}+\frac{v}{14}\right)+\frac{4\rho lv}{7}=2\rho lv$$

Therefore momentum remains conserved using the HRZ lumped mass matrix upon application of uniform velocity.

### 3.8.3 Conservation of Energy

The energy conservation property is another very important aspect of numerical algorithms for parabolic-hyperbolic problems. In many cases, as pointed out by West, Kane and Marsden [81,87], such algorithms do not conserve energy in an obvious way. In the case of Newmark's algorithm, the authors pointed out that for an arbitrary $\beta$, $\gamma < 1/2$, Newmark algorithms will dissipate energy, $\gamma > 1/2$ energy will increase and $\gamma = 1/2$ energy will remain oscillatory. Therefore, once mesh refinement is introduced, kinetic energy should be closely monitored. A sharp rise or fall in kinetic energy due to refinement or unrefinement reflects the weakness of the mesh refinement algorithm.

#### 3.8.3.1 Energy in Central Difference Method

Define $[V] = v_{n+1} - v_n, < V >= 0.5(v_{n+1} + v_n)$, and

Strain Energy: $SE_n = 0.5 * d_n K d_n$

Total Energy = Kinetic Energy + Strain Energy

$TE_n = SE_n + KE_n$

Kinetic Energy: $KE_n = 0.5 * v_n M v_n$

The difference of energies at time stations $t_{n+1}$ and $t_n$ is given by:

$$TE_{n+1} - TE_n = [TE] = [SE] + [KE]$$

The jump in strain energy is given by: *[SE] = [d]k<d>*. Similarly: *[KE] = [v]M<V>*.

Using the central difference method:

$$d_{n+1} = d_n + \Delta t v_n + 0.5\Delta t^2 a_n$$

$$v_{n+1} = v_n + 0.5\Delta_t (a_n + a_{n+1})$$

Hence,

$$[d] = d_{n+1} - d_n = \Delta t v_n + 0.5\Delta t^2 a_n$$

$$[V] = v_{n+1} - v_n = \Delta t < a > ,$$

and $$[d] = \Delta_t < V > -0.5\Delta t^2 a_n .$$

Using the equation of motion *(M[a]=-k[d]),*

$$M<a>=-K<d>$$

we get: $$M<a>= M[v] / \Delta t$$

Thus,
$$\begin{aligned}
[TE] &= [d]k < d > +[V]M < V > \\
&= [d]K < d > +\Delta_t < a > M < V > \\
&= [d]K < d > - < d > k\Delta_t < V > \\
&= < d > K([d] - \Delta t < v >) \\
&= < d > K(-).5\Delta t^2[a]) \\
&= 0.5\Delta t^2 < a > M[a] \neq 0
\end{aligned}$$

Hence energy is not conserved by the central difference method.

### 3.8.3.2 Kinetic Energy Before and After Refinement

As pointed out earlier, it is necessary to understand the kinetic energy conservation property once refinement is introduced. We are going to use refinement on the same one-dimensional mesh described in figure 3-2.

Kinetic Energy (KE) of the initial mesh:

$$KE_{12} = \frac{1}{2} v_{12}^T m_{12} v_{12} = \frac{1}{2} \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \rho l/2 & 0 \\ 0 & \rho l/2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \frac{1}{4} \left[ \rho l v_1^2 + \rho l v_2^2 \right]$$

$$KE_{23} = \frac{1}{2} v_{23}^T m_{23} v_{23} = \frac{1}{2} \begin{bmatrix} v_2 & v_3 \end{bmatrix} \begin{bmatrix} \rho l/2 & 0 \\ 0 & \rho l/2 \end{bmatrix} \begin{bmatrix} v_2 \\ v_3 \end{bmatrix} = \frac{1}{4} \left[ \rho l v_2^2 + \rho l v_3^2 \right]$$

Here we are ignoring the scalar ½ in the definition of kinetic energy since it does not alter our conclusion and makes calculations easier to see.

Total Kinetic Energy of the initial mesh:

$$KE = KE_{12} + KE_{23} = \frac{1}{4} \left[ \rho l v_1^2 + 2 \rho l v_2^2 + \rho l v_3^2 \right]$$

On application of uniform velocity $v_1 = v_2 = v_3 = v$, the total kinetic energy is:

$$KE = \rho l v^2$$

#### 3.8.3.2.1 Row-Sum Technique:

Kinetic Energy of the refined mesh:

$$KE_{12} = \frac{1}{2} v_{12}^T m_{12} v_{12} = \frac{1}{2} \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \rho l/2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \frac{1}{4} \rho l v_1^2$$

$$KE_{23} = \frac{1}{2} v_{23}^T m_{23} v_{23} = \frac{1}{2} \begin{bmatrix} v_2 & v_3 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & \rho l/2 \end{bmatrix} \begin{bmatrix} v_2 \\ v_3 \end{bmatrix} = \frac{1}{4} \rho l v_3^2$$

$$KE_{42} = \frac{1}{4}\begin{bmatrix} \frac{(v_1 + v_2)}{2} & v_2 \end{bmatrix} \begin{bmatrix} \frac{\rho l}{2} & 0 \\ 0 & \frac{\rho l}{2} \end{bmatrix} \begin{bmatrix} \frac{(v_1 + v_2)}{2} \\ v_2 \end{bmatrix} = \frac{1}{4}\left[ \frac{\rho l (v_1 + v_2)^2}{8} + \frac{\rho l v_2^2}{2} \right]$$

$$KE_{25} = \frac{1}{4}\begin{bmatrix} v_2 & \frac{(v_2 + v_3)}{2} \end{bmatrix} \begin{bmatrix} \frac{\rho l}{2} & 0 \\ 0 & \frac{\rho l}{2} \end{bmatrix} \begin{bmatrix} v_2 \\ \frac{(v_2 + v_3)}{2} \end{bmatrix} = \frac{1}{4}\left[ \frac{\rho l v_2^2}{2} + \frac{\rho l (v_2 + v_3)^2}{8} \right]$$

Total Kinetic Energy of the refined mesh:

$$KE = KE_{12} + KE_{23} + KE_{42} + KE_{25}$$

$$KE = \frac{1}{4}\left[ \rho l v_1^2 + \rho l v_3^2 + \frac{\rho l (v_1 + v_2)^2}{8} + \frac{\rho l v_2^2}{2} + \frac{\rho l v_2^2}{2} + \frac{\rho l (v_2 + v_3)^2}{8} \right]$$

If $v_1 = v_2 = v_3 = v$, the total kinetic energy is: $KE = \rho l v^2$

Thus if the structure is fairly rigid and velocity at every point is approximately the same, refinement will not add any energy to the system. This is especially true for most of the linear elastic small strain (strain < 5%) problems. This problem can be seen as three balls connected by two consecutive springs in a straight line. The difference in velocity of two consecutive points will cause strain in the element. Hence the difference in velocity has to be relatively low to satisfy the small strain assumption of the problem. It is also interesting to point out that,

1. For any arbitrary nonzero $v_1, v_2, v_3$, $\{V\}^T[M]\{V\} > 0$ after refinement . As pointed out in the section on mass matrix, this indicates that a lumped mass matrix constructed on a refined basis will stay positive definite.

2. $\frac{\partial(KE)}{\partial v} = 2\rho l v$; On application of uniform velocity, the first derivative of kinetic energy with respect to velocity gives us momentum of the system with uniform velocity.

Now consider the function,

$$F(v_1, v_2, v_3) = \text{Energy after refinement - Energy before refinement}$$

$$F(v_1, v_2, v_3) = \frac{\rho l (v_1 + v_2)^2}{8} + \frac{\rho l (v_2 + v_3)^2}{8} - \rho l v_2^2 \qquad (3\text{-}17)$$

It is easy to see that

$$\frac{\rho l (v_1)^2}{8} + \frac{\rho l (v_3)^2}{8} > F(v_1, v_2, v_3) > -\rho l v_2^2$$

$F(v_1, v_2, v_3) = 0$ will represent all the values of v1, v2 and v3 for which energy will remain conserved.

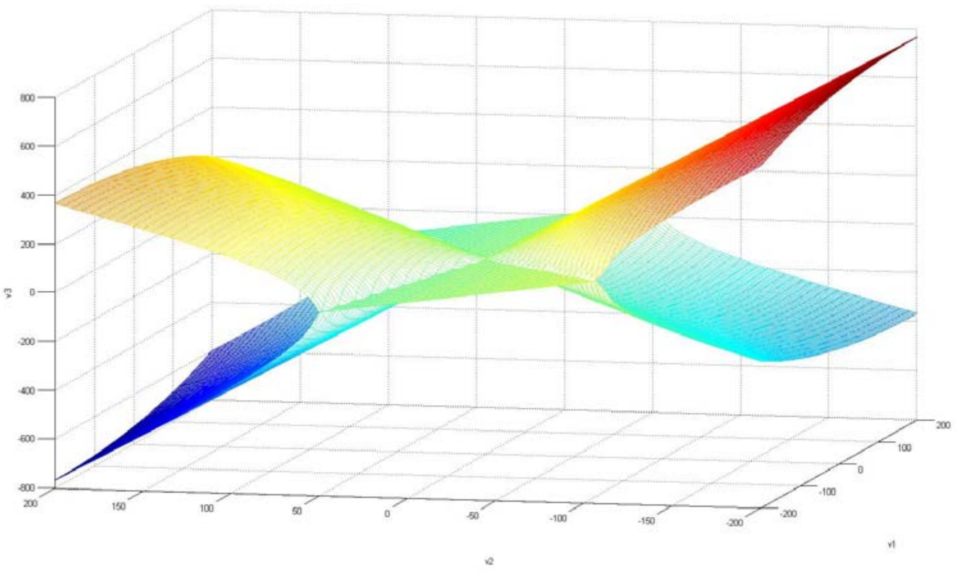


**Figure 3-7: Points representing no increase in energy on $v_1$, $v_2$, $v_3$ Cartesian space**

### 3.8.3.2.2 Using Consistent Mass Matrix

Consistent mass matrix is calculated in the previous section. Kinetic energy using a consistent mass matrix can be written:

$$KE = \frac{\rho l}{2}\begin{bmatrix} v_1 \\ (v_2+v_1)/2 \\ v_2 \\ (v_2+v_3)/2 \\ v_3 \end{bmatrix}^T \begin{bmatrix} 1/3 & 1/8 & 1/24 & 0 & 0 \\ 1/8 & 1/12 & 1/24 & 0 & 0 \\ 1/24 & 1/24 & 1/3 & 1/24 & 1/24 \\ 0 & 0 & 1/24 & 1/12 & 1/8 \\ 0 & 0 & 1/24 & 1/8 & 1/3 \end{bmatrix} \begin{bmatrix} v_1 \\ (v_2+v_1)/2 \\ v_2 \\ (v_2+v_3)/2 \\ v_3 \end{bmatrix}$$

$$= v_1(\frac{19}{48}v_1+\frac{5}{48}v_2)+(\frac{1}{2}v_1+\frac{1}{2}v_2)(\frac{1}{6}v_1+\frac{1}{12}v_2)+v_2(\frac{1}{16}v_1+\frac{3}{8}v_2+\frac{1}{16}v_3)+(\frac{1}{2}v_3+\frac{1}{2}v_2)(\frac{1}{12}v_2+\frac{1}{6}v_3)+v_3(\frac{5}{48}v_2+\frac{19}{48}v_3)$$

If velocities are uniform, $\vec{v}_1 = \vec{v}_2 = \vec{v}_3 = \vec{v}$, kinetic energy of the structure on the refined mesh can be written:

$$= \rho l\left( v(\frac{19}{48}v+\frac{5}{48}v)+(\frac{1}{2}v+\frac{1}{2}v)(\frac{1}{6}v+\frac{1}{12}v)+v(\frac{1}{16}v+\frac{3}{8}v+\frac{1}{16}v)+(\frac{1}{2}v+\frac{1}{2}v)(\frac{1}{12}v+\frac{1}{6}v)+v(\frac{5}{48}v+\frac{19}{48}v) \right)=\rho l v^2$$

Therefore kinetic remains conserved using a consistent mass matrix on application of uniform velocity.

Kinetic energy using unrefined mesh and a consistent mass matrix can be written:

$$KE = \frac{1}{6}\left(2v_1^2 + 2v_1v_2 + 4v_2^2 + 2v_2v_3 + 2v_3^2\right)$$

Therefore gain in kinetic energy for any arbitrary v1, v2 and v3 can be written:

$$KE\_gain = \frac{1}{48}\left(-7v_1^2 + 2v_1v_2 + 10v_2^2 + 2v_2v_3 - 7v_3^2\right)$$

Similar conclusions can be derived from kinetic energy gain as in the previous section. Zero kinetic energy gain points can be plotted using a v1, v2, v3 coordinate axis,

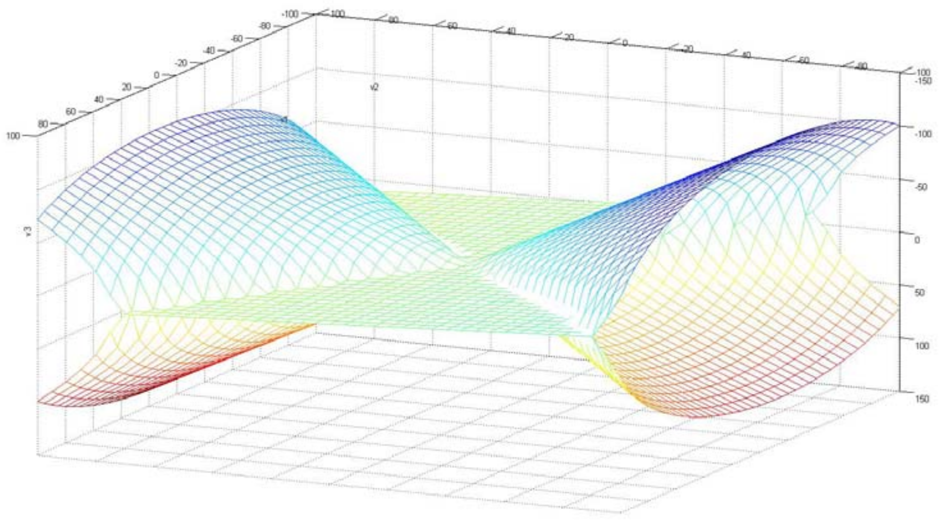


**Figure 3-8: Points representing no increase in energy on $v_1$, $v_2$, $v_3$ Cartesian space for consistent mass matrix**

### 3.8.3.2.3 Using HRZ Lumping

HRZ lumped mass matrix is calculated in a previous section. Kinetic energy using HRZ lumped mass matrix can be written as:

$$KE = \frac{\rho l}{14}\begin{bmatrix} v_1 \\ (v_1+v_2)/2 \\ v_2 \\ (v_3+v_2)/2 \\ v_3 \end{bmatrix}\begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix}\begin{bmatrix} v_1 \\ (v_1+v_2)/2 \\ v_2 \\ (v_3+v_2)/2 \\ v_3 \end{bmatrix}$$

$$KE = \frac{\rho l}{14}\left(4v_1^2 + \left(\frac{v_1+v_2}{2}\right)^2 + 4v_2^2 + \left(\frac{v_3+v_2}{2}\right)^2 + 4v_3^2\right)$$

If velocities are uniform, $\vec{v}_1 = \vec{v}_2 = \vec{v}_3 = \vec{v}$, momentum of the structure on the refined mesh can be written as:

$$\frac{\rho l}{14}\left(4v^2+\left(\frac{v\ +v}{2}\right)^2+4v^2+\left(\frac{v\ +v}{2}\right)^2+4v^2\right)=\rho l v^2$$

Therefore kinetic energy remains conserved using the HRZ lumped mass matrix on application of uniform velocity. Kinetic energy using an unrefined mesh and HRZ mass matrix can be written as: $KE=\frac{1}{2}\left(v_1^2+2v_2^2+v_3^2\right)$

Therefore gain in kinetic energy for an arbitrary v1, v2 and v3 can be written as,

$$KE\_gain=\frac{1}{28}\left(v_1^2+2v_1v_2-5v_2^2+2v_2v_3+3v_3^2\right)$$

Similar conclusions can be derived from kinetic energy gain as in the case of the row-sum technique. 0 kinetic energy gain points can be plotted as:

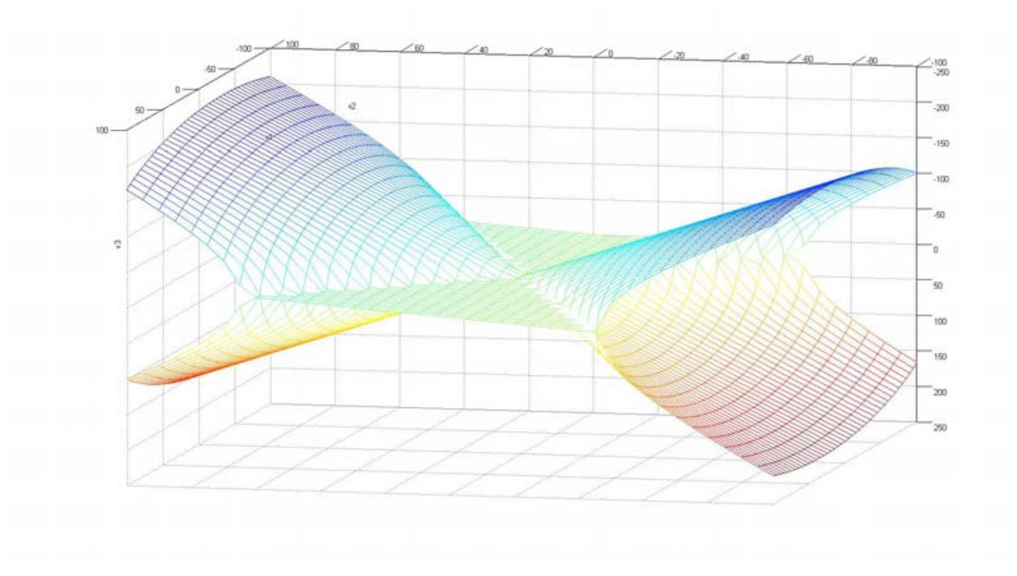


**Figure 3-9: Points representing no increase in energy on $v_1$, $v_2$, $v_3$ Cartesian space**

### 3.8.3.3 Conclusion from Conservation Behavior

1. Conservation properties with different types of mass matrices show very similar trends.
2. Although the consistent mass matrix shows superior convergence properties, computationally it is very expensive in comparison to the HRZ technique and row-sum technique. The HRZ technique is more expensive than the row-sum technique, since it requires calculation of the diagonal elements of the consistent mass matrix first. However, unlike the row-sum technique, the HRZ technique does not produce negative masses and produces comparatively reasonable results.
3. Energy and momentum almost stay conserved while solving small strain problems with CHARMS. For large strain problems, more research is required to understand the energy bounds produced with refinement and their impact on the numerical solution.

### 3.8.4 Conservation of Mass, Momentum and Energy with Unrefinement

In this simplistic example, it is easy to see that the difference in conserved quantities before and after unrefinement will stay the same as refinement, because one will end up with exactly the same mesh after deactivating finer functions and activating coarser ones. However, in more complex problems this does not hold true. Unrefinement by nature is lossy. However, numerical experiments have shown that the gain in computational cost due to unrefinement far outweighs the error induced by unrefinement.

## 3.9 Example

A bar made of alloy steel is excited on one of it its ends with a half sinusoidal forcing function. A linear elastic small strain material model is used. (Alloy steel (AS.049 9310 nickel-chromium-molybdenum) yields about 4% strain)

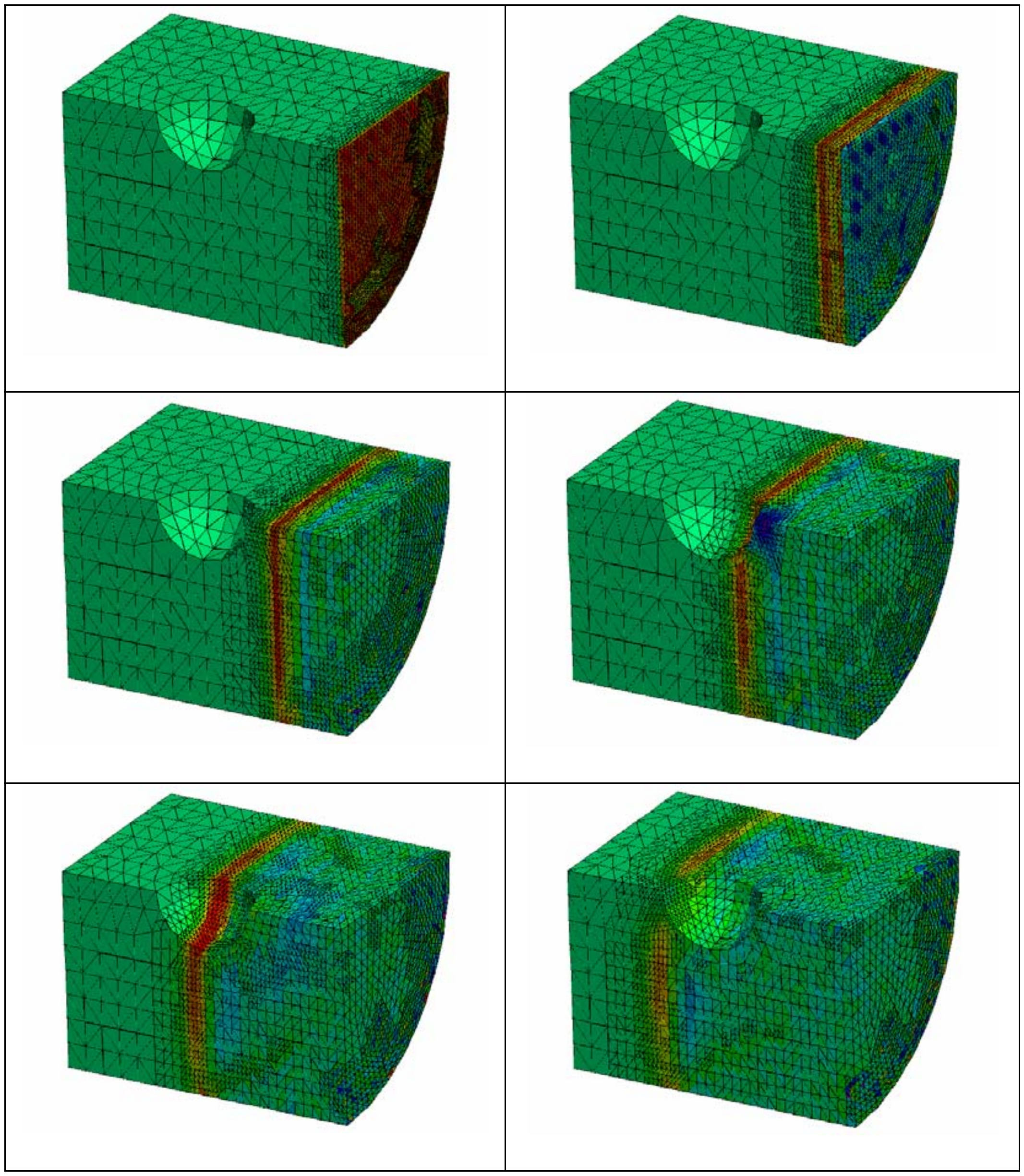
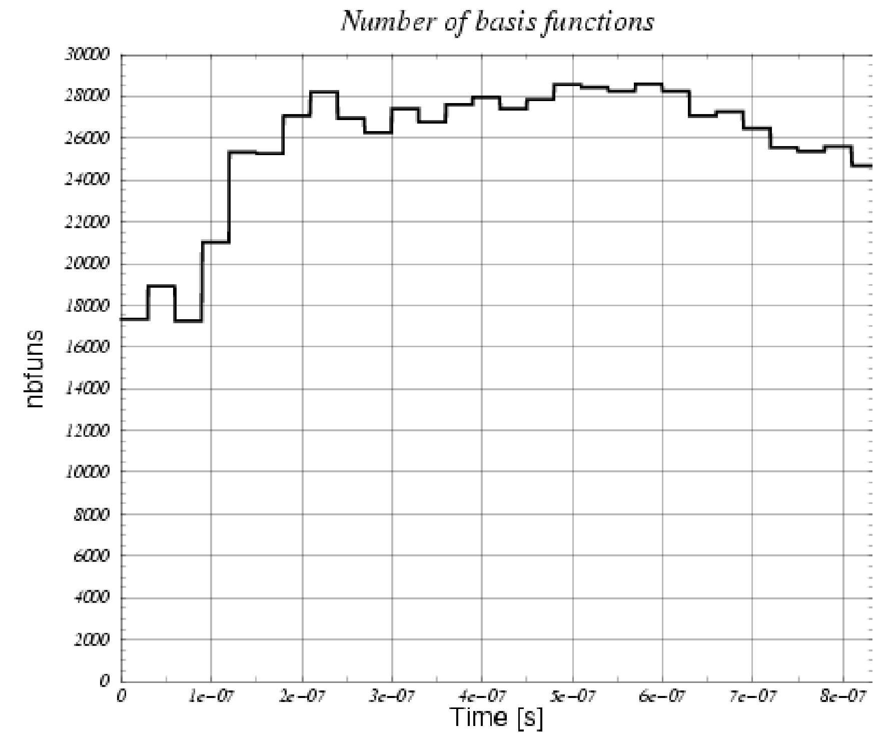


**Figure 3-10: Mesh refinement at different time steps as wave front travels through the bar.**



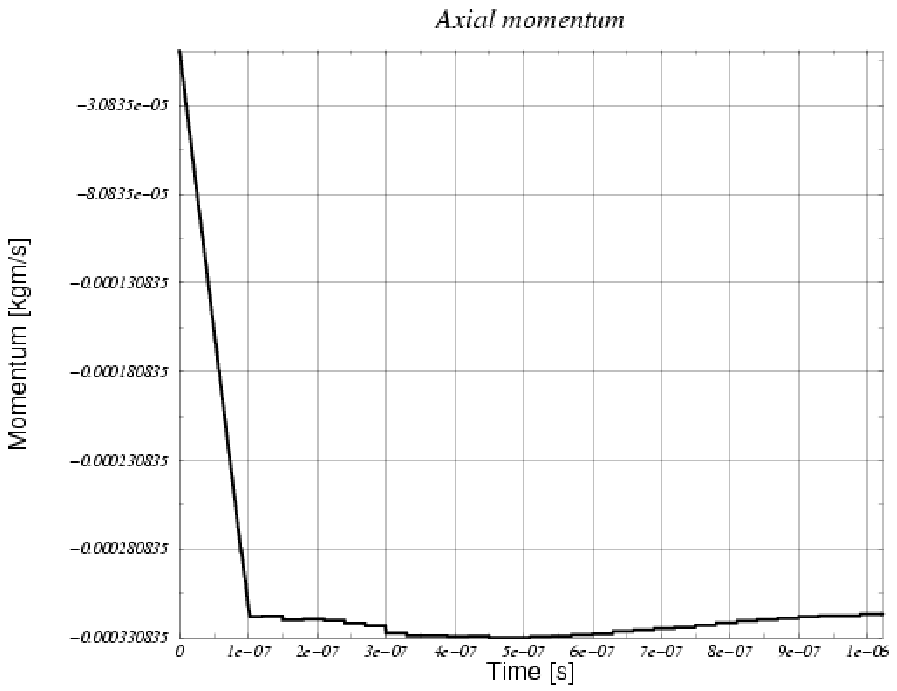


**Figure 3-11: Number of basis functions generated (top), Conservation of momentum in this process (bottom)**

We have already seen from momentum conservation properties of the Newmark method that in the absence of an external load, momentum will remain conserved. The problem presented here is a linear elastic small strain dynamic problem. Hence from the conservation properties of the adaptive refinement method, momentum remains almost conserved during refinement and unrefinement, as the applied load dies at the 1e-07 seconds. It is interesting to see that the adaptive refinement method keeps the problem size in check. After adding the basis functions up to time = 2e-07 seconds, the number of basis functions is almost kept constant by the process of refinement and unrefinement.

# CHAPTER 4 : A VALIDATION CASE

## 4.1 Background and Introduction

Finite element methods have been extensively utilized to validate non destructive evaluation and testing experiments on mechanical parts and assemblies. As indicated in the previous chapter, a 3D solution of guided waves problem needs a very fine mesh to capture the intermediate frequencies, which is very often the case when validating nondestructive testing problems. In the past, in order to avoid a huge computational cost, the researcher approached this problem by singling out the response of individual frequencies, and if required, combined them together using signal processing techniques. [74,75,98] used similar techniques for the simulation of defect detection using lamb waves. Gaveric [96] developed a technique for efficiently predicting mode shapes and dispersion curves between 1 KHz and 7 KHz frequency. Wilcox [97] simplified Gaveric's technique for a more general implementation in finite element codes. However, none of the techniques presented so far was able to simulate the full complexity of wave propagation problems related to NDE experiments, and their use was limited to a small subset of problems. The complete solution of the problem resulting from NDE experiments requires direct time integration of the equation of motion instead of analysis of the response of an individual set of frequencies. As pointed out in [73,75], in order to accurately capture such a response, between ten and twenty elements per wavelength will be needed. For problems involving an unsymmetric computational domain, such as the one mentioned in this chapter, the problem size might become very large. We solve this problem by beginning with a very coarse mesh and then tracking the wave front with the adaptive dynamic technique developed in a previous chapter.

## 4.2 Experimental Setup

The experimental setup is shown in Figure 4-2. Rails were placed on wood sleepers (0.61-m spacing) with steel pads and no fasteners. An instrumented hammer was used to excite a broadband signal at one end of the rail. Piezoelectric accelerometers were used to detect the waves as they propagated past each sensor. Two sets of accelerometer arrays were set up along the rail, each located four feet away from the defect location and eight feet from the rail ends. The two arrays measured reflection and transmission coefficients from the defects. Each array consisted of three uniaxial ICPÂ® accelerometers. One accelerometer was mounted in the vertical direction, another was mounted in the transverse direction, and the last was mounted in the longitudinal (rail-running) direction. Three types of head defects were manufactured by saw cutting: a transverse defect, an oblique defect at 20 degrees from the rail transverse direction, and a second oblique defect at 35 degrees from the rail transverse direction. Each defect type was cut to four increasing depths. The four defect sizes, indicated in Figure 4-1 as 1Q, 2Q, 3Q, and 4Q, were each increased by approximately one quarter of the head cross-sectional area. Three different tests were run for each of the defects investigated. The first test was a vertical strike test to generate a vertical bending mode. The second test was a transverse strike test to generate a lateral bending mode. The last test was a longitudinal strike for the generation of a longitudinal (axial) mode. For further information regarding the experimental results for the other defect sizes and orientations refer to [76]. These methods of wave generation have been previously shown to successfully isolate the intended vibrating mode. The propagating signals were analyzed by a joint time-frequency analysis based on the continuous wavelet transform (CWT). The advantage of the wavelet transform over other time-frequency analyses is its multi-resolution capability optimizes the time resolution and the frequency resolution within the limits of the Heisenberg uncertainty principle. The Gabor wavelet transform (GWT), which provides the best balance between time and frequency resolution, was used with center frequency 2 and shaping factor 5.336. Such values are appropriate for the study of dispersive multi-mode signals, including rail vibrations.

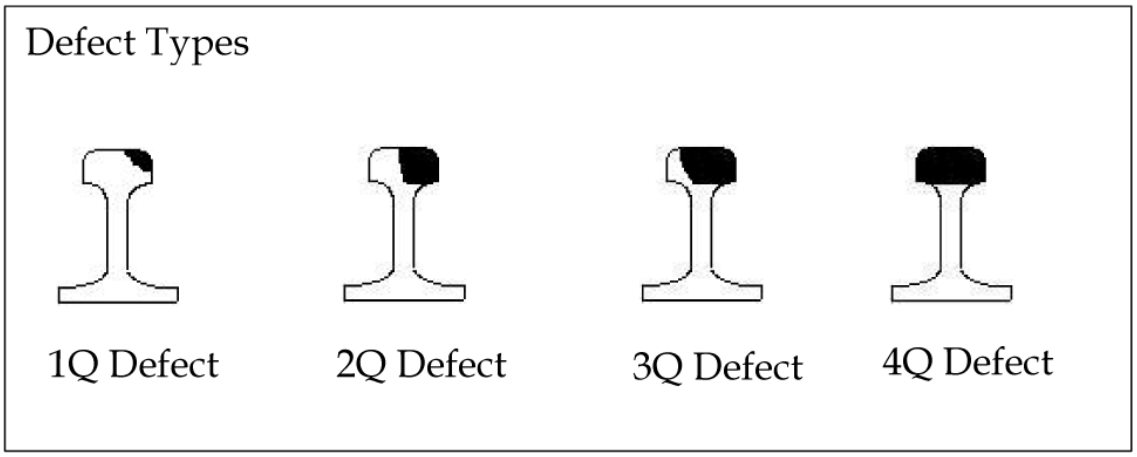


**Figure 4-1: Different types of defects.**

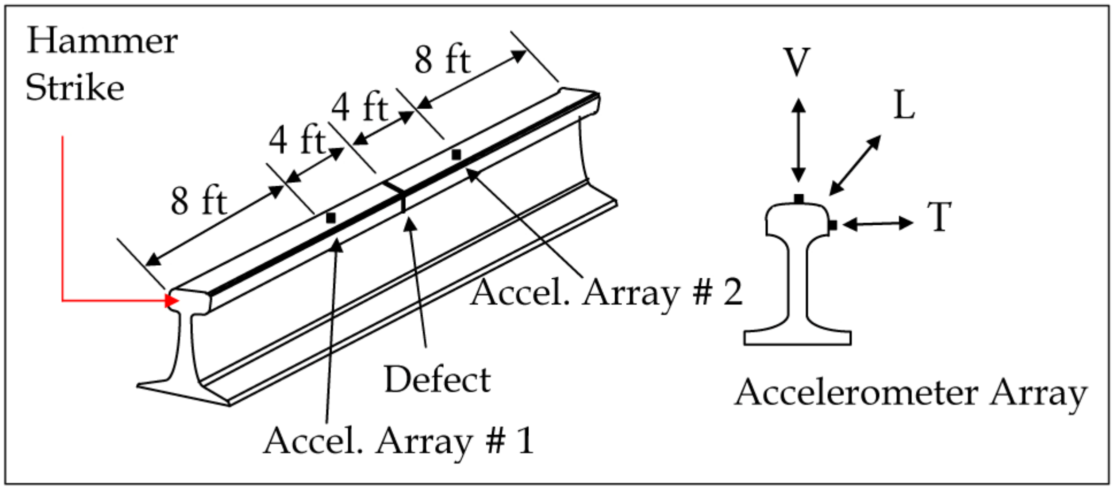


**Figure 4-2:** **Experimental Setup.**

## 4.3 Post Processing of Time Signal

In order to derive a conclusion from the time signal achieved by the experiment discussed in section 4.1, continuous wavelet transform was applied to the time signal. The continuous wavelet transform of a time signal is calculated:

$$Wf(u,s) = \int_{-\infty}^{+\infty} f(t) \cdot \frac{1}{\sqrt{s}} \cdot \psi^* \left( \frac{t-u}{s} \right) \cdot dt$$

where $\psi^*(t)$ is the complex conjugate of the mother wavelet. $\psi(t)$ is given by the following equation:

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}} \cdot \psi \frac{t-u}{s}$$

Here $u$ shifts the wavelet in time and $s$ controls the wavelet frequency bandwidth. Gabor wavelet provides the best balance between time resolution and frequency resolution. Therefore, it was chosen as the mother wavelet for continuous wavelet transform. In this case, Gabor wavelet is a Gaussian window defined as follows:

$$\psi(t)=\frac{1}{\sqrt[4]{\sigma^2\cdot p}}\cdot\exp\left(\frac{-t^2}{2\cdot\sigma^2}\right)=\frac{1}{\sqrt[4]{p}}\cdot\sqrt{\frac{\eta}{G_s}}\cdot\exp\left(\frac{-(\eta/G_s)^2t^2}{2}\right)\cdot\exp(i\cdot\eta\cdot t)$$

where $\eta$ is the wavelet center frequency, $\sigma$ is the standard deviation of Gaussian window and $G_s=\sigma\cdot\eta$.

The energy density spectrum of a continuous wavelet transform is defined as $P_W f(u,s)=|Wf(u,s)|^2$ and is commonly referred to as a scalogram. A scalogram indicates the energy of signal *f(t)* in the Heisenberg box of each wavelet window $\psi_{u,s}(t)$ around time *t* = *u* and angular frequency $\Omega=\eta s$. The scalogram in our case is normalized by a scaling parameter. Besides providing the time-frequency information of the signal component, the scalogram retains the signal energy content.

The scalogram was than used to calculate reflection coefficients for different frequencies. In Figure 4-2, sensors are mounted at array no. 1 and array no. 2. Let $A_i$ be the magnitude of the wave energy when it reaches array no. 1 and $A_r$ be the magnitude of returning echo energy from the defect at array no. 1. Then the Reflection coefficient *R* is given by,

$$R=\frac{A_r}{A_i}$$

## 4.4 Numerical Approximation

From our discussion so far, we are clear that it is very difficult for us to utilize the solution of the equation of motion in the frequency domain since we may not be sure about reflecting frequncies from an unknown shape and size defect. We have a better sense of the reflection coefficient. Therefore a direct time integration scheme is the correct approach for us. We utilized the explicit solver based on CHARMS as described in the previous chapter. The last chapter scrutinizes a variety of mass matrices, their conservation properties and their role in convergence of natural frequencies of the system. We decided to go with least expensive and computationally proven row-sum technique to construct a mass matrix. The solution of the equation of motion was achieved as described in the previous chapter.

## 4.5 Analysis of Numerical Results

The purpose of this research was to determine how the adaptive mesh refinement algorithm described in a previous chapter could successfully be used to model the types of long range guided rail waves discussed in the experimental portion of this chapter. The modeling of the interaction of these waves with defects similar to those studied experimentally was of special interest. Determining the parameters required for accurate modeling of guided rail waves was the primary goal. An algorithm that could accurately predict these types of waves could be used to simulate defects that are impossible to manufacture in the lab, thereby providing information that could not otherwise be collected. Although it was known that the finest portion of the mesh should produce elements that were somewhere between 7.5 mm and 3.75 mm, according to the conventions discussed in the previous chapter, the actual mesh size that would produce good results was initially unknown. Mesh size is a function of the number of adaptation levels used. The starting mesh was a uniform mesh consisting of eight-node solid elements with longitudinal (in the direction of the wave propagation) dimensions of 20 mm. This coarse (level zero) mesh was equivalent to sampling the highest frequency waves 3.75 times per wavelength, which is a third of the minimum required sampling rate of 10 samples per wavelength. This uniform mesh could be run on a PC running Linux with a 1 GHz Pentium III processor and 512 MB of ram. One test (calculation of the reflection coefficients for one defect) could be solved in roughly 24 hours. Unfortunately, this coarse mesh provided rather poor results. An example of the results for a vertical strike test (4Q zero degree defect) is shown in Figure 4-3. Although the scalogram for this test appears very similar to an experimentally derived scalogram, the coarseness of the mesh could not adequately simulate the higher frequencies where the reflection information would have been found (Figure 4-4). Although the results were disappointing, they were expected because it was already known that at least a 7.5 mm mesh would be required for the higher frequencies. Other defect sizes and orientations were studied and similar results were found. It was observed that with a 20 mm mesh size, about 425 MB RAM was required for solution of the problem when symmetry and sparsity of the stiffness matrix was used for calculation. Uniform division of the mesh to achieve half of the mesh size will result in the division of each brick element into 8 brick elements. As mentioned earlier, our required mesh size to capture all lengths was at least 7.5 mm or lower. That will give rise to between 1,684,800 to 13,478,400 degrees of freedom with eight node hexahedral elements. It was hard to reduce the size of the problem since the frequency of the waves that were traveling was unknown and the finite domain was asymmetric due to the presence of defects. This indicated a massive problem with several million degrees of freedom and a requirement of over 50 GB of RAM. Use of an implicit method is completely ruled out due to computational cost. Adaptive mesh refinement seems to be the only solution in similar circumstances.

The next round of tests used a mesh with one level of refinement like that shown in Figure 4-5. Still, results were not as good as they were supposed to be. We noticed that for small and medium size defects, the majority of the energy continued to travel past the defects. The wave front needed to validate the results was the one that traveled past the defects, not the smaller front that reflected back from the defect. This led to errors due to the fact that the portion of the rail affecting the reflection was unrefined because the larger wave front was far away, thereby reducing accuracy of the reflected signal. The initial mesh was slightly modified and the solver was modified to refine the mesh at the locations of highest strain energy only into the area between the hammer impact and defect. These changes produced remarkable improvement in reflection coefficients. At this stage, the reflection coefficients were modified to account for losses during experiment. We decided to achieve the results with even better precision by further increasing the level of refinement. Figure 4-7 shows results with two levels of refinement. Numerical values of reflection coefficient are very close to experimental values. Results are good enough for all practical purposes.

In order to produce results for various defect types in shorter period of time, a parallel version of FAMULS was created on an IBM RS6000 cluster named as Blue Horizon (http://www.sdsc.edu/Resources/bluehorizon.html) using MPI instructions.

Each node allowed access to 8 processors with 4GB of memory per node. Although with the existing algorithm all the cases can be run on the PC, computational time was significantly reduced this way.

## 4.6 Conclusion

Simulation of NDE experiment using CHARMS opens the door for various problems involving application of long range guided waves in a large computational domain. Unlike different techniques published in the past, there is no practical limitation in terms of frequency range. However, more work is required to reduce the high frequency noise very often seen in these problems. One way is to use Raleigh type damping as a design parameter. [69] shows a correlation of Raleigh damping coefficients with high frequency noise. Another possibility is to use a template mass matrix and solve the design coefficient for the least numerical noise. Use of composite model damping might also help. Similar techniques can be utilized in shock wave propagation problems where instantaneous response in a large finite domain is required.

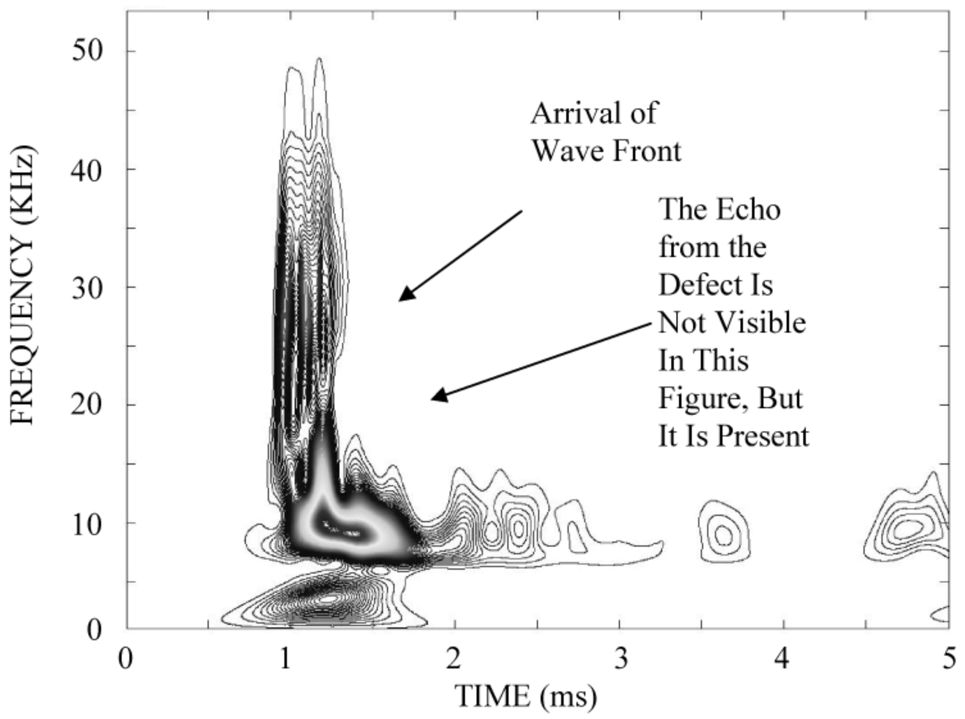


**Figure 4-3:** **Scalogram using coarse mesh.**

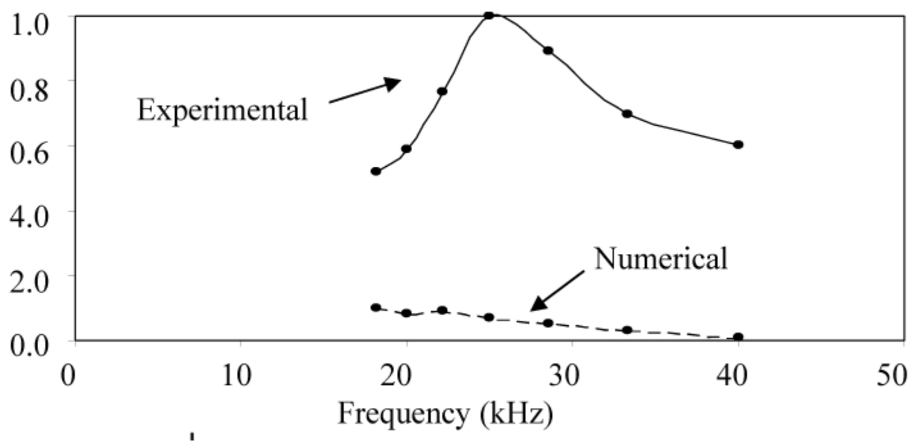


**Figure 4-4:** **Comparison of reflection coefficients with coarse mesh.**

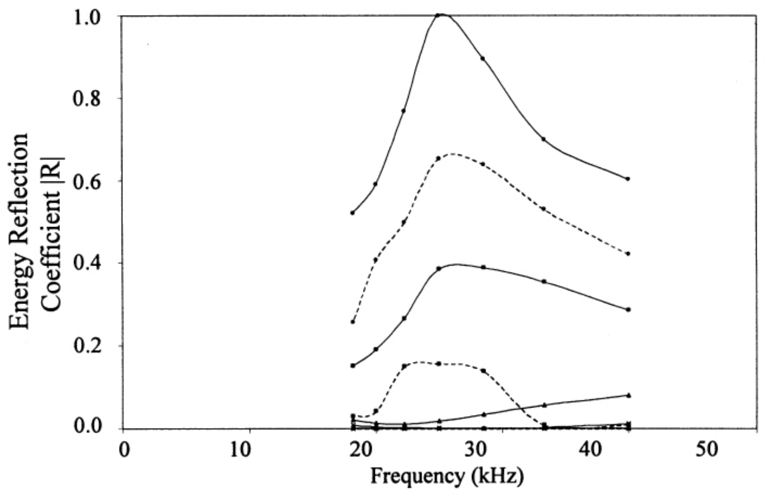


**Figure 4-5:** **Same-Mode Energy Reflection Coefficient Spectra Obtained for the Vertical Test of Zero Degree Defects. (●, 4Q Size Defect; ■, 3Q Size Defect; ▲, 2Q Size Defect; X, 1Q Size Defect; Dashed Lines Correspond to Numerical Results)**

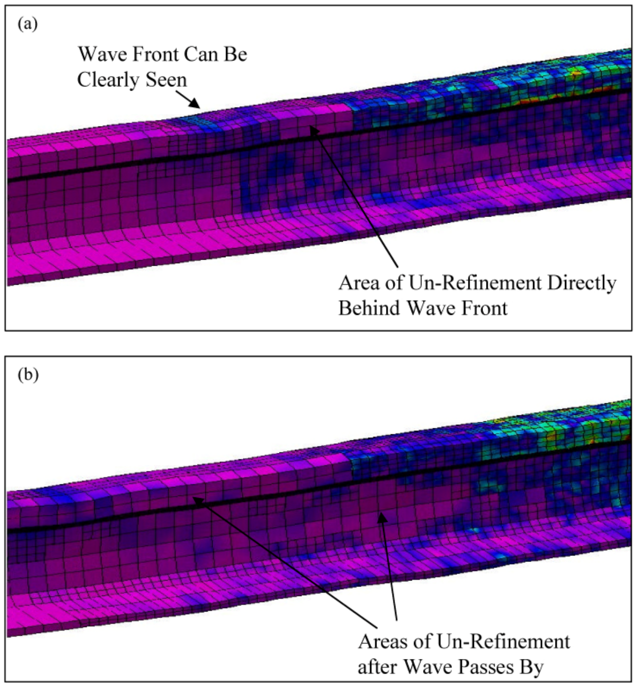

(a)
Wave Front Can Be Clearly Seen
Area of Un-Refinement Directly Behind Wave Front
(b)
Areas of Un-Refinement after Wave Passes By

**Figure 4-6:** **Wave Front Tracking and Mesh Refinement. (a) Rail with Refined Mesh, (b) Rail illustrating Wave Front Tracking & Mesh Refinement and Unrefinement. (One level of refinement is shown)**

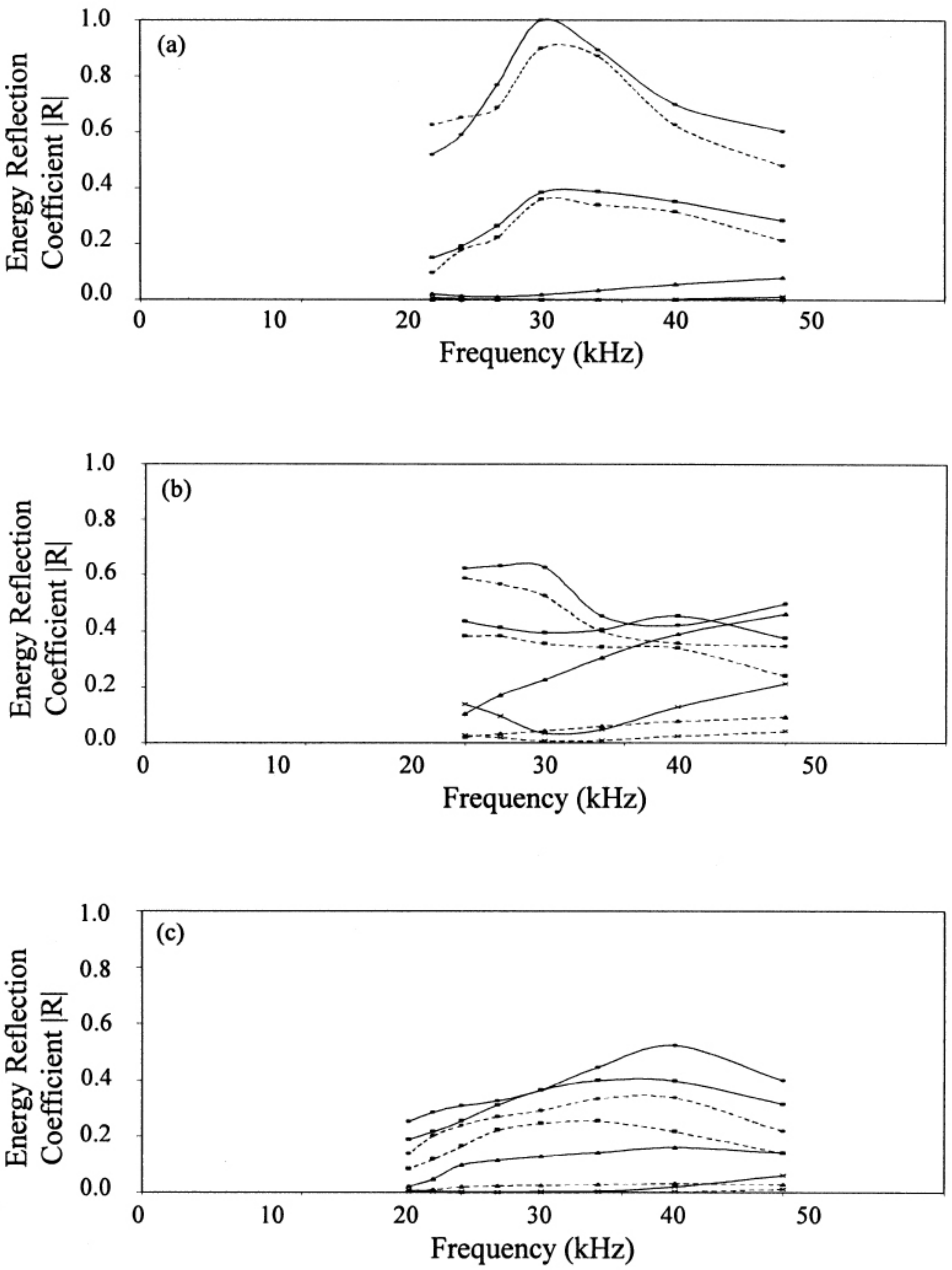


**Figure 4-7: Same-Mode Energy Reflection Coefficient Spectra Obtained for the Vertical Test of Zero Degree Defects. (●, 4Q Size Defect; ■, 3Q Size Defect; ▲, 2Q Size Defect; X, 1Q Size Defect; Dashed Lines Correspond to Numerical Results)**

# CHAPTER 5 : OUTLOOK

Moor's law [82], which relates the number of transistors on integrated circuits to time, has been quite successful in predicting evolution of computational resources over time. Despite the dramatic improvement in computational hardware, demand for computational resources for scientific applications has been unyielding. With powerful computers in hand, more complex problems started looking possible, and a dependence on computationally efficient algorithms to solve similar problems grew in tandem. Fast growing economies, demanding consumers, and a competitive marketplace have forced the products to be lean and more efficient. Creation of such products requires fast and more accurate analysis. With this changing scenario, among other aspects of analysis, much research effort is diverted towards the development of adaptive mesh refinement techniques which can utilize state-of-the-art computational resources optimally.

The previous chapter presents one of many applications of adaptive mesh refinement for dynamic solvers. The need for a general adaptive framework and solution technique is industry wide. The following are some of the research works in progress using CHARMS and a comment on what lies ahead.

## 5.1 Simulation of Tumor

### 5.1.1 Description

Over the last decade there has been tremendous progress in visualization of complex, hidden structures inside the human body during operation [85]. However, a collaborative effort of clinicians, computer scientists, and engineers is required to perform computer assisted image-guided therapy (IGT), and it generally takes place only in research hospitals. This research is oriented towards exploring and simplifying some of the issues that must be tackled in order to fulfill the full promise of these prototype systems in the area of image-guided neurosurgery (IGNS).

Some of the main challenges that neurosurgeons face during tumor resection include removing as much tumor tissue as possible, minimizing the removal of healthy tissue, and disruption of critical anatomical structures should be avoided at all costs. They should also know when to stop the resection process. Apart from these problems, the patient goes through intra-operative shape deformation of the brain as a result of tissue resection, retraction, and loss of cerebrospinal fluid. This causes a steady decline in the accuracy of the preoperative plan if it is not executed with the utmost care. The goal of this research is to quantify and correct for these deformations while a surgery is in progress by dynamically updating preoperative images in a way that allows surgeons to react to changing conditions. This research is published in [78] in great detail.

### 5.1.2 Role of CHARMS

Finite element analysis using CHARMS is going to be instrumental in the simulation of a patient's brain by using images obtained from an intra-operative scanner during surgery. The time constraint on this overall procedure is severe and allows only about two minutes for the finite element biomechanical simulation. Therefore, a parallel adaptive solver will be implemented to take advantage of the San Diego super computer center's hardware.

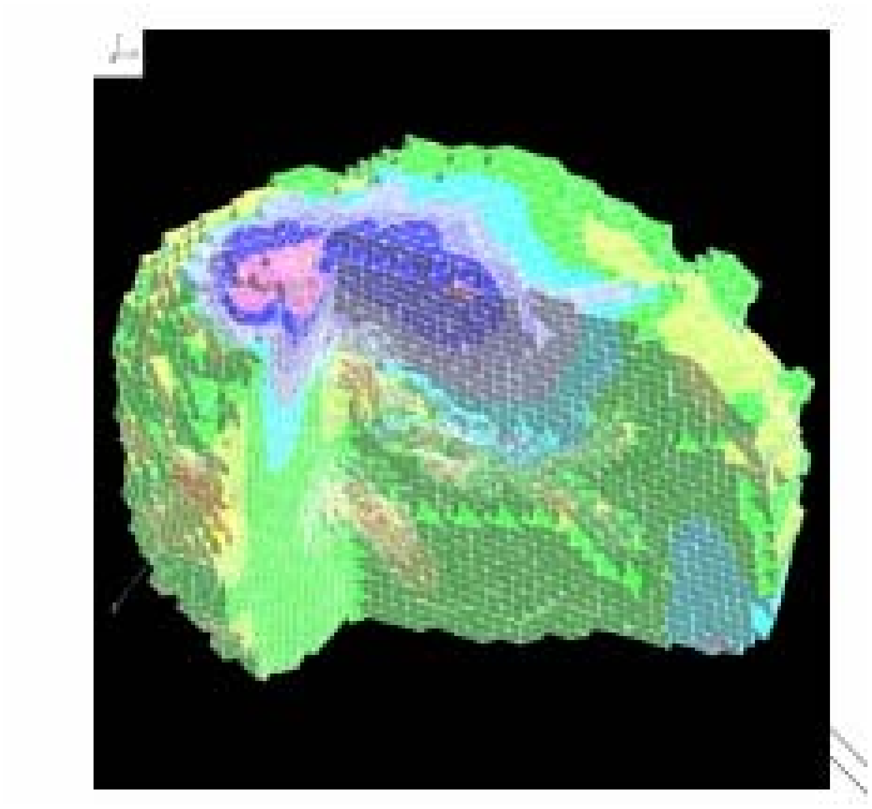


**Figure 5-1: Finite element biomechanical model of cross-sectioned brain.**

## 5.2 Pattern Recognition for NDE applications using Support Vector Machine

### 5.2.1 Description

The set of related supervised learning methods used for classification and regression is called Support vector machines (SVM). One of the highlights of this research suggests that SVM can be used for binary classification or pattern recognition to determine the type and size of defects in railroad tracks. Generally, the model produced using support vector classification only depends upon a subset of the training data, because the cost function for building the model does not consider training points that lie beyond the margin. Therefore, a large amount of data is required to produce a useful model. This work is described in [76] in great detail.

### 5.2.2 Role of CHARMS

Creation of such a large volume of data may not be possible from the NDE experiments, as explained in chapter 3, because of the requirement of enormous time and resources. Therefore, after successful numerical verification of experimental results with numerical results using CHARMS, the author proposed to create an SVM model using numerical results instead of experimental results.

## 5.3 Future Work

### 5.3.1 Applications of CHARMS in Bio-Mechanics Simulations

One of the challenging aspects of biomechanical problems is the fabrication of an initial mesh from data achieved from a CT scan or MRI, which is a collection of pixels inside the overall volume. Surfaces representing the domain boundary and various internal organs can be calculated using techniques such as adaptive deformation [77]. These surfaces are defined using surface triangulation instead of geometric entities such as curves, regions, contours, and surfaces. Therefore, most of the commercial meshing programs, which rely on the presence of geometric entities, cannot be used. Instead, more primitive tools [6] that can utilize a surface mesh to create a volume mesh are very often used. Creation of mesh using such tools requires a massaging of the surface mesh to create well-shaped triangles. In many cases, after the creation of the initial mesh, much manual patch work is required to replace bad quality elements. Convergence analysis of the solution with remeshing can be a nightmare in similar problems, since each iteration requires an increasing amount of effort to achieve a good quality mesh. Therefore, use of an adaptive mesh refinement technique for similar problems is almost essential during convergence analysis. Very often in similar problems, the area of interest is a very small subset of the overall domain [77, 78]. Adaptive mesh refinement techniques such as CHARMS can be very useful in optimizing computational cost while capturing singularities accurately in similar situations [31].

### 5.3.2 Links to Nonlinear Mechanics and Dynamics

We are inspired by the work done by Hauth et al.[83] and others [nonlinear graphics]. Much research has been published on visual simulation, computer graphics, and animation of deformable bodies using complex nonlinear models to assist surgeons by using hierarchical refinement. It seems that hierarchical schemes are very well suited for time integrators, and lessons learned from such visual simulation can be easily applied to nonlinear computational mechanics simulations. Transient schemes, such as the one shown in this thesis, can be easily used to solve nonlinear problems without major effort. However, more work is required to understand conservation properties in similar circumstances, and computational models need to be validated against experimental results.

### 5.3.3 Links to Visual Simulations

Unlike visual simulation models, computational mechanics models based on finite elements are generally tested rigorously for their numerical accuracy, and relatively little effort is spent on the visual appeal of the results. However synergies are happening both ways. Dynamic adaptive models such as the one developed during this thesis can be successfully applied to create high resolution computer animation and graphics [31]. However, the algorithm will have to be customized for both the computational effort and the visual aspects of the results.

### 5.3.3 Use of Asynchronous Variational Integrators

Asynchronous Variational Integrators, or AVIs [84], allow the selection of independent time steps in each element, and the local time steps need not bear an integral relation to each other. Their use of variational structure guarantees exact conservation of energy and momentum. That would be a useful property for adaptive simulation of elastodynamic problems, such as the one presented in this thesis.

### 5.3.4 Error Estimators

Current error estimation implemented in FAMULS relies on lumping the elemental error to each basis function. Therefore, any a posteriori error indicator that calculates elemental error can be used. However, it will be useful to have an error indicator that directly calculates error per basis function during refinement and unrefinement.

### 5.3.5 Lossy Unrefinement

Unrefinement is lossy by virtue. Replacement of finer shape functions with coarser ones may cause loss of information and unexpected artifacts during simulation. The approach described in [31] might work to alleviate this:

> "The deactivation of $\phi_i(x)$ is separated in two steps: (a) the solver no longer considers as unknown coefficient, $u_i$, and (b) the coefficient set to zero. In smooth deactivation, (a) the unknown, $u_i$, is immediately removed from the solvers reach and considered as a prescribed boundary condition, and (b) the value of the coefficient is prescribed by a smoothly decaying function over some finite time interval."

### 5.3.6 Implementation on Parallel Computers

Implementation of CHARMS based solvers on parallel computers was achieved initially using PETSC's support for MPI instructions on IBM RS6000 work stations. We are inspired by the Distributed Adaptive Grid Hierarchy (DAGH) method which supports the Burger-Olinger algorithm for parallel adaptive mesh refinement [DAGH website]. Adaptive finite difference method based on the Burger-Olinger algorithm [DAGH web site] is very similar in terms of data structure for CHARMS activation/deactivation protocols. Hence incorporation of CHARMS with DAGH type framework can produce an efficient parallel adaptive algorithm.

## 5.4 Conclusion

This thesis opens the door for a practical, viable implementation of CHARMS for time-dependent solvers in computational mechanics problems. The object oriented framework FAMULS is modified to incorporate the needs of an explicit dynamic solver. A consistent implementation of a wave propagation solver is studied and implemented with CHARMS. Finally, verification of the numerical solver is performed, with respect to experimental results from the NDE testing of railroad tracks. A brief analysis of conservation properties of the adaptive dynamic solver is also presented as a part of this thesis.

# REFERENCES

[1] Owen, S.J., Meshing Software Survey, Structured Grid Generation Software, http://www.andrew.cmu.edu/user/sowen/software/structured.html

[2] Thompson, J.F., 1985, *Numerical Grid Generation: Foundation and Applications*, Elsevier Science Pub. Co., North-Holland.

[3] Thompson, J.F., 1996, “A Reflection on Grid generation in the 90s: Trends Needs and Influences,” *5th International Conference on Numerical Grid Generation in Computational Field Simulations*, Mississippi State University, Mississippi State, MS, pp. 1029-1110.

[4] Bathe, K.-J., 1996, *Finite Element Procedures*, Prentice-Hall, Upper Saddle River, NJ.

[5] Belytscho, T., Liu, W.K., and Moran, B., 2000, *Nonlinear Finite Elements for Continua and Structures*, John Wiley & Sons Ltd., Chichester, UK.

[6] Krysl, P. and Ortiz, M., 2001, “Variational Delaunay approach to the generation of tetrahedral finite element meshes,” Int. J. Numer. Meth. Engng., **50**(7), pp. 1681-1700.

[7] Geostar Software, SolidWorks Corporation, http://www.cosmosm.com/pages/products/modules_GEOSTAR.html

[8] Almeida, R.C., Feijóo, R.A., Galeão, A.C., Padra, C., and Silva, R.S., 2000, “Adaptive finite element computational fluid dynamics using an anisotropic error estimator,” Comput. Methods Apple. Mech. Engorge. **182**(3-4), pp. 379-400.

[9] Marvelous, D.J., 1990, “Adaptive mesh generation for viscous flows using triangulation,” J. Comput. Phys., **90**(2), pp. 271-291.

[10] Peraire, J., Period, J., and Morgan, K., 1992, “Adaptive remeshing for three dimensional compressible flow computation,” J. Comput. Phys., **103**(2), pp. 269-285.

[11] Berger, M.J. and Oliger, J., 1984, “Adaptive mesh refinement for hyperbolic partial differential equations,” J. Comput. Phys., **53**(3), pp. 484-512.

[12] Radovitzky, R. and Ortiz, M., 1999, “Error estimation and adaptive meshing in strongly nonlinear dynamic problems,” Comput. Methods Apple. Mech. Engrg., **172**(1-4), pp. 203-240.

[13] Molinari, J.F. and Ortiz, M., 2002, “Three-dimensional adaptive meshing by subdivision and edge-collapse in finite-deformation dynamic-plasticity problems with application to adiabatic shear bending,” Int. J. Numer. Meth. Engng., **53**(5), pp. 1101-1126.

[14] Selman, A., Hinton, E., and Bićanić, N., 1997, “Adaptive mesh refinement for localised phenomena,” Computers & Structures, **63**(3), pp. 475-495.

[15] Arnold, D.N., Mukherjee, A., and Pouly, L., 2000, “Locally adapted tetrahedral meshes using bisection,” SIAM J. Sci. Comput., **22**(2), pp. 431-448.

[16] Bänsch, E., 1991, “Refinement in 2 and 3 dimensions,” Impact Comput. Sci. Engrg., **3**(3), pp. 181-191.

[17] Bey, J., 1995, “Tetrahedral grid refinement,” Computing, **55**(4), pp. 271-288.

[18] Bornemann, F., Erdmann, B., and Kornhuber, R., 1993, “Adaptive multilevel methods in three space dimensions,” Int. J. Numer. Meth. Engrg., **36**(18), pp. 3187-3203.

[19] Kallinderis, Y. and Vijayan, P., 1993, “Adaptive refinement-coarsening scheme for three-dimensional unstructured meshes,” AIAA J., **31**(8), pp. 1440-1447.

[20] Liu, A. and Joe, B., 1995, “Quality Local Refinements of Tetrahedral Meshed Based on Bisection,” SIAM J. Sci. Comput., **16**(6), pp. 1269-1291.

[21] Liu, A. and Joe, B., 1996, “Quality local refinements of tetrahedral meshed based on 8-subtetrahedron subdivision,” Math. Comp, **65**(215), pp. 1183-2000.

[22] Rivara, M.-C., 1992, “A 3-D refinement algorithm suitable for adaptive and multi-grid techniques,” Comm. Apple. Num. Meth., **8**, pp. 281-290.

[23] Speares, W. and Berzins, M., 1997, “A 3D unstructured mesh adaptation algorithm for time-dependent shock-dominated problems,” Int. J. Numer. Meth. Fluids, **25**(1), pp. 81-104.

[24] Buscaglia, G.C. and Dari, E.A., 1997, “Anisotropic mesh optimization and its application in adaptivity,” Int. J. Numer. Meth. Engng., **40**(22), pp. 4119-4136.

[25] Castro-Diaz, M.J., Hecht, F., Mohammadi, B., and Pironneau, O., 1997, “Anisotropic unstructured grid adaptation for flow simulations,” Int. J. Numer. Meth. Fluids, **25**(4), pp. 475-491.

[26] Dompierre, J., Vallet, M.-G., Bourgault, Y., Fortin, M., Habashi, W.G., 2002, "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III. Unstructured meshes," Int. J. Numer. Meth. Fluids, **39**(8), pp. 675-702.

[27] De Cougny, H.L. and Shephard, M.S., 1999, "Parallel refinement and coarsening of tetrahedral meshes," Int. J. Numer. Meth. Engrg., **46**(7), pp. 1101-1125.

[28] De l'Isle, E.B. and George, P.L., 1993, "Optimization of Tetrahedral Meshes," *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, I. Babuška, et al., eds., Springer-Verlag, Berlin, IMA Vol. **75**, pp. 97-128.

[29] Joe, B., 1989, "Three-dimensional triangulations from local transformations," SIAM J. Sci. Comput., **10**(4), pp. 718-741.

[30] Pain, C.C., Umpleby, A.P., de Oliveria, C.R.E., and Goddard, A.J.H., 2001, "Tetrahedral mesh optimization and adaptivity for steady-state and transient finite element calculations," Comput. Meth. Apple. Mech. Engrg., **190**(29-30), pp. 3771–3796.

[31] Grinspun, E., 2003, "The Basis Refinement Method," Ph.D. Dissertation, California Institute of Technology, Pasadena, CA.

[32] Krysl, P., Trivedi, A., and Zhu, B., 2004, "Object-oriented hierarchical refinement with CHARMS," Int. J. Numer. Meth. Engng., **60**(8), pp. 1401-1424.

[33] Endres, L. and Krysl, P., 2004, "Octasection-based refinement of finite element approximations on tetrahedral meshes that guarantees shape quality," Int. J. Numer. Meth. Engng., **59**(1), pp. 69-82.

[34] Zhu, B. and Krysl, P., 2003, "Data fitting and shape approximation with CHARMS mesh refinements." in preparation.

[35] Trivedi, A., Krysl, P., McNamara, J., and Lanza di Scalea, F., 2004, "Adaptive simulation of ultrasonic guided waves in railroad tracks using CHARMS," *Proc. 6th World Conference on Computational Mechanics* , Tsinghua University, Beijing, China.

[36] Grinspun, E., Krysl, P., and Schröder, P., 2002, "CHARMS: A simple framework for adaptive simulation," ACM Transactions on Graphics, **21**(3), pp. 281-290.

[37] Möller, P. and Hansbo, P., 1995, "On Advancing Front Mesh Generation in Three Dimensions," Int. J. Numer. Meth. Engrg. **38**(21), pp. 3551-3569.

[38] Borouchaki, H., George, P.L., Hecht, F., Laug, P., Saltel, E., 1997, "Delaunay mesh generation governed by metric specifications. Part I: algorithms. Part II: applications," Finite Elements Anal. Design **25**(1-2) pp. 61-83, 85-109.

[39] Frey, P.J. and George, P.L., *Mesh Generation Application to Finite Elements*, HERMES Science Europe Ltd., Oxford, Paris, 2000.

[40] George, P.L., Borouchaki, H., and Laug, P., 2000, "An efficient algorithm for 3D adaptive meshing," *Finite Elements: Techniques and Developments*, B.H.V. Topping, ed., Civil-Comp Ltd., Edinburgh, UK, pp. 1-11.

[41] George, P.L. and Hecht, F., 1999, "Nonisotropic grids," *Handbook of Grid Generation*, Thompson, J. et al., eds., CRC Press, Boca Raton, FL, pp. 20.1-20.29.

[42] Li, X., Shephard, M.S., and Beall, M.W., 2003, "Accounting for curved domains in mesh adaptation," Int. J. Numer. Meth. Engrg., **58**(2), pp. 247-276.

[43] Rivara, M.-C. and Iribarren, G., 1996, "The 4-triangles longest-side partition of triangles and linear refinement algorithms," Math. Comp., **65**(216), pp. 1485-1502.

[44] Rivara, M.-C., 1997, "New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations," Int. J. Numer. Meth. Engrg., **40**(18), pp. 3313-3324.

[45] Rivara, M.-C. and Inostroza, P., 1997, "Using longest-side bisection techniques for the automatic refinement of Delaunay triangulations," Int. J. Numer. Meth. Engrg., **40**(4), pp. 581-597.

[46] Wille, S.O., 1992, "A structured tri-tree search method for generation of optimal unstructured finite element grids in two and three dimensions," Int. J. Numer. Meth. Fluids, **14**(7), pp. 861-881.

[47] Plaza, A., Padrón, M.A., and Carey, G.F., 2000, "A 3D refinement/derefinement algorithm for solving evolution problems," Apple. Numer. Math., **32**(4), pp. 401-418.

[48] Plaza, A. and Carey, G.F., 2000, "Local refinement of simplicial grids based on the skeleton," Apple. Numer. Math., **32**(2), pp. 195-218.

[49] Language, H.P., 1999, *Computational Partial Differential Equations*, Springer-Verlag, Berlin.

[50] Zorin, D., and Schröder, P., 2001, “A unified framework for primal/dual quadrilateral subdivision schemes,” Comput. Aided Geom. Des., **18**(5), pp. 429-454

[51] Zorin, D., and Schröder P., eds., 2000, “Subdivision for modeling and animation,” ACM SIGGRAPH 2000 Course Notes.

[52] Yserentant, H., 1986. “On the Multi-Level Splitting of Finite Element Spaces,” Numer. Math., **49**(4), pp. 379-412.

[53] Duarte, C.A. and Babuška, I., 2002, “Mesh-independent p-orthotropic enrichment using the generalized finite element method,” Int. J. Numer. Meth. Engrg., **55**(12), pp. 1477-1492.

[54] Babuška, I. and Guo, B., 2001, “Direct and inverse approximation theorems for the p-version of the finite element method in the framework of weighted Besov spaces. Part I: Approximability of functions in the weighted Besov spaces,” SIAM J. Numer. Anal., **39**(5), pp. 1512-1538.

[55] Dorr, M.R., 1984, “The Approximation Theory for the *p*-Version of the Finite Element Method,” SIAM J. Numer. Anal., **21**(6), pp. 1181–1207.

[56] Dorr, M.R., 1986, “The Approximation of Solutions of Elliptic Boundary-Value Problems via the *p*-version of the Finite Element Method,” SIAM J. Numer. Anal., **23**(1), pp. 58–77.

[57] Babuška, I. Banerjee, U., and Osborn J.E., 2002, “On principles for the selection of shape functions for the Generalized Finite Element Method,” Comput. Meth. Apple. Mech. Engrg., **191**(49-50), pp. 5595-5629.

[58] Devine, K.D. and Flaherty, J.E., 1996, “Parallel adaptive *hp*-refinement techniques for conservation laws,” Apple. Numer. Math., **20**(4), pp. 367-384.

[59] Babuška, I. and Suri, M., 1994, “The *p* and *h-p* Versions of the Finite Element Method, Basic Principles and Properties,” SIAM Rev. **36**(4), pp. 578-632.

[60] Demkowicz, L., Pardo, D. and Rachowicz, W., 2002, “3D *hp*-Adaptive Finite Element Package (3Dhp90), Version 2.0: The Ultimate (?) Data for Three-Dimensional, Anisotropic *hp* Refinements,” Technical Report No. 02-24, TICAM, University of Texas at Austin, Austin, TX.

[61] Schwab, C., 1998, *p and hp-Finite Element Methods*, Clarendon Press, Oxford, UK.

[62] Rachowicz, W., Pardo, D., and Demkowicz, L., 2006, "Fully automatic *hp*-adaptivity in three dimensions," Comput. Meth. Apple. Mech. Engrg., **195**(37-40), pp. 4816-4842.

[63] Oden, J.T., Demkowicz, L., Rachowicz, W., and Westermann, T.A., 1989, "Toward a universal h-p adaptive finite element strategy, part 2. A posteriori error estimation," Comput. Meth. Apple. Mech. Engrg., **77**(1-2), pp. 113-180.

[64] Zienkiewicz, O.C. and Zhu, J.Z., 1987, "A simple error estimator and adaptive procedure for practical engineering analysis," Int. J. Numer. Meth. Engrg., **24**(2), pp. 337-357.

[65] Yserentant, H., 1992, "Hierarchical bases," *Proc. ICIAM 1991*, SIAM, Philadelphia, PA.

[66] Krysl, P., Grinspun, E., and Schröder, P., 2003, "Natural hierarchical refinement for finite element methods," Int. J. Numer. Meth. Engrg., **56**(8), pp. 1109-1124.

[67] Von Herzen, B. and Barr, A., 1987, "Accurate triangulations of deformed, intersecting surfaces," *Proc. ACM SIGGRAPH'87*, **21**(4), pp. 103–110.

[68] Samet, H., 1995, *Applications of Spatial Data Structures: Computer Graphics, Imaging, and GIS*, Addison-Wesley, Reading, MA, 2nd ed.

[69] Hughes, T.J.R., 2000, *The Finite Element Method: Linear static and dynamic analysis*, Dover Publications, Mineola, NY, 2nd ed.

[70] Booch, G., Jacobson, I., and Rumbaugh, J., 1995, *Unified Modeling Language for Object-Oriented Development*. Documentation Set, Version 0.91, Rational Software Corporation, Santa Clara, CA.

[71] Beall, M.W. and Shephard, M.S., 1999, "An object-oriented framework for reliable numerical simulation," Engineering with Computers, **15**(1), pp. 61–72.

[72] Zienkiewicz, O.C. and Taylor, R.L., 2000, *The finite element method: The basis, Volume 1*, Elsevier Butterworth-Heinemann, Oxford, UK, 5th ed.

[73] ANSYS User's Manual, 1992.

[74] Alleyne, D. and Cawley, P., 1991, "A two-dimensional Fourier transform method for measurement of propagating multimode signals," J. Acoust. Soc. Am., **89**(3), pp. 1159-1168.

[75] Moser, F., Jacobs, L.J., and Qu, J., 1999, "Modeling elastic wave propagation in waveguides with the finite element method," NDT&E Int., **32**(4), pp. 225-234.

[76] McNamara, J.D., 2003, "Health Monitoring of Rail Road Tracks by Elastic Waves based Non Destructive Testing," Ph.D. Dissertation, University of California, San Diego, San Diego, CA.

[77] Krysl, P., Cranford, T.W., Wiggins, S.M., and Hildebrand, J.A., 2006, "Simulating the effect of high-intensity sound on cetaceans: Modeling approach and a case study for Cuvier's beaked whale (*Ziphius cavirostris*)," J. Acoust. Soc. Am., **120**(4), pp. 2328-2339.

[78] Majumdar, A., Birnbaum, A., Choi, D.J., Trivedi, A., Warfield, S.K., Baldridge, K., and Krysl, P., 2005, "A Dynamic Data Driven Grid System for Intra-operative Image Guided Neurosurgery," *Lecture Notes in Computer Science: Computational Science - ICCS 2005*, Springer, Berlin, **3515**, pp. 672-679.

[79] Krysl, P., Damaser, M., Chukkapalli, G., Majumdar, A., Choi, D.J., Trivedi, A., Warfield, S.K., and Hoyte, L., 2006, "Computational Model of Levator Ani Muscle Stretch during Natural Birth," *5th World Congress of Biomechanics*, Munich, Germany.

[80] Lanza di Scalea, F. and McNamara, J., 2003, "Ultrasonic NDE of railroad tracks: Air-coupled cross-sectional inspection and long range inspection," *INSIGHT – Non-Destructive Testing and Condition Monitoring*, **45**(6), pp. 394-401.

[81] West, M., Kane, C., Marsden, J.E., and Ortiz, M., 1999, "Variational integrators, the Newmark scheme, and dissipative systems," *International Conference on Differential Equations*, B. Fiedler et al., eds., Berlin, pp. 1009-1011.

[82] Moore, G.E., 1965, "Cramming more components onto integrated circuits," *Electronics*, **38**(8).

[83] Hauth, M., Grob, J., and Straber, W., 2003, "Interactive physically based solid dynamics," *Eurographics/SIGGRAPH Symposium on Computer Animation.*

[84] Lew, A., Marsden, J., Ortiz, M., and West, M. "Asynchronous variational time integrators," *Archive for Rational Mechanics and Analysis*, **167**(2), pp. 85-146.

[85] Warfield, S.K., Talos, F., Tei, A., Bharatha, A., Nabavi, A., Ferrant, M., Black, P.M., Jolesz, F.A., and Kikinis, R., 2002, "Real time registration of volumetric brain MRI by biomechanical simulation of deformation during image guided neurosurgery," *J. Computing and Visualization in Science*, **5**, pp. 3-11.

[86] Melenk, J.M. and Babuška I., "The Partition of Unity Finite Element Method" *Comp. Meth. Appl. Mech. Eng.*, **152**, pp. 73-84.

[87] Simo, J.C., Tarnow, N., and Wong, K.K., 1992, "Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics," *Comp. Meth. Appl. Mech. Eng.*, **100**(1), pp. 63–116.

[88] Ehlers, W., Ammann, M., Diebels, S., 2002, "h-adaptive FE methods applied to single and multiphase problems," *Int. J. Num. Meth. Eng.*, **54**(2), pp. 219-239.

[89] Felippa, C.A., 2007, *Notes on Mass Matrix* http://www.colorado.edu/engineering/CAS/courses.d/IFEM.d/

[90] Felippa, C.A., 2006, "Construction of customized mass stiffness pairs using Templates," *J. Aerospace Engineering*, **19**(4), pp. 241-258.

[91] Archer, J.S., 1963, "Consistent mass matrix for distributed mass systems," *J. Str. Div. Proc. ASCE*, **89**, pp. 161–178.

[92] Archer, J.S., 1965, "Consistent mass matrix formulation for structural analysis using finite element techniques," *AIAA J.*, **3**, pp. 1910–1918.

[93] Melosh, R.J., 1962, "Development of the stiffness method to define bounds on the elastic behavior of structures," Ph.D. thesis, University of Washington, Seattle, WA.

[94] Hinton, E., Rock, T., and Zienkiewicz, O.C., 1976, "A Note on Mass Lumping and Related Processes in the Finite Element Method," *Earthquake Engineering and Structural Dynamics*, **4**, pp. 245-249.

[95] Goudreau, G.L., 1970, "Evaluation of Discrete Methods for the linear Dynamic Response of Elastic and Visco elastic solids," *UC SESM Report 69-15*, University of California, Berkeley.

[96] Gaveric, L., 1993, "Dispersion Curves of I-Profiled Beams by a Finite Element Method," *Proc. ACOUSTICS*, **15(**3), pp. 553-560.

[97] Wilcox, P., Evans, M., Diligent, O., Lowe, M., and Cawley, P., 2002, "Dispersion and Excitability of Guided Acoustic Waves in Isotropic Beams with Arbitrary Cross Section," *Review of Quantitative Nondestructive Evaluation*, **21**, pp. 203-210.

[98] Lowe, M., 1998, "Characteristics of the Reflection of Lamb waves from Defects in Plates and Pipes," *Review of Progress in Quantitative Nondestructive Evaluation*, **17**, pp. 113-120.