# Adaptive Mesh Refinement Data Visualization

Submitted to:
U.S. Department of Energy, Office of Science Notice 01-01: Annual Notice, Continuation of Availability of Grants and Cooperative Agreements, Advanced Scientific Computing Research, Mathematical, Information, and Computational Sciences

For the period: 1 Oct 2001 through 30 Sept 2004

*Principal Investigator:* Edward W. (Wes) Bethel, LBNL
*Co-Principal Investigator:* Terry J. Ligocki, LBNL
*Co-Principal Investigator:* Bernd Hamann, UC Davis and LBNL/NERSC

## 1 Executive Summary

Our proposal is concerned with the development of novel visualization and exploration techniques for Adaptive Mesh Refinement (AMR) data. AMR-based simulation techniques have gained substantial popularity in the computational science and engineering communities. This technology supports automatic adaptive refinement of meshes when phenomena are to be simulated at vastly differing magnitudes of scale, i.e., when extremely small-scale features must be captured with large-scale features [Berger]. The visualization community to date has done little research supporting the visualization and exploration of AMR data. About one year ago, NERSC's Visualization Group initiated joint efforts with two groups at LBNL/NERSC that investigate computational methods using AMR[1], and with faculty and students at UC Davis[2] to develop innovative AMR visualization techniques. The proposed effort is a direct extension of this ongoing collaboration.

The AMR structure, i.e., connectivity between mesh elements (vertices and cells), is rather simple: typically, each level of an AMR hierarchy consists of multiple structured, rectilinear meshes. It is therefore highly desirable to take advantage of this structure, and to develop specialized - and thus highly efficient - visualization and exploration methods for AMR data. To date, the visualization community has developed few methods that support direct and efficient visualization of AMR data. Our effort brings together the right teams to collaboratively develop dedicated AMR data visualization technology: (i) "producers" of AMR data (NERSC computational scientists and engineers, ANAG and CCSE) and (ii) visualization researchers and technology developers (LBNL Visualization and UC Davis Visualization and Graphics Research Groups). AMR data producers will help this effort by identifying specific needs for data analysis and evaluating our approaches and systems.

AMR data can simultaneously capture multiple scalar and vector properties, in addition to material boundaries (that are implicitly defined by certain fractional material information). The major objective of this proposal is to design, implement, and test highly specialized and efficient, error-controlled visualization techniques for very large AMR data. We will place special emphasis on the development of (i) novel volume visualization methods (ray casting, cell projection, "splatting"); (ii) feature recognition and tracking technology for time-varying AMR data; (iii) hardware-supported and -accelerated volume texturing techniques; and (iv) interactive methods supporting localized exploration - where local exploration may be understood as exploring a certain region in space, or exploring a specific level in the AMR hierarchy. In addition, we will develop robust and high-quality material-boundary approximation and rendering techniques, which will support the visual incorporation of extracted boundary information in scalar and vector

[1] Applied Numerical Algorithms Group (ANAG) led by Phil Colella, and the Center for Computational Sciences and Engineering, led by John Bell.
[2] UC Davis Visualization and Graphics Research Group

field visualizations. Another objective is the approximation and compression of AMR data using hierarchical B-splines and B-spline wavelets.

# 2 Technical Topics

The scope of this proposal is upon novel, efficient and usable techniques for visualization of AMR data. Based upon our prior experience and expertise, the proposed effort spans multiple related topics. First, the AMR grids pose special challenges for visualization. The fundamental challenge stems from the fact that AMR uses multiple, hierarchical grids that spatially overlap. Our approach for visualization of scalar data from AMR grids builds upon previous work by focusing on AMR-centric approaches. Visualization of vector field AMR data, particularly those techniques that use iterative solvers, must dynamically adjust integration step size to account for the varying size and hierarchical nature of AMR grids. In time varying AMR data, the time dimension is also adaptive during AMR computation, but is not stored in adaptive form. AMR provides the means to perform highly resolved computations in local regions of interest without incurring the expense of representing the entire problem domain at the finest level of resolution. We propose to take this form of compaction one step further by using B-spline basis functions as an alternative to uniform Cartesian grids. We discuss methods of processing and visualization of large data sets, with a set of approaches that use out-of-core and progressive techniques.

We have arranged our discussion of technical topics using a format that covers each individual topic in a sequential fashion. Each topic area begins with an introduction of the problem, continues with our approach to research, and concludes with discussion of expected results. All results are summarized in a three-year work plan, which appears in a subsequent section.

## 2.1 AMR Grids

The hierarchical data structures of AMR don't lend themselves to direct representation in the data models provided in traditional visualization packages. Prior work in AMR visualization has focused on grid-conversion algorithms that enable the reuse of existing software. Such approaches present unique technical challenges, which we discuss below. As an alternative, we propose a novel method of AMR grid representation specifically for use in visualization. Our overall approach will be two-fold: to explore new alternatives in the area of grid conversion and to break new ground in the area of visualization software that can directly process AMR data without the need for grid conversion.

### 2.1.1 Background and Motivation

By and large, the hierarchical and spatially overlapping grids of AMR don't lend themselves to use by traditional visualization algorithms. These traditional algorithms typically operate on a single structured grid or array of unstructured cells. A variety of techniques have been employed in the past to convert AMR data into a form suitable for use by traditional visualization algorithms [Norman]. One approach is to "flatten" the AMR hierarchy into a single uniform grid. Another is to convert all the cells in all the AMR grids into hexahedral cells of an unstructured grid. Yet another is to independently process all the uniform grids in the hierarchy. Each of these techniques has drawbacks.

When AMR data is flattened into a single uniform grid, the size of the data can grow immensely, especially in the case of AMR data with many levels of refinement. This happens because the individual cells of the new grid are the same size as the smallest cells in the AMR data. In addition, when the grid is flattened, the original data structures are lost: the data present in coarser cells is mapped onto multiple cells in the new grid, and multiple data values per cell must be resolved.

When AMR data is converted to an unstructured grid of hexahedral cells, the number of cells generated can be quite large, and connectivity information for all the cells must be stored

explicitly. The result is an inflation of data sets that may already be quite large. More importantly, visualization algorithms that operate on unstructured grids are much slower than the equivalent algorithms that operate on uniform grids (that compose the original AMR data). The latter algorithms, which operate on structured grids, can take advantage of the implicit, uniform connectivity of the cells composing the grid.

### 2.1.2 Iterative Approach to Grid Processing

In our initial investigations, we have experimented with an approach in which the visualization software iteratively processes each AMR grid in a sequential fashion. This approach has the advantage that no data is lost and no data explosion occurs due to flattening. More importantly, the original data is preserved throughout the visualization process, rather than being lost as part of a grid conversion process.

We have performed some preliminary implementations by extending a traditional visualization tool to process a variable number of uniform grids and to keep track of the metadata associated with the original AMR data. An example of this is ChomboVis [ChomboVis1, ChomboVis2, ChomboVis3] which was developed to support an AMR library, Chombo [Chombo], developed by the Applied Numerical Algorithms Group [ANAG] at LBNL/NERSC.

We have observed several problems using the iterative grid processing approach. First, there exists the potential for discontinuities along grid boundaries. In isosurfaces, these anomalies are most often visible as "cracks" or "seams". Second, there exists the potential for spatial overlap of visualization results when hierarchical grids occupy the same region of space (e.g., one grid contains several others). Finally, the notion of multiple overlapping grids is somewhat meaningless for some types of visualization, such as volume rendering.

### 2.1.3 Novel Approach to Grid Conversion

A better approach is to use new visualization algorithms that can operate directly on the AMR data. By doing so, none of the structure in the original data is lost. This is especially important when researchers are developing and debugging new AMR computations. These new visualization algorithms can realize the same types of efficiency gains as other AMR computations by taking advantage of the uniform grids that compose the AMR data. The challenge is defining and developing such visualization algorithms.

UC Davis visualization researchers and the LBNL Visualization Group have begun investigating these challenges. In the AMR data computed by ANAG, the data values are defined at discrete points in space and may have multiple values at these points. For many visualization algorithms, such a discrete and multivalued representation must be converted using some form of interpolation to a single-valued representation that is defined over the entire computational domain. In the case of single, uniform grids, trilinear interpolation is often used.

One approach we have been developing is called “grid stitching.” In grid stitching, we create a new single grid using data from the highest resolution grids and remove any portions of the lower resolution grids where there is overlap. The voids between these two regions are reconstructed using a “stitching algorithm.” The resulting grid contains only values from the original data – no interpolation is needed. Grid stitching produces grids that don't overlap and which elegantly accommodate regions of different resolutions. Using grid stitching as a basis of representation, we have developed several customized visualization techniques (e.g., isosurfacing and volume rendering) to create images of AMR data. Initial results of this work are presented in [Weber1, Weber2, and Weber3]. Grid stitching is a novel approach that addresses the difficulties inherent in AMR visualization. The promising initial results merit further examination and broader application.

### 2.1.4 Objective and Methods to be Developed

During the first year, we will be developing a robust grid stitching algorithm. Concurrently, we will develop a tool for generating crack-free isosurfaces from stitched grids. Grid stitching is the cornerstone upon which other visualization techniques are built. These are discussed in greater detail in the balance of this proposal.

## 2.2 Scalar AMR Visualization

Building upon the discussion of the previous section, we now turn our attention to the issues of scalar AMR data visualization. In this section, we discuss several approaches to AMR data volume rendering. Our approaches pursue topics in software-only methods, in hardware-accelerated methods, and hybrid adaptive methods to reduce graphics load.

### 2.2.1 Background and Motivation

Volume visualization is often compared to a "digital X-ray": images created using volume rendering are constructed by sampling data along rays, accumulating color and opacity. One of the keys to effective volume rendering is selection of a “good” color table, which includes colors, transparency, and a transfer function that maps scalar data into color. Whereas isosurfaces and cut planes are topologically two dimensional, volume rendering is truly 3D in the sense that no data is thrown away while creating the image. The images created by volume rendering are extremely valuable to scientific researchers, and are often considered an essential part of a portfolio of visualization techniques. Volume rendering is an expensive process, however, and there has been much prior work in the field of accelerating or optimizing volume rendering, as well as improving the visual fidelity of the final images.

Most AMR volume visualization algorithms are derived from existing techniques applied to structured and unstructured meshes. However, the hurdles they must overcome are how to efficiently deal with the overlapping cells as well as how to properly handle the visual artifacts that result from the discontinuities that occur at coarse-fine mesh boundaries. Naive application of any volume rendering technique to AMR data will result in the refined regions having pronounced discontinuities in opacity. This is because the composite of the projection of the refined grid over a coarse grid doubles the apparent opacity per unit volume in the regions where the grids overlap. Even techniques that are able to eliminate these overlapping regions are still confounded by the same boundary discontinuities that create cracks in isosurfaces (as described in the previous section). The unique features of AMR data sets require us to explore more sophisticated data management and line integral interpolation techniques.

### 2.2.2 Prior Work

There has been some seminal work on volume visualization of AMR data by other groups, but it is a largely undeveloped area of research. Some researchers have employed techniques that treat the AMR data in its native form (e.g., without flattening). Kwan-Liu Ma has implemented a massively parallel software ray tracer on the T3E using scalable cell-projection volume rendering techniques on data pre-formatted to eliminate regions of overlapping cells [Ma]. The team of Donna Cox and Bob Patterson have employed the Pixar Star-Renderer to create high quality volume rendered images suitable for HDTV production for the PBS series "Mysteries of the Universe” [Cox]. In this work, the data cells were filtered to eliminate overlaps. Then each cell in the AMR data set was replaced with a Gaussian splat 1.33 times the size of the cell. The Star-Renderer draws such splats extremely fast. While this technique produces very high quality imagery, it is not sufficiently accurate as a technique to support scientific investigation.

Other groups flatten the AMR hierarchy into a single uniform grid. While this method leads to loss of fidelity in the final image, it enables use of the large number of existing algorithms for regular grids that have well understood behavior and performance. John Bell's group at LBNL/NERSC uses a serial software shear-warp on AMR data that has been flattened to a uniform grid. Likewise Sarah Anderson of the Laboratory for Computational Science and Engineering at

University of Minnesota samples data into uniform grids in order to volume render time sequences of 1024^3 volumes in near-real-time using SGI Reality Monster multiple-pipe hardware rendering solutions[3]. Bethel uses a combination of Image Based Rendering (IBR) techniques, parallelism in software volume rendering, and pipelining to create Visapult, a unique architecture for remote volume rendering [Bethel]. Visapult has the attractive properties that it scales well, and effectively decouples parallax interactivity from the latency inherent in network-based applications.

Research topics for AMR scalar data volume visualization include methods for parallel, software-only methods, parallel and hardware-accelerated methods, and methods for combining volumetric rendering with geometric data, such as that produced by traditional visualization (isosurfaces, cut planes, etc.). The techniques provide a continuum of speed vs. visual fidelity with a wide range of resource requirements.

### 2.2.3 Software Techniques: Ray Casting Methods

In traditional ray casting, at least one ray is cast into the volume for every pixel in the resulting image (most often ray-bundles are used to provide some oversampling to improve the quality of the results). Along the ray segment that penetrates the volume, the cell data is typically sampled using a fixed step size accumulator, and color and opacity (and sometimes radiance) are integrated using alpha blending. Ray casting through rectilinear data exploits the regular structure of 3D arrays in memory to greatly accelerate performance. However, the hierarchical structure of AMR data prevents naïve direct application of this approach.

We propose to extend ray casting to render AMR data in native form. One of the fundamental problems to be solved is dynamic adjustment of the step size as a function of the grid resolution. The integrator must detect when it has crossed into a region of different refinement so that it can adjust its step size to half the cell size of the region it has entered, and also modify its coefficient of integration. Performing such a conditional check at each iteration will slow the algorithm substantially, since intersection tests with a large number of bounding boxes must be performed at each step. Also, conditional statements can be particularly inefficient on modern microprocessor architectures, as the pipeline “bubbles” caused by mispredicted branches are extremely costly and cause the processor to stall for many cycles. One must pre-compute all of the bounding box intersections for the line integral, and compute the line integral for each of these sections independently. Subsequently, the segments are reassembled in a separate compositing operation. In this way, tests for intersection with the refined regions can be moved outside of the innermost loop, thereby greatly increasing performance. Space partitioning trees and other techniques can be employed to speed the performance of box-intersection searches.

Finally, many (but apparently not all) AMR frameworks require that refined grids must provide at least once cell of buffering between the refined children and the boundary that intersects a coarse parent. From the standpoint of the ray, this means you can never descend more than one level of the hierarchy at a time along the integral's path. Exploiting this feature can further reduce the number of grids that need to be examined while pre-computing the bounding box intersections.

The data in AMR grids associated with finite-difference calculations is defined only at cell centers. Since the cast rays will not pass directly though the cell centers, we need a representation for the data values at every point in-between the discrete locations where the data is defined. The interpolation function used to construct the data values in these open spaces can be described in terms of Piecewise Continuity. C0 continuity is simply uses the value from the cell throughout the entire cell. C1 continuity is a piecewise linear interpolation of value using a 6-point distance-weighted stencil of the nearest neighboring cells. C2 continuity implies a higher-order interpolation function between the neighboring cells, but this interpolation function should match the basis function used by the numerical simulation to be accurate. Even then, there may still be

[3] This work was a demonstration project at SC98, and has no formal reference.

visible discontinuities at boundaries that are normally compensated for within the AMR numerics that use complex flux-correction methods. The application of flux correction methods and hierarchical higher order (C2) smoothing to these line integrals is an entirely unexplored area of inquiry.

Early experiments by Gunther Weber this past summer[4] using C1 smooth representations of the volume display significant boundary artifacts when viewed in certain orientations. We will evaluate higher order interpolation functions to minimize artifacts during discrete sampling when ray casting.

### 2.2.4 Software Methods: Cell Projection

During ray casting, intersections between the ray and cells are computed as the integrator steps along the portion of the ray that penetrates the volume. In the cell projection method, the line integral representing a ray's penetration through an individual cell is computed for each cell. The resulting values are accumulated along a ray to produce a final pixel value.

Cell projection methods are often employed for volume rendering of unstructured data sets. The 3D cells are projected into screen space and line integrals through the cell are computed for every pixel the cell covers. Segments can be composited either while they are being computed, or after all of the segments have been calculated. Likewise, the overlaps between refined and unrefined cells can be eliminated either before the integral is computed, just as with the fixed-step-size ray casting. Alternatively the ray fragments can be sorted after all of the cell segment integrals have been computed in order to eliminate redundant segments. While the former might be computationally less expensive, the latter offers us a method of rapidly adding and removing the contributions of different levels of the AMR hierarchy, providing a framework for progressive update.

Cell projection methods suffer from the same discontinuities as described above for the fixed-step-size ray tracing techniques, and for the same reasons. However, it has been suggested that applying the grid stitching technique used for crack-free-isosurfaces to cell-projection based volume rendering, and using a linear interpolation (C1 smooth), might entirely eliminate these artifacts. To do this, we intend to use Nelson Max's fast cell projection ray tracer as a basis for this methodology.

### 2.2.5 Hardware Methods: Hardware Textures with Stencils

The traditional technique used to implement volume rendering with 3D hardware acceleration is to load the volume data up as a 3D texture, or a stack of 2D texture mapped slices. These textures are applied to a stack of simple polygons. The alpha blending operator provided by the graphics hardware performs the same integration used in ray casting, but at much faster rates. This approach limits the amount of data displayed to the size of available texture memory, which is typically on the order of 64 megabytes, or requires texture paging, which severely limits performance. In order to display data that is larger than available texture memory, data size must be reduced by combining some number of adjacent slices with volume rendering, and using the result as a single slice of texture data. Visapult [Bethel] uses exactly this technique.

The demands of the video-gaming market have led to a strong emphasis on texture-mapping capabilities in the PC-desktop video cards market, including the ability to apply more than one texture simultaneously to a single polygon (multi-texturing). Some of these cards can even apply multiple textures with almost the same performance as only applying one. It is an interesting avenue of experimentation to see if this capability can be exploited to reduce the framebuffer memory bandwidth bottleneck. These rapid advances in texturing performance in the consumer market provide strong impetus to investigate texture-based techniques.

[4] Weber is a UC Davis graduate student who spent the summer of 2000 at LBL with the Visualization Group.

We envision two approaches that improve upon the traditional methods. In both cases, the volumes are drawn using the traditional texture slicing approach for each of the grids that comprise the hierarchy. The goal is to eliminate regions of overlap where the slices for the refined grids overlap those of the coarse grids. The first approach is pre-process the slice textures to create transparent windows in them where they overlap refined slices from refined grids. This is accomplished by zeroing the alpha values in regions of overlap prior to uploading the texture to the graphics card. The second approach is essentially a multipass technique that uses the stencil buffer combined with texturing. In this approach, we begin by considering a slice of the coarsest grid. Regions of the coarse grid will be overlapped by slices of the refined grid that exist on the same slice plane. So we project each of these refined slices into screen space and rasterize them into the stencil buffer. Then when the coarse slice is drawn and textured, the stencil buffer will cut out the regions where it is overlapped by the refined slices. This approach has the drawback of placing a heavy load on stencil buffer operations.

### 2.2.6 Hardware Methods: Stripped Textures

Another avenue of future research is a modification of the texture-mapping approach to eliminate regions of overlap by subdividing the volume into smaller blocks, so that sub-blocks that overlap can be thrown out while still retaining the overall hierarchy and block-structuring. This eliminates the overlapping regions by throwing out the actual overlapping data, rather than creating windows in the textures where these overlaps exist. This would greatly reduce superfluous usage of texture memory, but at a cost of preprocessing the grids (using the host CPU) to slice the AMR grids into these strips. One possible problem with this approach is that it is less efficient to load a number of small textures than it is to load a single large texture. Techniques for packing multiple such smaller texture "strips" into a larger texture may be required to maximize efficiency.

### 2.2.7 Hardware Methods: Polygonal Representation of Cells

In this approach, each cell of the volume is drawn using three semi-transparent polygons. The colors of each polygon vertex are a function of data, color table and transfer function. This technique has been used successfully as a volume data previewer. In modern graphics architectures, however, we are more concerned with bandwidth to the graphics hardware than with transformation and lighting operations. The host adapter bandwidth required when sending this number of polygons to the graphics card is considerably greater than that required for textures or for surface geometry. However, such bandwidth consumption should be balanced against the texture-mapping performance of the given graphics card. We also expect to encounter fill limitations, due to the significant number of overlapping transparent polygons. In addition, the polygons must be dispatched in depth order to the graphics hardware, which adds somewhat to the computational burden on the host processor.

### 2.2.8 Other Methods: View-dependent Adaptive Approaches

Most cells in the deeper levels of an AMR hierarchy map to less than a single pixel on the screen when displayed in an overall view of the data set. When zooming in for closer examination of the refined levels, individual refined cells become visible and the majority of cells in the surrounding coarse grids become clipped. A view-dependent adaptive strategy would attempt to volume render only the cells that lie within the viewport, and also project to greater than a single pixel in resolution. Such adaptation, when applied to the polygonal approach, can greatly reduce the graphics load by simply drawing fewer cells. This is another avenue of investigation. This technique will also have applications to rendering of polygonal geometries derived directly from the cell data (like slices and isosurfaces).

### 2.2.9 Objectives and Methods to be Developed

During the first year, we will develop a technique for volume rendering native AMR data with ray casting. We will develop and evaluate methods for hardware accelerated volume rendering that employ stripped textures and stenciling. Texture images for refined regions in stripped textures and stenciling will be generated using IBR techniques.

In the second year, we will develop a cell-projection method of volume rendering that uses stitched grids. By taking into account view-dependent information, we will improve the performance of the cell-projection method. Performance of the ray casting volume renderer will be improved by including space-partitioning techniques and novel methods for grid indexing and intersection testing.

In the third year, we will experiment with increasing performance of the basic cell-projection technique by migrating cell segment alpha compositing into graphics hardware. Visual fidelity of the ray casting will be improved through integration of flux-correction data from AMR codes in order to reduce/eliminate discontinuities at coarse/fine grid boundaries.

## 2.3 Vector Field and Time Varying AMR Visualization with Feature Tracking

Traditional vector field visualization challenges are exacerbated by the multiple, hierarchical grids present in AMR data. Ultimately we would like to allow CFD researchers identify features present in AMR data sets and track these features through time. This will require extending indirect vector field visualization techniques (e.g., streamline generation) to AMR data so that features can be identified. Tracking the feature accurately through time will then require computing and storing additional AMR data during the computation. This will require working closely with the researchers doing AMR computations and may require a reexamination of AMR data storage. We have a close collaboration with two groups that develop AMR codes for CFD computations and are in a good position to explore these problems.

### 2.3.1 Background and Motivation

The usual difficulties associated with vector field visualization are further compounded in AMR due to multiple hierarchical grids that spatially overlap. While it is possible to employ the "standard" direct visualization techniques, such as hedgehogs, to AMR vector fields, the usual difficulties encountered with those methods (e.g., visual clutter) are amplified by the presence of nested and hierarchical AMR grids. With direct techniques, the grids can be treated independently or coarser data can be suppressed where finer data is present.

Indirect vector field visualization techniques (e.g., particle advection, integral curve generation, and topology extraction) require a single-valued vector field defined over the entire computational domain. To achieve this the vector fields defined on each grid must be combined (e.g., stitched) to effectively form a single vector field. Once this is done, indirect vector field visualization techniques, such as streamline generation, can be adapted to compute a path through the vector field. In this class of techniques, the integration step size is sensitive to the spatial dimensions of the underlying cells. Dynamically and efficiently computing a step size that is sensitive to cell size is an open research topic.

Using indirect vector visualization techniques, features in the AMR data can be identified by researchers. They would then like to track these features through time varying AMR data. Time varying AMR data further compounds the challenges presented thus far for scalar and vector fields. During the AMR computation, the grids may be repositioned over time to ensure numerical stability. In addition, the time step used for grids at different resolutions varies. In other words, AMR grids in the computation are adaptive in the time dimension. So, time varying AMR computations with two spatial dimensions have a third time dimension, and can be viewed as three dimensional AMR data sets (adaptive in all three dimensions). Similarly, time varying AMR

computations with three spatial dimensions form AMR data sets of four dimensions. However, AMR data is not stored in this fashion. Therefore, a certain amount of temporal data is lost when the AMR file is written. This makes it difficult or impossible to accurately visualize time varying AMR data and track features through time.

### 2.3.2 Approach

We feel that one of the most challenging problems in vector field visualization of AMR data is extending indirect techniques, such as streamline generation, particle advection, and topology extraction. Our approach is to first create a single-valued representation defined over the entire computational domain from the discrete, multivalued AMR vector data. We achieve a single-value representation in three steps. First, we take the AMR vector data and divide it into its scalar components. This step creates one AMR scalar data set for each component of the vector field. Second, each scalar component undergoes scalar interpolation, at which time the grids are stitched into one. Third, scalar data values from the separate stitched grids, which are topologically identical, are simply combined to form a single-valued representation of the original AMR vector data defined over the entire computational domain. Unfortunately, this approach isn't always valid due to characteristics of vectors that are not represented in a collection of scalar fields. For example, linear interpolation of the scalar components would not preserve the unit vector lengths in a normalized vector field. In most cases, important vector characteristics can be preserved if special characteristics are taken into account.

Once a single-valued representation of the vector AMR data has been obtained, then established indirect vector field visualization techniques can be modified and applied. In the case of techniques that integrate a path through a vector field, the integration set size must take into account the size of the cells composing the stitched grid to obtain accurate results. When the path leaves one cell and enters the next, if the new cell has a different size then the path needs to be stopped at the cell boundary and continued from there with a new step size. One integration step could also cause the path to pass through many finer resolution cells. Detecting this and controlling the integration step size to account for varying cell sizes in an efficient manner is an important and open research issue.

Using the techniques discussed above, streamlines of vector field AMR data can be computed accurately. One of the groups we work with would like to use these *streamlines of vorticity* (the mathematical curl of the velocity) to visualize vorticity tubes in 3D AMR data. A vorticity tube is generated by taking a closed curve and generating a streamline of vorticity through each point on the curve. Thus, the closed curve is swept though space to create a tube. Researchers would then like to track vorticity tubes through time.

Working with time varying AMR data will require either modifying the AMR computations themselves or modifying the way AMR data is stored. Either approach addresses the same problem. Namely, not enough information is currently stored from a time varying AMR computation to accurately compute paths through vector fields in time and space. Specifically, not enough time steps are stored. We propose to investigate both approaches in the limited context of tracking vorticity tubes. The researchers we work with have suggested computing an additional vector field that will allow the vorticity tubes, once identified, to be moved through time and thus tracked. This uses a special property of streamlines of vorticity. Namely, they are "frozen in the fluid" which simply means they travel with the fluid as it moves. For less specialized problems it will be necessary to store more data from the AMR computation. Specifically, the AMR data will need to be stored at different time scales for different resolution grids. More time steps will need to stored for finer grids than for coarser grids, and enough will need to be stored in all cases to ensure that paths can be accurately traced through time varying AMR vector fields. This will result in data that is adaptive in time and space. Visualizing this type of data is an open research problem.

### 2.3.3 Objectives and Methods to be Developed

In the first year, we will develop techniques for representing discrete, multivalued AMR vector data as a single-valued vector field defined over the entire computational domain, develop techniques for computing integral curves from AMR data, and for computing and rendering vorticity tubes from time varying AMR data.

In the second year, we will collaborate with AMR developers to store time varying AMR data with time as one of the dimensions of the AMR data. We will develop parallel versions of the integral curve and vorticity tube calculations. We will also track vorticity tubes through time varying AMR vector data. Working with ANAG at LBNL/NERSC, we will explore additional "features" and define ways to "track" them over time.

In the third year, we will examine features in vector data relevant to other scientific domains, such as cosmology. In addition, we will consider feature tracking as the sweeping of a feature through time. For example, a curve resulting from 2D AMR data would sweep out a surface as it is tracked through time. Viewing this surface or something derived from this surface might give information about the feature and it's evolution.

## 2.4 Embedded Boundaries

Some CFD computations take into account not only the fluid and its movement but also the presence of bodies that occupy regions of space and contain no fluid. The boundary between these bodies and the balance of the computational domain is represented indirectly in AMR data produced by the CFD computations. Visualization of these boundaries, called "embedded boundaries," (EB) is challenging due to the lack of explicit information about the boundary: the boundary representation must be computed from implicit information. We propose to approach the problem of embedded boundary visualization by extending visualization tools to take into account the EB information.

### 2.4.1 Background and Motivation

ANAG at LBNL/NERSC has begun to integrate EBs into their AMR computations. This takes an already difficult visualization problem and makes it harder: the EBs are not represented explicitly as geometry, but are represented implicitly as volume fractions, area fractions, and connectivity to other neighboring volumes.

More specifically, in ANAG's embedded boundary representation, every cell in each grid is marked as regular, irregular, or covered. Regular cells lie entirely within the fluid and contain no part of the embedded boundaries, and covered cells contain no fluid and contain no part of the embedded boundaries. In contrast, an irregular cell implies the presence of one or more fluids, and the presence of one or more embedded boundaries. Each irregular cell contains one or more volumes of fluid (VOF), the volume fraction occupied by each VOF along with the area fraction each VOF shares with VOFs in other irregular cells, the centroid for each VOF, and a single normal vector. If an explicit EB is needed then it must be computed from this implicit information.

### 2.4.2 Approach

We propose to address the visualization of AMR data with EBs using two approaches. In one approach, we compute an explicit EB representation that meets some or all of the constraints imposed by the implicit information present in the irregular cells. Visualization techniques would then be extended to take into account the EB as additional geometry. We see this approach using the following sequence of steps:

1. The generation of explicit boundaries from the implicit information. The difficulty of this problem depends on which of the implicit constraints are met by the explicit representation (including continuity and smoothness at the boundary between cells).

2. The extension of single, uniform grid visualization techniques to single, uniform grids with embedded boundaries.
3. The extension of the above techniques from single, uniform grids with EBs to AMR data with EBs. This will build on other work being done to visualize AMR data without EBs.

The other approach is to use the implicit information directly (e.g., the normal) in the visualization process in order to produce images that contain information about the data, as well as the embedded boundaries. In this approach, only steps (2) and (3) above are necessary, as no explicit boundaries need to be generated.

The process of handling single, uniform grids with EBs and then AMR data with EBs parallels the way ANAG is developing their algorithms. This means we can use their libraries and algorithms to generate test data and can provide them with visualization capabilities. Such interaction strengthens both efforts.

Finally, the basic, overall approach to extending traditional visualization techniques to data with EBs is to proceed with the traditional technique inside regular cells and new, specialized techniques inside irregular cells. For example, a ray casting volume renderer proceeds as usual through regular cells. When an irregular cell is encountered, if the ray penetrates an embedded boundary, the ray would be terminated. A similar approach can be used in particle advection. As the particle is advected, a traditional algorithm would be used while inside regular cells. While inside an irregular cell, the particle continues to move only so long as no boundary is encountered. We expect there to be domain- or problem-specific variations on this basic theme.

### 2.4.3 Objectives and Methods to be Developed

In the first year, we will compute and render explicit embedded boundaries from single grids containing implicit EBs. We will produce volume renderings directly from the same data.

During the second year, we will extend visualization techniques for single grids to single grids containing implicit EBs. We will do this by generating explicit embedded boundaries and also by working directly from the implicit embedded boundary information.

In the third year, we will extend visualization techniques for AMR data sets developed earlier to AMR data sets with EBs. Again, we will do this by generating explicit embedded boundaries and also by working directly from the implicit embedded boundary information.

## 2.5 Hierarchical B-splines and Semi-orthogonal Wavelets for AMR Data

AMR data, in their "proto-typical form," are given as nested rectilinear meshes. In the 2D case, each AMR level is usually defined on a mesh consisting of (often uniform-sized) rectangles; in the 3D case, a level is usually defined on a mesh consisting of (often uniform-sized) hexahedra ("cuboids"). Furthermore, the vertices in any level of an AMR mesh hierarchy defined over an n-dimensional domain have valence 2n (except those vertices lying on the mesh boundary). AMR grids, at any level of resolution, are therefore defining "tensor-product" topologies that can be used directly to define, for example, tensor-product B-spline basis functions.

AMR data, despite the fact that they are constructed adaptively and in a hierarchical fashion, often exhibit large degrees of redundancy. Finer grid structures are imposed in more complicated regions only, but the use of uniform-spaced grids is restrictive and can lead to larger-than-necessary storage requirements. It is our goal to come up with improved and more general "non-uniform" B-spline representations for AMR data that lead to significant compression.

B-splines have been used traditionally, and with great success, for the representation of curves and surfaces. Their potential for volumetric data approximation, such as 3D AMR data, has not

been recognized. In fact, most currently used visualization algorithms are considering only a linear variation (tetrahedra) or trilinear variation (hexahedra) of functions values over a single grid cell. To fully exploit the power of B-splines, higher polynomials should be considered for scientific, especially 3D data approximations. We will therefore also investigate the efficient usage of B-spline-based data representations for the purposes of interactive data visualization and exploration. Hardly any work has been done in this area. One approach is discussed in [LaMar].

Hierarchical B-splines were introduced into the field of multi-level surface editing by Forsey and Bartels in 1988 [Forsey]. A generalized hierarchical B-spline approach can be used for a multiresolution approximation framework for AMR data. Furthermore, such an approach can also be extended to develop semi-orthogonal B-spline wavelets (tensor products) for 2D, 3D, and, possibly, 4D (=3D time-varying) AMR data [Stollnitz]. An entire hierarchical B-spline-based visualization framework can be envisioned supporting level-of-detail, view-dependent, and adaptive rendering. Much of the "mathematical infrastructure" to do this exists, and we propose to extend existing methods for AMR data consisting of strictly rectilinear meshes at each level of resolution.

### 2.5.1 Background

Tensor-product B-spline functions are defined by (i) a polynomial degree and (ii) so-called "knot vectors" in each dimension, see [Farin]. A structured, rectilinear AMR grid automatically defines the knot vectors for B-spline basis functions by the spacing of the grid cells in each dimension. For the purposes of this project, we assume that one constant function value is given for and associated with the center point of each grid cell. Therefore, one AMR grid (each level in it) defines one constant B-spline (a piecewise constant function). It is now possible to use polynomial degree and knot vectors as degrees of freedom to "design" better B-spline representations for each AMR grid (and at each level), i.e., representations of arbitrary, but low, degrees and smaller knot vectors. Adjusting polynomial degree and choosing knot vectors in an "optimal" fashion can lead to substantial compression of B-spline functions, see [Goldman]. We would build on and extend so-called "knot deletion" algorithms to come up with much more compact representations of all sub-grids within a given AMR hierarchy.

Hierarchical B-splines were introduced into multi-level surface editing by Forsey and Bartels in 1988 [Forsey]. A hierarchical B-spline approach can also used for a generalized multiresolution approximation framework for AMR data. To locally edit and refine a simple B-spline curve, for example, the method of Forsey and Bartels inserts additional knots in the area of refinement, and the locally added detail is stored as a local "offset" to the original shape. We plan to develop efficient techniques to also represent the finer AMR levels in an AMR hierarchy as offsets to the coarser representations. We are convinced that great savings in storage can be achieved using such a hierarchical B-spline approach, by optimizing the two degrees of freedom we have: polynomial degree and knot vectors.

Further reduction can be achieved by constructing semi-orthogonal B-spline wavelets for AMR data (see [Stollnitz]). Once "optimal" polynomial degree and knot vectors are identified, we would construct semi-orthogonal B-spline wavelet basis functions to most compactly represent an entire AMR hierarchy.

### 2.5.2 Objectives and Methods to Be Developed

In the first year, we will concentrate on the development of hierarchical B-splines for 2D and, especially, 3D AMR data. The most crucial aspect of this work will be the efficient determination of "good" polynomial degrees and optimized knot vectors to represent the various offset levels in the hierarchical B-spline approximation.

In the second year, we will focus on the construction of semi-orthogonal B-spline wavelets applied to the hierarchical B-spline representations developed in the first year. Furthermore, we will start with the design of interactive visualization and data exploration techniques for volumetric

hierarchical B-spline and volumetric B-spline wavelet representations. At this stage, we will emphasize efficient utilization for 3D slicing, contouring (isosurfacing), and volume rendering.

In the third year, we will emphasize further development of efficient and adaptive visualization methods for B-spline data representations. We will focus on the development of view-dependent and region-of-interest methods, where the pre-computed B-spline representation is to be accessed and traversed in an optimized fashion. We also plan to develop interactive techniques allowing a user to manipulate color transfer functions in such a way that only specific levels in a data hierarchy are considered (or are highly emphasized) in volume visualizations.

## 2.6 Large Data Issues

A persistent problem in scientific computing and visualization is the enormous expansion in data sizes produced by modern simulation codes. This is increasingly true of AMR codes as they become mature and scale up in terms of performance and memory utilization. However, under many circumstances, researchers still demand tools for real time interactive exploration of their data sets. This has spawned research into a variety of out-of-core, hierarchical multiresolution, and progressive-update visualization techniques [Grosso, Westermann, Neubauer]. Much literature has focused on efficient creation of multiresolution data from fixed resolution data sets using a variety of sampling and wavelet-based data reduction techniques [Zhu, Muraki]. The native AMR data sets that are the focus of our research have a natural hierarchical multiresolution structure that lends itself well to out-of-core and progressive processing techniques. However, the techniques developed for native AMR data will have impact on all hierarchical multiresolution visualization strategies.

### 2.6.1 Out-of-core Strategies

Traditional visualization pipeline paradigms load the entire simulation data set into memory at one time and process it through a number of stages. This is simply not possible with large data sets that exceed the memory size of computer systems, hence the necessity to use of out-of-core techniques. These techniques are typically applied to block-structured grids that have been decomposed into page-sized subdomains [Cox], as well as domain-decomposed unstructured grids [Ma]. The block structured AMR grids already possess a natural domain-decomposition that is very suitable for efficient disk loading.

Non-adaptive out-of-core methods can make more efficient use of available memory on a given system via data loading that is controlled by the application. The biggest benefit is that arbitrarily large AMR data sets can be processed on a workstation regardless of memory size. A non-adaptive methodology simply cycles through the AMR grids in round-robin fashion, feeding them through the visualization pipeline and accumulating geometry for interactive viewing. This method has been employed successfully in the ChomboVis2 tool co-developed by ANAG and the LBNL Visualization Group [ChomboVis1, ChomboVis2, ChomboVis3]. We want to expand on this work by incorporating a series of progressively more sophisticated adaptive strategies into the ChomboVis tool.

The central theme of many adaptive out-of core strategies is to load and process grids based on selection criteria that depends only on the AMR grid metadata. For example, some useful and objective criteria would include view dependency, data dependency, and a time budget to maintain interactivity. A visibility test can reduce processing time by loading only those grids that appear within the view frustum. Related, grids that map to sub-pixel resolution could be culled from processing. Some data dependent operations, such as isosurfacing, would load only those grids that contain the threshold value. Finally, the number of grids that are processed could be limited by a time budget, such as the rendering frame rate.

### 2.6.2 Progressive Rendering

Whereas the focus of adaptive loading methods is criteria that eliminate grids entirely from the visualization-processing pipeline, the focus of Progressive Rendering methods is on ranking the grids to define the order in which they are processed. The criteria used for this ranking are essentially the same as for the adaptive loading, but the rank may be constructed from weighted combinations of those criteria. Additional criteria for grid sorting relevant to progressive rendering are refinement and proximity. Refinement as a criteria means that it may be desirable to render coarse grids before refined grids, so coarser grids have a higher ranking. Proximity means that a user or feature-detection system might have identified a point of interest, such as a vortex core. Promixity-based criteria would give precedence to grids in the region near the feature of interest.

The criteria are used to create a floating point "ranking" of each grid of the AMR hierarchy. The grids must then be sorted by their "rank" to create a specific order in which they will be loaded and processed. We intend for the next generation of ChomboVis development to employ threading techniques that permit background processing by the visualization pipeline simultaneously with interaction by the user. The grid metadata is first passed through the pipeline, and each filter in the pipeline applies its "ranking" criteria to the grid. On the second pass, the grids are sorted by the data source, loaded, then propagated through the visualization pipeline in their ranked-sorted order. This further improves the usability of the visualization system by providing interactivity to the user from the moment that the first grid navigates the visualization pipeline.

### 2.6.3 Objectives and Methods to be Developed

During the first year, we will implement a progressive volume renderer for interactive preview of large data sets (100GB). We will develop requirements and specifications for a framework of a general visualization pipeline based on out-of-core techniques, with special attention to end-to-end design, and metrics for weighting and prioritizing the loading and processing grids.

During the second year, we will perform a preliminary implementation of the instrumented pipeline that uses out-of-core and progressive approaches, and deploy this pipeline in ChomboVis.

In the third year, we will expand grid ranking to include end-to-end system metrics, including rendering time.

# 3 Work Plan

In each of the preceding technical topics, we presented the yearly objectives for each of the projects. They are summarized and integrated in this section.

## 3.1 Year One

- Initial activities focus on generating a single-valued representation defined over the entire computational domain from a discrete, multivalued AMR data model using a technique called grid stitching.
- To complement grid stitching, a “crack free” isosurface tool will be developed and tested concurrently. We will develop a technique for volume rendering native AMR data using ray casting.
- Hardware-accelerated methods of volume rendering native AMR data will be built using stripped texture and stenciling approaches.
- Compute integral curves from AMR vector data and use this to compute and render vorticity tubes.
- Compute and render explicit embedded boundaries from single grid implicit EB data, and produce volume renderings directly from single grid implicit EB data.
- Develop a Hierarchical B-spline representation.

- Implement a progressive volume renderer used for interactive preview of large data sets (100GB).
- Develop requirements for a framework of general visualization using out-of-core techniques, including end-to-end design and specification of the visualization pipeline, and metrics for weighting and prioritizing grid loading and processing.

### 3.2 Year Two

- Volume rendering of stitched grids with ray casting and cell projection.
- Use view-dependent information to improve performance of ray casting and cell projection.
- Include space partitioning to improve performance of ray casting.
- Track vorticity tubes in time varying AMR vector data.
- Store time varying AMR data so that the time dimension is an explicit dimension of the AMR data grids.
- Extend visualization techniques for single, uniform grids to single, uniform grids with EB using explicit boundaries and directly using the implicit boundary information.
- B-spline wavelet representations and interactive techniques
- Implement a system for visualization based on out-of-core and progressive approaches. Deploy this system in ChomboVis.
- Focus on measuring and responding to visualization processing time. Include view and data-based operations to reduce processing load in progressive visualization.

### 3.3 Year Three

- Quantify and visualization statistics and errors associated with AMR data.
- Add support for hardware-accelerated operations to the cell-projection technique.
- Integrate flux-correction data from AMR codes to reduce/eliminate discontinuities at coarse/fine grid boundaries.
- Look at vector features relevant to other scientific computations using AMR and work toward tracking these features in time varying AMR vector data.
- Extend visualization techniques for AMR data sets developed earlier/above to AMR data sets with EB using explicit boundaries and directly using the implicit boundary information.
- Efficient and adaptive utilization of B-spline representations.
- Include rendering time in end-to-end performance measurement

## 4 Budget

We are requesting $781K per year for a three year period (see attached). This budget will cover two FTEs from LBNL, provide support for four GSRs from UC Davis (cost sharing with UC Davis), cover ¼ FTE of UC Davis faculty, and provide for computer equipment maintenance and upgrades, and a modest amount of domestic travel for paper presentations.

## 5 Bibliography

[Bethel] W. Bethel, B. Tierney, J. Lee, D. Gunter, S. Lau, Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization. In Proceedings of SC00, Dallas, Texas, November 2000.

[Berger] M. Berger and P. Colella, “Local Adaptive Mesh Refinement for Shock Hydrodynamics,” Journal of Computational Physics, 82:64-84, May 1989. LLNL Report number UCRL-97196.

[Chombo] http://seesar.lbl.gov/anag/chombo

[ChomboVis1] http://seesar.lbl.gov/anag/chombo/chombovis.html

[ChomboVis2] Terry J. Ligocki, "Implementing a Visualization Tool for Adaptive Mesh Refinement Data using VTK, Visualization Development Environments 2000 Proceedings, http://w3.pppl.gov/vde2000/eproceedings.html

[ChomboVis3] Terry J. Ligocki, Brian Van Straalen, John M. Shalf, Gunther H. Weber, Bernd Hamann, "A Framework for Visualizing Hierarchical Computations", NSF/DoE Lake Tahoe Workshop on Hierarchical Approximation and Geometrical Methods for Scientific Visualization, http://graphics.cs.ucdavis.edu/~hvm00/program.html

[Cox] M. Cox, D. Ellsworth, ‘‘Application-Controlled Demand Paging for Out-of-Core Visualization,’’ Proceedings of Visualization ’97, October 1997, pp 235-244.

[Weber1] Gunther H. Weber, Oliver Kreylos, Terry J. Ligocki, John M. Shalf, Hans Hagen, Bernd Hamann, "Extraction of Crack-free Isosurfaces from Adaptive Mesh Refinement Data", NSF/DoE Lake Tahoe Workshop on Hierarchical Approximation and Geometrical Methods for Scientific Visualization, http://graphics.cs.ucdavis.edu/~hvm00/program.html

[Weber2] G. Weber, B. Hamann, K. Joy, T. Ligocki, K. Ma, J. Shalf, Visualization of Adaptive Mesh Refinement Data, In Visual Data Exploration and Analysis VIII, Proceedings of the SPIE, Photonics West – Electronic Imaging, Volume 4302, January 2001.

[Weber3] G. Weber, O. Kreylos, T. Ligocki, J. Shalf, H. Hagen, B. Hamann, K. Joy, “Extraction of Crack-Free Isosurfaces from Adaptive Mesh Refinement Data,” to appear in Data Visualization 2001, Proceedings of VisSym 2001, Springer-Verlag, Vienna, Austria, May 2001.

[Norman] M Norman, J. Shalf, G. Daues, S. Levy, "Diving Deep: Data-Management and Visualization Strategies for Adaptive Mesh Refinement", Computing in Science and Engineering, July 1999.

[Farin] G. Farin, Curves and Surfaces for CAGD, 4th edition, Academic Press, San Diego, California, 1997.

[Forsey] D.R. Forsey and R.H. Bartels, Hierarchical B-spline refinement, in: J. Dill, ed., Proc. SIGGRAPH '88, Computer Graphics 22(4), pp. 205-212.

[Goldman] R.N. Goldman and T. Lyche, Knot Insertion and Deletion Algorithms for B-spline Curves and Surfaces, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1993.

[Grosso] R. Grosso, C. Luerig, T. Ertl, ``The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data,'' Visualization '97, October 1997, pp. 387-394.

[LaMar] E.C. LaMar, B. Hamann, K.I. Joy (19), High-quality rendering of smooth isosurfaces, Journal of Visualization and Computer Animation 10, pp. 79-90.

[Ma] K. Ma and S. Leutenegger, ``Fast Retrieval of Disk-Resident Unstructured Volume Data for Visualization,'' DIMACS Workshop on External Memory Algorithms and Visualization, May 1998.

[Muraki] S. Muraki, ``Multiscale Volume Representation by a DoG Wavelet,'' IEEE Transactions on Visualization and Computer Graphics, Vol 1, No 2, June 1995.

[Neubauer] R. Neubauer, M. Ohlberger, M. Rumpf, and R. Schwirer, ‘‘Efficient Visualizations of Large-Scale Data on Hierarchical Meshes,’’ Proceedings of Visualization in Scientific Computing ’97, Springer Wien, 1997.

[Stollnitz] E.J. Stollnitz, T. D. DeRose and D.H. Salesin, Wavelets for Computer Graphics, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996.

[Westermann] R. Westermann, ‘‘A Multiresolution Framework for Volume Rendering,’’ Proceedings of the 1994 Symposium on Volume Visualization, October 1994, pp 51-57.

[Zhu] Z. Zhu, R. Machiraju, B. Fry, and R. Moorhead, ‘‘Wavelet-Based Multiresolutional Representation of Compressed Field Simulation Datasets,’’ Proceedings of Visualization ’97, October 1997, pp 151-158.