Future Directions for Adaptive Mesh Refinement in ASCI and Other LLNL Simulation Projects

R.D. Hornung
S.R. Kohn

Center for Applied Scientific Computing

UCRL-ID-128332
August 1997
Future Directions for Adaptive Mesh Refinement in ASCI and Other LLNL Simulation Projects

Richard D. Hornung Scott R. Kohn

Abstract

We have surveyed various computational applications at Lawrence Livermore National Laboratory to determine future directions for adaptive mesh refinement (AMR) research and simulation technology. AMR techniques were originally developed for the numerical solution of hyperbolic equations in computational fluid dynamics. However, scientists are now beginning to extend this AMR methodology to new applications domains, including problems modeled by tightly coupled systems of hyperbolic and elliptic/parabolic equations, neutron transport, hybrid approaches that combine different physical models (e.g., discrete and continuum), and Arbitrary Lagrangian Eulerian (ALE) methods.

Extending the traditional AMR approach to these applications domains will require a better understanding of numerical methods, algorithms, fast linear and nonlinear solvers, parallelization techniques, and software frameworks. The future needs identified in this survey provide the basis for ongoing AMR research within the Center for Applied Scientific Computing (CASC) in support of ASCI and other Laboratory efforts. To that end, we have begun the development of a parallel prototyping framework that will facilitate numerical and algorithmic exploration in AMR applications and enable scientists to rapidly explore extensions of existing AMR technology.
# Contents

1 Introduction .................................................. 1

2 Adaptive Mesh Refinement Methodologies ....................... 2
   2.1 Structured Refinement .................................... 2
   2.2 Unstructured Refinement .................................... 2
   2.3 Tradeoffs between Structured and Unstructured Approaches .................... 5
   2.4 Summary .................................................... 6

3 Overview of Structured Adaptive Mesh Refinement ............... 6
   3.1 Time Refinement ........................................... 7
   3.2 Communication of Boundary Information .................... 7
   3.3 Adaptive Regridding ....................................... 10
   3.4 Synchronization between Levels ............................ 10
   3.5 Summary .................................................... 11

4 Overview of AMR Code Efforts ................................ 11
   4.1 Odyssey ..................................................... 12
      4.1.1 Odyssey Model Equations and Numerical Methods ................. 12
      4.1.2 Odyssey Code Structure .................................. 13
      4.1.3 Potential Contributions to Odyssey .......................... 13
   4.2 ARES ....................................................... 14
      4.2.1 ARES Model Equations and Numerical Methods .................. 14
      4.2.2 ARES Code Structure ..................................... 16
      4.2.3 Potential Contributions to ARES ............................. 16
   4.3 ParFlow ...................................................... 17
      4.3.1 ParFlow Model Equations and Numerical Methods ............... 17
      4.3.2 ParFlow Code Structure ................................... 20
      4.3.3 Potential Contribution to ParFlow .......................... 20
   4.4 Ardra ....................................................... 21
      4.4.1 Ardra Model Equations and Numerical Methods .................. 21
      4.4.2 Ardra Code Structure ...................................... 22
      4.4.3 Potential Contribution to Ardra ............................ 22
   4.5 Hybrid Applications ........................................ 23
1 Introduction

This report details the results of a survey of various code efforts at Lawrence Livermore National Laboratory (LLNL) that either employ adaptive mesh refinement (AMR) currently or for which the use of AMR is planned. The goals of this survey were to determine the computational needs for AMR at LLNL and to identify common threads in development efforts. This survey provides the basis for ongoing AMR research within the Center for Applied Scientific Computing (CASC) in support of ASCI\(^1\) and other Laboratory efforts.

Structured adaptive mesh refinement (AMR) is a computational technique that supports the numerical solution of systems of partial differential equations using varying degrees of spatial and temporal resolution within the computational domain. It focuses computer resources (CPU time and memory) where they are needed most, in localized regions of a computational domain based on numerical accuracy requirements. AMR is an important technology needed to support large, physically and numerically well-resolved, three-dimensional ASCI simulation efforts.

The structured AMR approach is rooted in the work of Berger, Colella, and Oliger [6, 7]. The term structured refers to the use of logically-rectangular refinement patches. Local resolution is implemented via a hierarchy of nested levels of spatial and temporal local grid refinement, where the computational grid associated with each level is composed of a union of patches. Structured AMR was originally introduced for applications in shock hydrodynamics [6, 7]. More recently, structured AMR has been applied successfully to other problems in fluid dynamics [1, 2, 32, 36, 52, 53], porous media flow [30, 31], nonlinear solid mechanics [57], neutron transport [33], and shear band formation [25]. There are other AMR methodologies, such as unstructured refinement and structured cell-by-cell refinement, that differ fundamentally in numerical methods and software support. In this survey, we address advantages and disadvantages of the various approaches and highlight reasoning prompting the choice of one approach over another.

Experience has shown that AMR applications require substantially more complicated programming and longer development time than codes employing uniform computational grids. Based on the results of this survey, we are developing a general, flexible, parallel software framework for structured AMR within CASC. This framework will provide an important tool to facilitate future AMR research and development. We aim to provide a re-usable, easily extended software framework that can support the wide variety of applications of interest to the Laboratory. The infrastructure will provide an environment for fundamental AMR numerical and algorithmic exploration within CASC and a basis for application-specific AMR development in various Laboratory programs. The advantages of this approach include reduced code development time and broader, more in-depth research into numerical methods for AMR applications.

This report contains five sections and an appendix. In Section 2, we briefly describe unstructured, structured cell-by-cell, and patch-based structured AMR methodologies. Section 3 provides a more detailed introduction to structured AMR techniques as developed by Berger, Colella, and Oliger. In Section 4, we present results from our survey of AMR needs in ASCI and other code development efforts at LLNL. There, we describe concerns related to each code group and present a summary of research issues and potential impact for our proposed AMR effort. Since we advocate the use of structured AMR, our discussion is centered on applications that we believe are well-suited to this type of adaptive refinement. Section 5 gives a brief overview of the AMR framework development effort in CASC. In particular, this effort is driven by design principles formulated after careful consideration of the results of this survey. Finally, Appendix A describes the basic radiation transport and hydrodynamics equations used in two of the ASCI applications in this survey.

\(^1\)Accelerated Strategic Computing Initiative
2 Adaptive Mesh Refinement Methodologies

Numerical simulations, especially those requiring complex physics in three spatial dimensions, tend to be very computationally intensive. Often, high resolution is needed only in localized portions of the computational domain. Therefore, a grid that uniformly covers the entire domain may be unnecessarily fine in regions away from physical phenomena of interest. Adaptive mesh refinement concentrates computational effort near local phenomena where it is needed most during the course of a simulation. By supporting spatial—and, in some cases, temporal—refinement in an efficient, straightforward manner, local refinement can provide desired resolution with reduced computation costs (in both memory and execution time). In this section, we describe three primary AMR methodologies: structured patch-based refinement, structured cell-by-cell refinement, and unstructured refinement.

2.1 Structured Refinement

The structured AMR methodology [6] (which we describe in more detail in Section 3) employs a hierarchy of nested refinement levels. Structured AMR provides an environment in which a small number of computationally rich tasks (defined in terms of operations on a single refinement patch) can be organized in a highly structured fashion. Computational cells on each level of this hierarchy are clustered to form a collection of logically-rectangular patches. The coarsest level of the grid hierarchy induces an abstract integer lattice cell index space for the computational domain. Each successively finer level, in turn, is a refinement of this global index space. Data dependencies between patches are described in terms of these abstract index spaces, greatly simplifying bookkeeping. Figure 1 illustrates a simple structured AMR grid for which the distinct levels of refinement are separated. Figure 2 shows the composite grid configuration flattened to highlight the local spatial refinement.

2.2 Unstructured Refinement

Other approaches to local mesh refinement that are not based on the patch approach have been developed for a variety of applications [5, 20, 21, 22, 34, 35, 41, 47, 48, 54]. These local mesh refinement strategies can...
be partitioned into two categories, which we shall refer to as unstructured and cell-by-cell. These approaches do not cluster cells to form patches; rather, cells are managed individually. Consequently, unstructured and cell-by-cell strategies require fundamentally different grid and data management structures from the patch-based methodology. For the most part, these alternative approaches cannot exploit the simple notion of an abstract index space to manage data communication among cells easily. Thus, management of the grid structure and data is highly coupled, is more complicated operationally, and requires data structures significantly more complex [22, 48] than structured AMR.

Unstructured refinement algorithms usually focus on general finite element grids in which the nodes have arbitrary location and connectivity. Figure 3 shows an unstructured grid example that provides comparable resolution to that shown in Figure 2. Since the computational nodes, and thus the data, are not arranged in a pattern of regular connectivity, managing communication between neighboring nodes involves substantial indirect addressing. Such unstructured indirect addressing can result in severe performance problems on modern cache-based computer architectures, which are memory bandwidth and latency limited and require high degrees of cache locality to effectively exploit floating point units. Furthermore, because the description of the grid structure is very complex, graph connectivity information often must be distributed across processor memories for parallel execution, especially for grids that change dynamically. Thus, communication and data dependency resolution can be substantially more expensive for unstructured approaches when compared to structured approaches.

Furthermore, unstructured mesh applications do not readily support local time refinement since the data associated with unstructured meshes is not stored in a hierarchical fashion corresponding to distinct refinement levels. Therefore, unstructured refinement has been employed solely to gain local spatial accuracy for solving elliptic partial differential equations (PDEs) [5, 48] or for implicit time integration [34, 54]. In doing so, unstructured methods often attempt to obtain maximum accuracy for a fixed cost (e.g., a fixed number of computational cells) [47, 54]. R"ude [54] presents an interesting adaptive refinement approach that constructs a hierarchy of finite element spaces during an iterative solution process to meet accuracy requirements for elliptic PDEs. However, it is not clear whether this approach is applicable to solution methods employing local time refinement.

In the cell-by-cell approaches [22, 41], computational cells are refined individually, but the cells have
Figure 3: An unstructured grid showing comparable local resolution to the structured AMR grid shown in Figure 2.

Figure 4: A composite grid hierarchy refined in a cell-by-cell fashion.
uniform geometry and regular connectivity, as illustrated in Figure 4. The grid configuration is managed by tree-like data structures that maintain the child-parent relationships between cells. As a result, data management can be interpreted in a hierarchical fashion with respect to the underlying grid, and so distinct grid refinement levels can be considered relatively easily. Thus, this refinement approach supports temporal grid refinement in a reasonably straightforward manner. Figure 5 shows the exploded view of the parent-child relationships for the composite grid configuration of Figure 4. The dashed arrows represent some of the child-parent connections.

In the cell-by-cell approach, no attempt is made to cluster refined patches to exploit data locality, as is done in the structured approach. The lack of structure makes data management and parallelism more complicated and introduces more indirection overhead and corresponding performance problems; however, substantial progress has been made in the application of hp-adaptive refinement techniques to hyperbolic problems [21] within this paradigm.

2.3 Tradeoffs between Structured and Unstructured Approaches

There are several application-dependent reasons to choose one of the three adaptive mesh refinement methodologies over the others. The various tradeoffs involve ease of implementation, cache utilization and single processor performance, ease and efficiency of parallelization, and efficiency and accuracy of the applicable numerical methods.

The individual cell approaches require complicated pointer-based data structures to manage inter-cell communication. In these less-structured methodologies, computational tasks must be organized by operations associated with single grid cells. This results in irregular difference schemes and extensive use of indirect addressing to retrieve information stored for adjacent cells. If the refinement configuration is sufficiently complicated, bookkeeping overhead may dominate useful computational work.

Unstructured and cell-by-cell approaches may be able to accommodate complex geometries more easily and produce refined regions covering less of the computational domain than the patch approach does. To maintain logically rectangular patch structures, structured AMR often refines more cells than are necessary for adequate resolution. Thus, less-structured methods may seem to allow for more efficient refinement of complex physical features. However, recent significant work suggests that structured local refinement is
well-suited to problems involving irregularly-shaped regions [1, 15, 16, 28, 52].

The patch-based structured AMR approach has several advantages over the other refinement strategies for scientific computing applications.

1. Patches exploit data locality. Numerical kernels can be programmed as though the computational grid is uniform, since data are stored as logically rectangular arrays.

2. Patch location and size and the extent of inter-patch communication are easily characterized in terms of simple region calculus operations based on the underlying abstract index spaces.

3. Data can be easily distributed in blocks for parallel applications. Because the structure of the AMR hierarchy (based on the abstract index space) can be represented succinctly, structural information can be replicated across processor memories. Thus, communication patterns representing the data dependencies between patches can be computed in parallel without interprocessor communication.

4. Since numerical data associated with the adaptive grid is stored in a hierarchical fashion corresponding to levels of grid refinement, local time refinement algorithms can implemented in a straightforward, recursive manner.

5. Numerical routines used to solve the equations on an adaptive grid configuration are easily separated from the implementation details of the AMR data structures. Aside from the implementation of patch boundary conditions, subroutines for numerical computation in patch-based AMR applications can be identical to those used in uniform grid calculations.

6. Logically-rectangular arrays of data are very amenable to the design and implementation of efficient and accurate numerical procedures, such as shock-capturing schemes and hierarchical multigrid linear algebra routines [6, 57, 44].

2.4 Summary

In summary, AMR can reduce the computational cost of numerical simulations significantly. The tradeoff is that the implementation of an adaptive algorithm presents substantially greater programming effort and complexity than that required for applications using only uniform grids. Of the three types of AMR described here, the structured approach has substantial advantages over the others in terms of data management and computational efficiency, especially for applications that can exploit local time refinement. Furthermore, we believe that this approach can be made sufficiently general to apply to most applications currently considered to be the domain of less-structured refinement approaches.

3 Overview of Structured Adaptive Mesh Refinement

The structured AMR approach that we describe is rooted in the methodology introduced by Berger, Colella, and Oliger [6, 7]. The structured adaptive grid is maintained as a hierarchy of levels of nested local grid refinement. Each level is comprised of a union of parallelepiped regions, each of which consists of a logically-rectangular cluster of grid cells called a patch. The configuration of patches on any level but the coarsest is determined by refining a collection of cells on the next coarser level. That is, the patches are defined in terms of index spaces representing recursive refinements of the coarsest grid. Nested refinement refers to the fact that the union of patches on a level is contained within the union of patches corresponding to the next coarser level, if it exists. An example of the sort of grid structure generated in this manner is shown in Figure 2.
3.1 Time Refinement

In the traditional AMR scheme, systems of hyperbolic conservation laws are integrated in time using explicit conservative difference methods. Time integration over the hierarchy generally requires time refinement to satisfy local Courant-Friedrichs-Lewy (CFL) conditions on the various levels of spatial grid refinement. The time integration procedure can be viewed as a recursive algorithm that advances data on one hierarchy level and then advances data on successive finer levels. The recursive procedure associated with a particular level is as follows.

1. Determine the initial time increment for integrating the data on the level.

2. Loop over timesteps on the current level until the simulation time of the current level reaches the new simulation time on the next coarser level. Each of the timesteps may potentially use different time increments (based on solution changes and CFL stability requirements).
   (a) Advance the patches (i.e., integrate the conservation laws) on the current mesh level through the time increment associated with the timestep.
   (b) If it exists, advance the next finer level to the new time on the current level by recursively invoking this level integration process on the next finer level.
   (c) If appropriate, regrid all levels finer than the current level.

3. When the timestep loop has completed, synchronize the solution on the current level with the next coarser level.

Figure 6 illustrates a hypothetical timestep sequence for a problem with three levels of grid refinement (labeled 0, 1, and 2 from coarsest to finest) and a refinement ratio of 4. The numbers on the diagram indicate the order of the timesteps on the three levels of mesh refinement. The recursive time-stepping process begins on the coarsest level. Next, levels 1 and 2 are advanced to the new time associated with level 0 by repeatedly applying the recursive timestep algorithm to those levels.

Before performing time integration on a particular level, the algorithm must calculate a suitable time increment. The standard approach for explicit conservative difference methods invokes the CFL criterion. This approach yields the largest stable time increment that may be taken on the spatial grid at that level. Smaller time increments than required by the CFL condition will almost always be used to facilitate regridding and synchronization with other levels in the hierarchy. If the current level is the finest in the hierarchy and the level does not allow additional refinement, then the algorithm is free to use the largest time increment allowed by numerical stability. If the current level does allow refinement, however, then the number of timesteps taken on the level is usually chosen to be a divisor of the grid refinement ratio between levels. As AMR branches out into new problems areas, new timestep selection methods and time integration routines will be required [25, 26]. Although our proposed AMR infrastructure is based on the needs for this standard algorithm, it will not restrict users in any way from developing alternative time integration strategies.

3.2 Communication of Boundary Information

One of the more important considerations of any AMR approach is the proper communication of boundary information among refinement patches in the composite grid hierarchy. Communication of neighboring cell data precedes nearly every patch operation. It is therefore vital that the communication of boundary data is efficient and robust. The method used to communicate boundary information must also be very general, since the structured grid configuration can be fairly complicated. In the structured AMR approach, each patch is surrounded by a layer of ghost cells. These ghost cells completely separate physical boundary conditions and coarse-fine interface conditions from the details of the difference equations; thus, the numerical treatment of all patch in the hierarchy is essentially the same.
Figure 6: Hypothetical timestep sequence for refinement ratio of 4. Here, we have three mesh levels: coarse, fine, and intermediate. The circles indicate the timesteps and levels where regridding of finer levels is initiated. The regrid interval is 2. Notice that after step 38 on the intermediate level, we defer regridding to after the return from the recursive call from the coarse level after step 22.
Recall that coarse patches are integrated before fine patches. This implies that coarse patches must provide boundary data to finer patches. In this setting, ghost cell values arise from one of three possible sources, depending on the location of the cell within its associated level and, more generally, within the computational domain. Ghost cell data are first sought from user-prescribed boundary data at the edge of the computational domain and then from interior cells to patches on the same refinement level. If all of the ghost cell data cannot be obtained in one of these two ways, then data are recursively sought from successively coarser levels, until all of the ghost cell data are obtained. Figure 7 illustrates a situation for which ghost cell data must be obtained from each of the three possible sources: physical boundaries, neighboring patches at the same level of refinement, and patches at coarser levels of refinement.

When considering the communication of data among the various levels of the AMR hierarchy, it is vital to keep in mind that the values of mesh-based quantities at a particular point in space may depend in a complicated fashion on other unknowns. Certain quantities, such as the conserved unknowns, are cell-centered and may represent cell-averaged masses. In contrast, numerical fluxes associated with a conservative finite-difference method correspond to centers of space-time sides of cells. Special care must be taken to resolve the ambiguity of the definition of different variables on the adaptive grid hierarchy. This is true in the specification of ghost cell data as well as synchronization of data between different levels when they have been integrated to the same point in time. New procedures for communicating data between levels will need to be explored as AMR is introduced to different applications. Often, this communication is straightforward and is based on notions of conservation or finite element space nesting. In other situations, this process can be very complicated and require methods based on physical interpretation of the quantities involved [30].
3.3 Adaptive Regridding

Another important piece in the AMR puzzle is that the grid configuration changes as the simulation evolves. Since we are interested in local refinement in the context of time-dependent problems involving transient phenomena, the AMR algorithm must invoke a regridding procedure that moves the refinement regions at certain points during time integration. We adhere to three main design principles in the regridding process. First, we would like to locate new fine patches only where they are needed to resolve important flow features. If possible, we would like to avoid unnecessarily large refined regions to minimize storage and computational costs. Second, we must maintain proper nesting of the levels of refinement so that important solution features are not moved off of refined regions before the next regridding phase. Third, we would like to avoid moving the grid after every timestep on each level since grid determination and data movement can be very costly.

The circles at certain timesteps in Figure 6 indicate the times at which the algorithm initiates the regridding. When the regridding process starts at a particular level, grid refinements at all finer levels may be recalculated. Note that we adapt the grid after a fixed number of timesteps on each level; here, we use two since the refinement ratio is even. Note that regridding may be deferred to a coarser level. If the timestep is the last in the loop on the current level, then the fine level must be at a synchronization time with the next coarser level. If it is time to regrid at the next coarser level, then regridding can be deferred to that level, avoiding redundant work. This situation appears in the figure after step 38.

3.4 Synchronization between Levels

Finally, we address data synchronization between levels (commonly known as relensing in the context of the standard hyperbolic AMR algorithm). When we perform numerical integration on an adaptive grid, it is vital that patches on different mesh levels communicate in the proper fashion. We have noted previously that boundary data for fine patches may be obtained from coarser patches. However, fine patches typically contain the most accurate numerical results. Thus, at times when it is appropriate for patches on different levels to be synchronized, we use information obtained on fine patches to modify the data on coarser patches.

When integrating hyperbolic conservation laws, we employ discretization methods that conserve mass. The numerical approximations are written as conservative differences. For example, a conservation law

$$\frac{\partial m}{\partial t} + \nabla \cdot F = q$$

is rewritten via the divergence theorem in integral form

$$\int_{\Omega} m|_{l_2} dx - \int_{\Omega} m|_{l_1} dx = \int_{\Omega} \int_{l_1}^{l_2} F \cdot \hat{n} ds + \int_{l_1}^{l_2} q dt dx.$$  

Given this format, it is natural to communicate between coarse and fine computational scales via the flux boundary integrals. Contributions on fine grids are summed over the integration time and then used to replace coarse grid values. Therefore, data synchronization between different grid levels requires us to accumulate and store additional information, such as numerical fluxes and flux integrals, on patches during the course of integration. The synchronization of coarse and fine levels is a three step process. Assume the time integration algorithm is at a synchronization point between successive levels. First, we replace the stored coarse fluxes at coarse cell edges intersecting boundaries of overlying fine patches with time and boundary integrals of the fine fluxes. Second, we repeat the conservative difference on the coarse patches using the coarsened flux information. Third, we replace data in coarse cells underlying fine cells with an appropriate average of fine cell values. This process allows us to enforce mass conservation over the entire grid hierarchy so that our approximation is consistent with the equation we wish to solve. We remark that the synchronization process is well-understood only in the context of applying explicit, conservative finite difference methods to systems of hyperbolic conservation equations. As AMR is extended beyond this traditional problem domain, new synchronization algorithms must be explored.
3.5 Summary

This brief description of some AMR algorithmic issues is based on requirements for solving hyperbolic conservation laws using explicit conservative difference methods on an adaptive grid employing both time and space refinement. The components of the algorithm serve as points of departure for future AMR development. The incorporation of AMR into other application domains requires the exploration of new algorithms. Time integration needs are different for problems that do not fit into the standard explicit conservative difference template. The issues associated with data synchronization and consistency across the grid hierarchy also require further research. Despite recent work extending structured AMR to problems in nonlinear solid mechanics [57], shear band formation in elastic-plastic solids [25], and flow in porous media [30, 31], much work remains to apply AMR to a wider range of important simulation problems. We address many of these issues in the remainder of this report.

4 Overview of AMR Code Efforts

The research into adaptive mesh refinement simulation technology within CASC is driven by the development needs of ASCI and non-ASCI codes at LLNL. We have surveyed the following code efforts to better understand current and future computational requirements:

- Odyssey (A Division)
- ARES (B Division)
- ALE3D (B Division)
- KULL (X Division)
- AMTRAN (B Division)
- ParFlow (CASC)
- Ardra (CASC)
- Hybrid applications (Garcia, Alder, Garaizar, et al.)

The first four codes are ASCI projects that simulate some form of the radiation transport and hydrodynamics equations (see the Appendix). Of these four, only Odyssey and ARES are based on structured grids and therefore candidates for structured AMR. ALE3D and KULL employ alternative types of adaptivity technology using unstructured grids; consequently, they require different numerical methods and software frameworks.

AMTRAN is a production neutronics code in B Division that currently utilizes a form of spatial mesh refinement directly coupled to the discrete ordinates numerical approach. Because AMTRAN is a somewhat mature code, it is unlikely that CASC will be in the position to contribute significantly to this effort. ParFlow is a subsurface simulation project aimed toward hydrology applications. Ardra is a neutron transport code used to study nuclear well logging, tokamak reactor flow, and medical applications. The DSMC/NS (Direct Simulation Monte Carlo/Navier Stokes) effort involves multi-physics modeling by Alejandro Garcia (Department of Physics, San Jose State University) and Berni Alder (LLNL). Each of these efforts use structured computational grids; thus, they provide excellent test problems for our structured AMR project.

We have included the two CASC codes and the multi-physics applications in our survey to better understand what types of AMR technology may be necessary in the future. While our focus is on the needs

---

2The numerical and software needs of ALE3D and KULL are being addressed by the scalable numerical algorithms effort within CASC.
of ASCI codes, many interesting and unique computational issues involved in Ardra, ParFlow, and hybrid application codes will become increasingly important to ASCI as more complex physics is introduced into simulation models.

For example, the DSMC/NS project of Garcia and Alder models fluid flow using two different physical models. DSMC/NS simulates fluid flow in rarefied regions via a Direct Simulation Monte Carlo method based on randomized particle interactions; other flow regimes solve the standard Navier-Stokes equations. Garaizar has developed an AMR algorithm that combines an elasticity/plasticity continuum model with a discrete constitutive relation to study shear band formation in granular materials [25]. H Division is interested in pursuing similar multi-physics simulations in the areas of crack propagation and fracture. Here the crack tip would be modeled using atomistic methods with empirical interatomic force laws whereas the material bulk would be modeled using the standard continuum equations.

The introduction of local grid refinement into Ardra applications requires the extension of robust pre-conditioners developed for uniform grids to composite grid configurations. Research in this area will be applicable to the needs of ASCI neutronics efforts that use some form of local grid refinement.

In the following five sections, we describe the Odyssey, ARES, ParFlow, Ardra, and hybrid applications projects in more detail. For each effort, we present the model equations and numerical methods, software implementation, and potential contributions from CASC. Section 4.6 summarizes our potential contributions to the various code efforts and outlines outstanding research issues.

4.1 Odyssey

Odyssey is an ASCI rapid prototype code for hydrodynamics and radiation diffusion under development in A Division. It is based on previous AMR development within the Applied Mathematics Group (AMG) of the Center for Computational Science and Engineering (CCSE) while at LLNL (this group has since moved to LBNL). Although AMR is a mature method for problems in hydrodynamics, it has not yet been applied to applications that involve coupled radiation diffusion processes. Thus, Odyssey is intended as a testbed to explore AMR simulation technology as an alternative to the Lagrangian or ALE (Arbitrary Lagrangian Eulerian) techniques currently in use.

The current development team is comprised of the following individuals:

<table>
<thead>
<tr>
<th>Name</th>
<th>Department</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stewart Brown (PI)</td>
<td>A Division</td>
<td><a href="mailto:sabrown@llnl.gov">sabrown@llnl.gov</a></td>
</tr>
<tr>
<td>Thomas Kelleher</td>
<td>Environmental Computer Applications</td>
<td><a href="mailto:kelleher1@llnl.gov">kelleher1@llnl.gov</a></td>
</tr>
<tr>
<td>Michael Lambert</td>
<td>CASC</td>
<td><a href="mailto:mlambert@llnl.gov">mlambert@llnl.gov</a></td>
</tr>
<tr>
<td>Doug Peters</td>
<td>CASC</td>
<td><a href="mailto:peters3@llnl.gov">peters3@llnl.gov</a></td>
</tr>
</tbody>
</table>

4.1.1 Odyssey Model Equations and Numerical Methods

The physics being modeled by the Odyssey development team is based on the coupled radiation diffusion and hydrodynamics equations described in detail in the Appendix. In the longer term, the code will include neutron transport, Monte-Carlo transport, ray tracing (laser energy deposition), and complicated material modeling. However, the most immediate concerns are (1) developing practical iterative solution procedures for the linear system of equations resulting from the discretization of the radiation diffusion equation, (2) parallelizing the code for distributed memory architectures (e.g., ASCI Blue Pacific), and (3) determining appropriate coupling methods for the combined radiation diffusion and hydrodynamics simulations.

The basic hydrodynamics model in Odyssey is that of Euler's equations for gas dynamics. This model describes the motion of an inviscid, compressible fluid. Odyssey currently implements two numerical methods to integrate the system of conservation equations in time. The first is a new approach by Chang [13] based on space-time conservation elements. The second approach, due to Colella [17], is an explicit time integration
process based on the Godunov method [27] and the MUSCL scheme of van Leer [60]. It is formally second-order accurate in space and time. Unlike other hydrodynamics methods, neither of the methods is formulated as Lagrangian integration step followed by a re-mapping procedure [9]. Rather, Eulerian integration is performed as a single time integration step. Operator splitting is used in multiple spatial dimensions.

The time integration algorithm in Odyssey employs the standard AMR time refinement strategy for hydrodynamics based on the work of Berger, Oliger, and Colella [6, 7]. Recall that time refinement means that data corresponding to fine-scale refinement patches may be at a different simulation time than that associated with coarser patches. Although local time refinement can significantly reduce simulation times and limit unwanted numerical diffusion away from refined regions, it introduces substantial complications in coupling with the radiation diffusion equations (see Section 4.1.3). Currently, the hydrodynamics calculations are working with AMR; however, Odyssey is not yet functioning at the level of a production code.

The radiation diffusion model in Odyssey resembles the three-temperature model described in Section A.2. Olson [49] describes this approximation and some of the issues surrounding multiple materials and interface reconstruction. The currently proposed solution method for the coupled system of diffusion equations on an adaptive grid hierarchy is based on an Alternating Direction Implicit (ADI) approach as a local solver for a refinement level [43]. It is not clear how the unknowns on the various levels of the hierarchy are to be coupled in the solution process.

4.1.2 Odyssey Code Structure

Odyssey began as a derivative of the AMR hydrodynamics code developed in the CCSE Division at LLNL. Portions of the code retain some of the features of the BoxLib C++ [19] class library from which it began. It has been modified to fit within the PACT simulation framework [12] developed at LLNL. Most high-level routines in Odyssey are written in C++ while the numerical kernels are written in C.

Consistent with the BoxLib class structure, Odyssey uses data distribution in terms of logically rectangular patches, with numerous unknowns represented on each patch. Odyssey supports structured AMR on orthogonal grid discretizations in Cartesian, spherical, and cylindrical coordinate systems. As in the original AMR framework, the numerical procedures are decomposed into computations performed on locally rectangular subdomains with appropriate coupling among patches.

For parallel execution on shared memory machines, Odyssey uses a bag-of-tasks paradigm. All patches to be integrated are placed on a shared work queue. When processors become available, they remove a patch from the work queue and perform the required numerical work. This type of approach is natural for shared memory machines with a flat memory model, such as the Cray J90. However, a shared work queue approach does not scale to cache-based shared memory machines since the tasks may migrate between processor memories, resulting in a significant amount of memory traffic to redistribute cache data and invalidate stale cache lines. Major changes will be required for distributed memory computers, such as the ASCI Blue Pacific, that employ message passing instead of a shared address space for interprocessor communication.

4.1.3 Potential Contributions to Odyssey

We have identified three primary areas in which the CASC AMR effort can impact the Odyssey project. The first area of research is the solution of the three coupled radiation diffusion equations within an AMR framework. The Odyssey code team is currently pursuing an ADI solver [43]. However, it is not clear how to couple the various refinement levels of the hierarchy with ADI. We propose to develop a conjugate gradient (CG) iterative solver for adaptive grids with an FAC (Fast Adaptive Composite) preconditioner [45] to accelerate the solution process. We will compare the robustness and computational efficiency of the ADI approach with CG-FAC.

The radiation diffusion equations can be difficult to solve using fast solvers such as multigrid because of highly discontinuous coefficients. Also, there are interesting issues with respect to discretization on the
interfaces of coarse and fine grids of an AMR hierarchy. Robust multigrid-preconditioned iterative solvers for uniform grids were developed for the ParFlow project (see Section 4.3). We will leverage this experience along with developments associated with the multigrid iterative solver project within CASC. In addition, we are pursuing new discretization and multigrid approaches with other researchers in CASC and CASC academic subcontractors from the University of Colorado at Boulder (Steve McCormick and Tom Manteuffel).

The second research issue concerns the coupling between the hydrodynamics and radiation diffusion equations in an AMR framework. As described in Section 3, AMR for hydrodynamics has traditionally employed time refinement, which means that fine levels advance in time using smaller time increments than coarser levels. Thus, hydrodynamic variables on fine patches are defined at different times than those on coarser patches. However, the radiation diffusion equations are parabolic and thus require global information at a single simulation time. The current plan in Odyssey is to couple these two equations using operator splitting at a very coarse time granularity (i.e., the discrete integration times corresponding to the coarsest grid level), although it is probable that this will not result in satisfactory solutions for most cases. In all likelihood, a more sophisticated coupling will be required. The coupling of parabolic/elliptic equations with hyperbolic equations within an AMR framework is still an outstanding research question for the AMR community.

The final research area is the development of techniques for the efficient scalable implementation of AMR on large parallel message passing architectures. AMR algorithms are complicated, and parallelization makes them even more so. Structured adaptive mesh applications are difficult to implement on parallel architectures because they rely on dynamic, complicated data structures with irregular communication patterns. On message passing platforms, the programmer must explicitly manage data distribution across processor memories and orchestrate interprocessor communication. Such implementation difficulties soon become unmanageable and can obscure the mathematics behind the algorithms. Research is needed into the proper programming abstractions, load balancing methods, processor assignment algorithms, and software framework needed to support AMR on massively parallel architectures. We aim to provide an environment for the exploration of these concerns with our proposed AMR framework.

4.2 ARES

ARES is an ASCI radiation diffusion and hydrodynamics code under development in B Division. This code extends the capabilities of the previous CALE code to treat problems in three spatial dimensions. Both ARES and CALE employ ALE (Arbitrary Lagrangian Eulerian) numerical techniques, which have been used extensively in Laboratory production codes. The Eulerian approach of Odyssey described in the previous section is based on the calculation of fluxes across fixed mesh cells. In contrast, ALE mesh points are allowed to advect with the material in a Lagrangian fashion. To avoid severe distortions in the computational grid, mesh points are occasionally re-mapped to new, more numerically stable configurations.

The current ARES development team is comprised of the following individuals:

Gary W. Carlson B Division carlson6@llnl.gov
Frank R. Graziani B Division grazianil@llnl.gov
Gregory M. Greenman B Division greenman1@llnl.gov
Thomas L. McAbee (PI) B Division mcabee1@llnl.gov

4.2.1 ARES Model Equations and Numerical Methods

The physics modeled by ARES is based on the coupled radiation diffusion and hydrodynamics equations described in detail in the Appendix. These equations are similar to those in Odyssey. The primary difference between Odyssey and ARES lies in the solution methodology: Odyssey uses an Eulerian solution strategy with adaptive mesh refinement whereas ARES employs an ALE approach whereby the grid moves with the
flow field. In this section, we begin with a basic description of the ALE approach and then discuss the ARES code in more detail.

Our description of the ALE method follows that of the seminal paper by Hirt, Amsden, and Cook [29]. We will solve the Euler equations to simplify the presentation; however, ALE can be applied to more complex systems of equations. Pressures \( p \), internal energies \( I \), total conserved energies \( E \), cell volumes \( V \), and densities \( \rho \) or masses \( M \) are all stored at cell centers. Mesh coordinates \( x \) and velocities \( u \) are stored at cell vertices.

The simplest ALE procedure consists of two phases. The first phase is an explicit Lagrangian computation. In the second phase, which is optional, the mesh can be moved to a new configuration. During re-zoning of the mesh, it will be necessary to compute convective fluxes across cell interfaces to account for the movement of fluid between cells as cell boundaries change.

The Lagrangian step uses operator splitting to update the conserved quantities. The phase begins with all quantities at current time \( t^n \). First, a new velocity field \( u^L \) and a new energy field \( E^L \) (where the superscript denotes the value at the end of the Lagrangian phase) are computed using approximations to the integral form of the conservation laws with mesh coordinates, pressures, densities, and forces corresponding to the current time. To avoid time step constraints due to sound speeds, it may be necessary to calculate intermediate pressures \( p^L \) by applying a Newton-Raphson iteration to an equation implicitly relating pressure, density, internal energy, and cell volumes. Then, using the new velocities \( u^L \), the mesh points are moved according to the cell volume evolution approximation

\[
    x^L = x^n + \Delta t u^L 
\]

for a timestep \( \Delta t \). Finally, the new density \( \rho^L \) is calculated for each new cell as the quotient of the original cell mass divided by the new cell volume.

Lagrangian computations alone are not adequate for describing flows undergoing large distortions. In such cases, the mesh can become tangled, resulting in numerical instabilities. ALE eliminates these problems by adjusting the mesh vertices with respect to the fluid so as to maintain a reasonable mesh structure. Whenever the mesh is moved, however, material must be redistributed. This exchange of conserved quantities between cells can be interpreted as a convective flux, reminiscent of the flux across cell interfaces in the Eulerian approach.

The re-zoning phase takes the Lagrangian quantities of the previous step and updates them to the beginning of the next timestep \( t^{n+1} \). First, all quantities are converted to conservation form (e.g., velocities are converted into momenta) so the redistribution will be conservative of mass, momentum, and energy. Next, the mesh vertices are moved according to some metric that describes a "good" mesh. The movement of a mesh vertex will result in a mass, momentum, and total energy exchange among adjacent cells based on the new cell volumes. After the redistribution, physical quantities are converted back from conservative form (e.g., momenta are converted back into velocities) and the algorithm begins with the Lagrangian computation for the next timestep.

Thus far, our discussion of the ALE procedure has concentrated only on a simplified version of the hydrodynamics equations. We now describe the ARES radiation diffusion equations. ARES employs the standard operator splitting integration scheme to decouple the hydrodynamics and radiation diffusion solvers. Equations of state and constitutive models (e.g., material stress models) are the same as in CALE.

The radiation diffusion approximation uses a finite element discretization on tetrahedra. A single temperature unknown is used to model the radiation field in "clean zones," and two temperature unknowns are used within "mixed zones" (i.e., those containing more than one material). Currently, the solution procedure for the diffusion equation explicitly forms the matrix corresponding to the discretization on the entire grid. This matrix is symmetric due to the conservation properties of the discretization. It has the following block structure:

\[
    \begin{bmatrix}
        C & B \\
        B^T & M
    \end{bmatrix}
\]
where $C$ corresponds to the clean zones and $M$ corresponds to the mixed zones. Submatrix $C$ has a 27-point stencil structure and $M$ has a block $(2 \times 2)$ 27-point stencil structure. The ARES code currently uses a conjugate gradient iterative method with either a first-order Von Neumann preconditioned or two-step point Jacobi preconditioner. The ARES development group is willing to change its current sparse matrix storage format to accommodate new solution strategies, such as those under development for the ASCI Scalable Linear Algebra project. A matrix-free approach may potentially facilitate the introduction of geometric multigrid solvers and simplify the integration with AMR.

### 4.2.2 ARES Code Structure

Like its predecessor CALE, ARES is written in C and employs the ALE integration strategy on block-structured computational grids. That is, the data associated with the Lagrangian computational mesh is stored as a collection of logically rectangular arrays. Each of these blocks is surrounded by "ghost" zones to specify boundary conditions. The existence of the boundaries is shielded from users as much as possible; that is, developers refer to actual node indices and not ghost indices. All communication to support boundary conditions on the blocks is performed internally within the ARES environment.

Some interesting capabilities exist in ARES to specify physical regions containing different materials in terms of primitive shapes, such as ellipses, cones, spheres, etc. The numerical routines are implemented in a form of generalized cylindrical coordinates; for example, spherical coordinates are approximated within this framework.

Target problems for ARES will eventually involve 20-50 million spatial zones. Simulations involving one million zones are the near-term goal. The code is implemented in the SPMD parallel programming model using MPI message passing.

### 4.2.3 Potential Contributions to ARES

Two potential contributions to ARES were identified during our survey effort. First, the ARES code group is interested in pursuing some form of multigrid or adaptive relaxation procedure \([55]\) to accelerate convergence in the radiation diffusion solver. Since the radiation diffusion computations dominate the overall execution time, it is essential to solve these equations quickly\(^3\). Second, in one to two years, they would like to incorporate their ALE technology into an AMR framework. AMR offers ALE higher spatial resolution as well as local time refinement.

We foresee a number of interesting, albeit very difficult, problems integrating ALE with AMR. Traditionally, AMR has used Eulerian integration approaches structured around fluxes computed across cell boundaries. The Eulerian approach has the advantage of a relatively simple flux conservation scheme at coarse-fine interfaces for time refinement. Viewing the numerical integration procedures in this context facilitates the enforcement of mass conservation over the composite grid configuration generated during the AMR process. In contrast, ALE is based on a Lagrangian moving mesh approach with occasional re-mapping of the grid.

Using the ALE integration method within AMR will necessitate the complete re-design of the traditional AMR flux conservation scheme at coarse-fine grid interfaces. As described in Section 3, AMR computes boundary conditions for fine grids by time-interpolating coarse grid data (assuming, of course, that no fine grid data or physical domain boundary values are present). During the time integration of the fine grid, time-space flux integrals are accumulated at coarse-fine interfaces. When the fine grids has been advanced to the time level of the underlying coarse grids, coarse grid cells adjoining fine grid cells are updated with the new fine grid fluxes. This approach guarantees conservation across time and space scales.

\(^3\)The scalable linear systems effort in CASC is currently collaborating with the ARES code group on the development of new preconditioning techniques to accelerate the solution of the radiation diffusion equations.
However, ALE does not use a flux-based integration scheme; therefore, another type of conservation method must be developed to synchronize the fine and coarse grids. A further complication is that coarse grid cells will move during the coarse grid advance. Thus, filling fine grid ghost cells will require time-interpolation of physical quantities (e.g., density and velocity) and mesh locations to reflect the changing cell geometries. The re-zoning algorithm on the fine grid must also ensure that fine grid cells at coarse-fine interfaces do not become too distorted in relation to the adjoining coarse cells. Recent work at Los Alamos National Laboratory (LANL) on related overlapping grid problems may be useful in the resolution of these issues in AMR [15, 16].

ALE will also require changes in the down-scaling of conserved quantities between fine and coarse scales. In traditional AMR, fine grid cells always nest (in the physical coordinate space) within corresponding coarse cells. However, with ALE, fine grid vertices will move during the time integration. Thus, a more general form of volume-weighted averaging will be needed.

A final research issue lies in the solution of the parabolic radiation-diffusion equations. Here, we anticipate difficulties similar to those of Odyssey. The opacity function used in the diffusion model is a function of temperature, density, and spatial position. It will be heterogeneous and potentially very discontinuous if neighboring materials possess very different properties. This will complicate the application of linear system solution techniques on a composite grid structure if a multigrid approach is used. ALE adds the further complication that the cell geometry is locally irregular and therefore the stencil computations become more involved.

4.3 ParFlow

ParFlow is a porous media flow and transport code originally developed in CCSE at LLNL, with support from the Earth Sciences and Environmental Protection Departments. The code is currently under continuing development within CASC. ParFlow is targeted toward the application of massively parallel computing technology, high-resolution numerical methods, and scalable solution algorithms to problems in water resource management and groundwater remediation. In these applications, numerical simulations are used to answer questions about water quality, recharge and reuse of recycled water (i.e., treated wastewater), impacts of historical agricultural practices (i.e., fertilizers), increased use and exploitation of groundwater resources, and the influence of the ocean on coastal aquifers (i.e., salt water intrusion). Fluid flow models in these application areas are typically characterized by three-dimensional multiphase flow equations with multiple reacting chemical species.

The current development team is comprised of the following individuals:

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chuck Baldwin</td>
<td>CAO/CASC</td>
<td><a href="mailto:baldwin@xanthus.llnl.gov">baldwin@xanthus.llnl.gov</a></td>
</tr>
<tr>
<td>Bill Bosl</td>
<td>CAO/CASC</td>
<td><a href="mailto:wjbosl@odysseus.llnl.gov">wjbosl@odysseus.llnl.gov</a></td>
</tr>
<tr>
<td>Rob Falgout (PI)</td>
<td>CASC</td>
<td><a href="mailto:falgout@bacchus.llnl.gov">falgout@bacchus.llnl.gov</a></td>
</tr>
<tr>
<td>Rich Hornung</td>
<td>CASC</td>
<td><a href="mailto:hornung@orphus.llnl.gov">hornung@orphus.llnl.gov</a></td>
</tr>
<tr>
<td>Steve G. Smith</td>
<td>CAO/CASC</td>
<td><a href="mailto:ssmith@mercury.llnl.gov">ssmith@mercury.llnl.gov</a></td>
</tr>
<tr>
<td>Carol Woodward</td>
<td>CASC</td>
<td><a href="mailto:carol@poseidon.llnl.gov">carol@poseidon.llnl.gov</a></td>
</tr>
</tbody>
</table>

4.3.1 ParFlow Model Equations and Numerical Methods

The general form of flow and transport in porous media begins with a multiphase, multicomponent mass balance of the form

$$\frac{\partial \theta_j \rho_j \omega_j^i}{\partial t} + \nabla \cdot \left[ \theta_j \rho_j \omega_j^i \mathbf{v}_j + D_j^i \right] = I_j^i + R_j^i + S_j^i,$$

where $i$ is a fluid component index and $j$ is a fluid phase index. Here, $\theta_j$ is the phase volume fraction, $\rho_j$ is the phase density, and $\omega_j^i$ is the mass fraction of component $i$ in phase $j$. The flux term includes the
volumetric phase velocity $v_j$ and a non-convective flux $D_j^i$ modeling diffusive and dispersive transport. The terms on the right-hand side include mass transfer $I_j^i$, species reaction $R_j^i$, and an external source term $S_j^i$. Generally, the mass equations are subject to a constraint that the fluid fills the void space in the porous matrix.

Many simplifications to these equations are adopted when developing application-specific models and numerical methods. First, the standard multiphase, multidimensional extension of Darcy’s law replaces general momentum balance considerations in most cases. Darcy’s law relates the total volumetric flow rate of each phase $v_j$ to the phase pressure gradient and gravitational forces:

$$v_j = -K\lambda_j[\nabla p_j - \rho_j g]. \quad (6)$$

Here, the phase mobility (ratio of relative permeability to viscosity) is $\lambda_j$, and $p_j$ is phase pressure. Second, issues surrounding general energy balance are usually ignored completely in groundwater models. However, if the physical system is non-isothermal (e.g., in steam flooding processes), some form of energy balance must be present. Third, once relevant fluid and rock properties are identified, empirically determined equation-of-state models (pressure-volume-temperature relationships), and constitutive relations (pressure-saturation-relative permeability relationships) are employed so that the system of equations is closed and all unknowns can be determined. The aforementioned pore volume constraint is typically the unifying principle behind the various relationships. Almost universally, the issue is not whether such simplifications are necessary, but which are appropriate when considering (the often competing factors) of model accuracy and computational expediency [46].

The general mass balance equations are simplified in other ways when deemed appropriate for the application at hand. In one case, the equations are summed over components to derive a multiphase form in which individual phase composition is ignored:

$$\frac{\partial \theta_j \rho_j}{\partial t} + \nabla \cdot [\theta_j \rho_j v_j] = I_j + S_j. \quad (7)$$

Here several identities have been applied: $\sum_i \omega_j^i = 1$, $\sum_i D_j^i = 0$, $\sum_i R_j^i = 0$. Although this multiphase form simplifies mass balance considerably, the equations can still be quite complicated as many important multiphase processes require the description of a variety of diverse phases, including aqueous liquid, non-aqueous liquid, gas, solid, micro-emulsion, etc. In another case, it may be reasonable to assume local chemical and thermodynamic equilibrium exists. This is based on the notion that equilibrium is achieved much more rapidly than transport occurs. In the case of compositional modeling, this assumption allows the component mass conservation equations to be summed over phases. Then, one can consider total mass conservation of each component. To reduce the number of unknowns in a particular system, similar components are often combined to form a smaller collection of pseudo-components. For example, all air components—except perhaps oxygen, which is crucial for most chemical reactions—may be grouped into the gas phase. Alternatively, since some species, particularly those comprising hydrocarbon pollutants, are not significantly soluble in water, they may be lumped together to form the non-aqueous phase.

In most cases, if the model describes incompressible flow, one can obtain an elliptic pressure equation by summing the mass conservation equations and applying the volume balance constraint (in this case, $\sum_i d_j = \phi$, where $\phi$ is porosity). In the compressible case, these equations cannot be simplified so easily. The primary reason for this is that phase densities, and thus the phase volumes, are pressure-dependent. Moreover, the flux terms in the equations may also require complicated phase behavior models to determine the contribution of each fluid phase to the flow of each chemical component. One option is to employ a fully implicit integration approach and apply a quasi-Newton iteration to integrate the equations and satisfy the constraints and constitutive relations simultaneously. However, this precludes the use of an AMR strategy employing local time refinement. An alternative is to linearize the constraint to form a parabolic pressure equation [58, 59]. If the compressibility is substantial, it may be necessary to iterate between the mass and pressure equations until the volume constraint is sufficiently satisfied.

Currently, ParFlow is capable of solving porous media flow problems involving a single incompressible fluid phase containing several chemical constituents that move passively within the fluid phase. The equations
describing single phase, multicomponent processes consist of mass conservation

$$\frac{\partial (\phi \rho c_i)}{\partial t} + \nabla \cdot (c_i \rho \nu_i) = q_i + \nabla \cdot (D \cdot \nabla c_i),$$  \hspace{1cm} (8)$$

where again \( i \) is a component index, single phase Darcy's law

$$v = -K\lambda \cdot (\nabla p - \rho \bar{g}),$$ \hspace{1cm} (9)$$

and simple phase composition equilibrium

$$\sum_i c_i = 1.$$ \hspace{1cm} (10)$$

In this case, the constraint that the fluid fills the pore volume is satisfied trivially, eliminating the need to consider fluid phase saturation. In the mass equation, \( c_i \) is the mass concentration of fluid component \( i \), \( \rho \) is the total fluid density, \( \phi \) is the void volume fraction (i.e., porosity), \( D \) is the hydrodynamic dispersion tensor, and \( q_i \) is a source term for component \( i \).

Currently, the modeling and numerical capabilities in ParFlow are limited so that permeability \( K \) must be a scalar function of spatial position. Generally, it is important to model anisotropic effects in hydrology problems by representing permeability as a tensor quantity where this tensor is at least anisotropic. Also, the numerical methods in ParFlow are incapable of treating the dispersion term in Eq. (8), which is very important to most groundwater flow applications. The dispersion coefficient \( D \) is typically modeled as a full 3 x 3 tensor. The current iterative solver for the incompressible pressure equation utilizes a multigrid preconditioned conjugate gradient (MGCG) method that is very robust and scalable. However, incorporating the additional physics mentioned above will require enhancements to the current linear solver technology. Plans to do so are currently under way.

To solve the above equations, ParFlow employs an IMPES (IMplicit Pressure, Explicit Saturation) numerical process in which the hyperbolic and elliptic parts of the equations are separated and concentrations and fluid pressure are solved for separately. Since the fluid and porous medium are assumed incompressible, \( \rho \) is a constant in time and spatial position, and \( \phi \) is a constant function of time. By summing the mass conservation Eqs. (8) over components, applying phase equilibrium, Eq. (10), then substituting Darcy’s law, the following elliptic equation for pressure is obtained

$$-\nabla \cdot (K\lambda \cdot (\nabla p - \rho \bar{g})) = \frac{1}{\rho} \sum_i q_i = q.$$ \hspace{1cm} (11)$$

Eq. (11) is solved first, then used to construct the fluid velocity field. Finally, the mass conservation equations are integrated by applying an explicit, conservative, finite-difference approximation to Eq. (8). Currently, a second-order Godunov method is used for this purpose. Additional transport schemes, such as streamline methods, are slated to be explored in the near future.

Simple multiphase and compressible flow modeling are also being considered for ParFlow. A Richards’ equation model is being developed to treat variably-saturated flow (i.e., co-existing air and water phases). This model is useful for studying water table fluctuations and water seepage through the unsaturated zone. Richards’ equation can be stated as,

$$\frac{\partial (s \rho \phi)}{\partial t} - \nabla \cdot (K\lambda \cdot (\nabla p - \rho \bar{g})) = q,$$ \hspace{1cm} (12)$$

where \( p \) is the water pressure, \( s(p) \) is the water phase saturation (the fraction of pore space occupied by water), \( \rho(p) \) is the water phase density, \( \phi(p) \) is the porosity of the medium, and \( \lambda(p) \) is the mobility of water. Compressibility effects of both the medium and fluid are contained in the model in that both porosity and density are pressure-dependent. Also, both saturation and relative permeability are nonlinear functions of pressure. To approximate Richards’ equation, a cell-centered finite difference scheme equivalent to the lowest order mixed finite element method is used. For the time discretization, backward Euler time stepping is applied. This method of discretizing Richards’ equation leads to a system of discrete nonlinear equations.
Further enhancements to study more complicated compressible flow and multiphase transport (mentioned above) require more involved numerical solution procedures. Multiphase flow processes exhibit behavior typical of the solutions to both hyperbolic and elliptic/parabolic partial differential equations. For example, pressure changes are felt quickly throughout the reservoir when the flow is incompressible or only slightly compressible. In contrast, fluid fronts (interfaces) tend to move with much slower speeds and can be fairly sharp. However, the fronts may be spread by dispersive mixing and capillary effects. In any event, a sophisticated combination of complicated numerical methods is required to efficiently and accurately treat porous media problems involving a variety of physical and chemical mechanisms. The numerical issues are made significantly more complex when AMR is introduced. The explicit, conservative method used to integrate the non-dispersive transport equation can easily be incorporated into the traditional AMR algorithm. However, the coupling of the transport to the flow in these problems adds considerable complexity to the incorporation of AMR. These issues have been explored with some success in Hornung et al. [30, 31]. There are many interesting numerical and computational science challenges associated with the resolution of these issues. Moreover, problems in this area serve as an excellent testing ground for the application of AMR because of their inherent numerical complexity.

4.3.2 ParFlow Code Structure

ParFlow is implemented in C and supports a considerable amount of modularity allowing the user to build different application-specific implementations of ParFlow by editing a call-graph. It is also designed to run on a wide range of computing platforms, ranging from desktop PCs to massively parallel supercomputers. The data structures in ParFlow support storage of data in logically-rectangular arrays corresponding to parallelepiped regions in an abstract index space. The implementation of numerical routines in ParFlow exploits this notion. ParFlow is built using a sophisticated message passing layer (called AMPS) that allows developers to implement numerical operations in a distributed memory environment in terms of stencil operations.

4.3.3 Potential Contribution to ParFlow

The numerical treatment of field-scale simulation of flow through porous media is generally computationally expensive and substantially under-resolved. Conventional simulations do not adequately resolve important flow features for several reasons. These difficulties illuminate the need for AMR as well as massively parallel computing resources. First, much is unknown about the influence of fine-scale heterogeneity and physical processes on macroscopic flow and transport behavior. Often, it is not practical to incorporate sufficiently detailed domain information into a simulator, nor is it even clear how to do this in many cases. Some sort of adaptive resolution capability appears to be essential to approach necessary resolution in field-scale simulations. Second, when a multiphase fluid mixture is driven through a porous media, complicated fluid interface structures result. Standard, low-resolution, computational methods have difficulty representing detailed fronts (due to reservoir heterogeneity and unstable fluid interfaces) properly on a numerical grid. Adaptive local refinement allows the concentration of numerical effort where it is needed to resolve the important flow features. Third, porous media flow often involves important information on many physical scales. Representing physically meaningful data on various length and time scales efficiently during a numerical simulation is often an unwieldy task. A flexible, robust AMR environment will encourage significant computational research in this area. Understanding the issues surrounding numerical simulation on a hierarchy of computational scales in AMR forces one to consider issues related to understanding the underlying physical phenomena on different scales. ParFlow has potential to make strong contributions toward the resolution of some of these issues in the some important application areas. The incorporation of ParFlow into a suitable AMR environment will strengthen this potential.

A general AMR framework will support numerical and algorithmic research into scalable linear and non-linear solvers as well as integration methods for problems of mixed type associated with ParFlow applications. The development of robust, efficient multigrid preconditioned conjugate gradient linear solvers for problems
posed on composite grid structures will be a primary concern. In addition, more robust multigrid-based algorithms are needed to address the introduction of new physical processes such as tensor permeability and dispersion coefficients. It will also be beneficial to explore the combination of AMR with scalable and efficient nonlinear solution strategies to treat Richards' equation. Both pursuits will potentially benefit several ASCI code efforts included in this survey. Finally, we mention the combination of AMR with alternative transport methods for incompressible, single-phase flow problems in ParFlow. Currently, a temporal subcycling algorithm is being developed and a streamline transport method is being considered for use in ParFlow. The goal of temporal subcycling is to employ explicit time integration methods for hyperbolic conservation laws (e.g., Godunov's method) more efficiently by applying the Courant-Friedrichs-Lewy (CFL) stability restriction locally rather than globally. Streamline methods decouple the transport grid from the Cartesian flow grid. For certain problems, this may allow reasonably accurate transport simulations orders of magnitude faster than those using grid-based methods. Increasing the efficiency of AMR by combining it with either of these transport capabilities is an interesting research prospect with potential benefits to other Laboratory simulation efforts.

4.4 Ardra

Ardra solves the steady-state Boltzmann transport equation that models the transport of neutrons through an isotropically scattering background medium. It was originally developed by researchers in the Numerical Mathematics Group (NMG) in CCSE at LLNL, but the code is now under continuing development within CASC. Ardra has been designed to be a general-purpose neutron transport code applicable to several important problems in this area, including both fission and fusion reactors, nuclear well logging in oil and gas exploration, and medical treatments. Currently, only well logging problems can be run, since this is the only type of problem for which geometry and cross-section generators have been implemented.

The current development team is comprised of the following individuals:

Peter N. Brown (PI) CASC brown42@llnl.gov
Britton Chang CASC changl@llnl.gov
Milo R. Dorr CCSE/LBNL Milo_Dorr@lbl.gov

4.4.1 Ardra Model Equations and Numerical Methods

Ardra solves the deterministic neutron transport model described by the steady-state Boltzmann transport equation

$$\Omega \cdot \nabla \psi(x, E, \Omega) + \sigma(x, E)\psi(x, E, \Omega) = \int_0^\infty \sigma(x, E' \rightarrow E) \int_{S^2} \psi(x, E', \Omega') d\Omega' dE' + q(x, E, \Omega),$$

where \( x \in D \), a region in \( \mathbb{R}^3 \), and \( \Omega \in S^2 \), the unit sphere in \( \mathbb{R}^3 \). This equation models the transport of neutrons through an isotropically scattering background material. The unknown quantity \( \psi(x, E, \Omega) \) represents the flux of neutrons at \( x \) moving in direction \( \Omega \) with energy \( E \). Appropriate flux boundary conditions are required for \( \psi \) on the boundary of \( D \). The total and scattering cross-sections are \( \sigma(x, E) \) and \( \sigma(x, E' \rightarrow E) \), respectively. The external, volumetric neutron source is \( q(x, E, \Omega) \). Ardra currently allows only down-scattering from higher energy groups to lower energy groups; this limitation is appropriate for application domains, such as nuclear well logging, for which energetic neutrons are created only at neutron sources \( q \).

The neutron transport equation is discretized using a multigroup, discrete ordinates, diamond difference method [40,42]. In the multigroup approach, the energy \( E \) is restricted to a finite interval partitioned into energy groups

$$E_{\text{max}} = E_0 > E_1 > \cdots > E_G = E_{\text{min}}.$$
The discrete ordinates approximation is discretized in angle by selecting a finite set of points \( \Omega_{m}, m = 1, \ldots, M \) on the unit sphere and corresponding quadrature weights \( w_{m} \). The diamond difference operator provides an approximation for the gradient of \( \psi \).

One drawback to the discrete ordinates approximation is the ray effects that arise in problems with small scattering ratios and localized sources, such as in the simulation of nuclear well logging tools. These ray effects are caused because the discrete ordinates method models neutron fluxes in a finite number of discrete directions that may be unable to resolve highly localized sources. Ardra eliminates ray artifacts through the use of a harmonic projection technique developed by Brown and Dorr [10]. This approach generates solutions that match the quality of a spherical harmonic basis but at the lower cost of the discrete ordinates approximation.

The multigroup, discrete ordinates equations are solved by iterating between the differential and integral terms using a process called source iteration. Within each energy group, source iteration uses the current iterate \( \psi^{i} \) to evaluate the right hand side of Eq. (13) to obtain what is called a “scattering source.” The left-hand side of the Boltzmann equation is then solved for \( \psi^{i+1} \) using this scattering source as the right-hand side. Source iterations are performed until convergence. Brown et al. [3, 11] have developed a Diffusion Synthetic Acceleration (DSA) preconditioner to accelerate the source iteration process; however, Ardra does not yet implement this preconditioner.

4.4.2 Ardra Code Structure

Ardra is implemented in C and appears to be fairly modular. The code uses a communication layer to insulate the numerical algorithms from machine-specific details. In principle, Ardra can be extended to new applications by writing a set of well-defined functions describing problem geometry and scattering cross-sections; these changes should not affect the main parts of Ardra that implement the general discrete ordinates, multigroup algorithm.

Ardra is unique in that it has been parallelized in space, angle, and energy groups [23]. Previous efforts have focused on exploiting concurrency for only an individual variable (e.g., space) or pairs of variables (e.g., space and direction). To obtain scalability with respect to each of the phase space variables, the linear systems solvers in Ardra have been designed to exploit all concurrency with respect to space, angle, and energy groups. This is necessary to expose all possible parallelism for a given problem and to provide flexibility to treat a wider range of applications. Ardra performs in a scalable manner on a wide range of distributed memory computers ranging from workstation clusters to massively parallel computers [23].

4.4.3 Potential Contribution to Ardra

The Ardra development group believes that some form of local spatial refinement would play a useful role in neutron transport simulations. For example, AMR refinement patches could be used in nuclear well logging modeling to better resolve neutron scattering near the neutron source and detectors in the borehole sonde. High local resolution is also needed at some material interfaces, as neutron scattering at material interfaces is determined by the accurate reconstruction of the interface in the spatial discretization. Local refinement may also be useful for proposed future applications of the Ardra simulator, such as radiation transport in clouds or medical imaging.

Jessee et al. [33] describe preliminary work integrating steady state neutron transport into an AMR framework. They solve the Boltzmann equations in two dimensions using a discrete ordinates method (without harmonic projection) with a single energy group. Their computational results indicate that AMR produces better solutions than a uniform grid approach given fixed memory requirements. However, their naive composite grid solver was nearly five times slower than the corresponding uniform grid solver for the same overall accuracy. These performance numbers indicate that further research is needed into better solution methods than simple source iteration on composite grid hierarchies, especially for the more complex
coupled multigroup equations. For example, extensions of the DAS preconditioner [3, 11] could accelerate solutions for diffusive regimes on AMR grids. Better solver techniques for the neutron transport equations would also impact the AMTRAN neutronics code in B Division.

4.5 Hybrid Applications

Hybrid methods incorporate multiple physical models within the same application for problems for which a single model would be inappropriate. For this reason, they are often referred to as multiphysics methods. A typical hybrid application models processes on fine and coarse length scales using different equations. Fine scale equations may not accurately represent the physics on the coarser scales; alternatively, they may be too expensive for solution over the entire computational domain. AMR technology can be used to marry the two scales, since it supports both the spatial and temporal refinement to communicate between the coarse and fine regimes.

One interesting approach for computational fluid dynamics combines the Navier-Stokes equations with a discrete particle method (e.g., Direct Simulation Monte Carlo—DSMC) in a single simulation [26]. Such hybrid methods are appropriate for situations in which the standard partial differential equation description of flow is not accurate in a localized region. In other words, important phenomena co-exist on very disparate length scales that cannot be modeled using the same equations. Such applications include read/write head movement over the platter in a computer disk drive, a blunt body entering the atmosphere, or boundary layers. We are collaborating in this area with Alejandro Garcia, a physicist at San Jose State University, and Berni Alder at LLNL.

Another hybrid application involves shear band formation and growth in elastic/plastic solids. Xabier Garaizar, an applied mathematician who will join our AMR effort in CASC, has combined an elastic/plastic continuum model with a discrete constitutive relation to study shear band formation [25]. His approach extends the standard AMR algorithm with special time stepping and front tracking needed to follow the evolution of the shear band.

Although hybrid methods are still an area of active research, there are indications that multiphysics applications will become increasingly important to LLNL as computational scientists begin to incorporate more sophisticated physics into their simulation codes. Materials scientists are investigating the integration of continuum and atomistic models for applications in defects in solids [56] and crack propagation [37]. Such multiphysics simulation techniques may be of future interest to H Division and to the Chemistry and Material Science Division.

4.5.1 Model Equations and Numerical Methods

We begin this section with a description of the Direct Simulation Monte Carlo/Navier-Stokes (DSMC/NS) approach to illustrate some of the salient characteristics of the method and hybrid applications in general. In DSMC/NS, hydrodynamic flows are described on the continuum scale by the Navier-Stokes equations or approximations to them, such as the Euler equations for inviscid flow. Such a description is appropriate for cases in which a microscopic representation of the collection of fluid molecules is unnecessary. However, the Navier-Stokes formulation breaks down when physical mechanisms begin to operate on characteristic length scales comparable to the mean free path for gas molecules. DSMC/NS is targeted toward applications for which a molecular description of the dynamics is imperative. Examples of such systems include certain shock phenomena, flows around spacecraft re-entry vehicles, and micron-scale flows associated with air slider bearings in a disk drive.

The Navier-Stokes equations model the conservation of mass density $\rho$, momentum density $p = \rho v$, and
energy density $E$ according to the system of conservation laws

\[
\begin{align*}
\frac{\partial \rho}{\partial t} &+ \nabla \cdot (\rho v) = 0 \\
\frac{\partial \rho v_i}{\partial t} &+ \nabla \cdot (\rho v_i v + T_i) = 0, \quad i = 1, \ldots, 3 \\
\frac{\partial E}{\partial t} &+ \nabla \cdot ((E + P)v - \tau v + q) = 0
\end{align*}
\]  

(15)

Here, $P$ is pressure, $q$ is the heat flux vector, and $T_i = T e_i$ ($e_i$ is the axis vector in the $i$-th coordinate direction) is the stress tensor. According to Cauchy's stress principle, the $ij$ entry in $T$ is $T_{ij} = P \delta_{ij} - \tau_{ij}$, where $\tau_{ij}$ is the shear stress tensor and $\delta_{ij}$ is the Kronecker delta function. Initial plans are to employ MacCormack's method to integrate the equations. The MacCormack method uses a predictor-corrector formulation by which the fluxes are centered in both space and time.

DSMC is a particle-based scheme for integrating the nonlinear Boltzmann equation [8]; it represents the state of the system by the positions and velocities of a collection of particles. Unlike molecular dynamics, in which successive particle collisions are calculated exactly using explicit interaction potentials, DSMC generates particle interactions stochastically using scattering rates and post-collision velocity distributions determined from kinetic gas theory. Each particle typically represents thousands of gas molecules. Thus, DSMC amounts to solving the Boltzmann equation using a representative sample from the velocity distribution.

DSMC is an accurate simulation technique if the dimensions of "collision cells" (into which the particles are sorted and evolved) are less than a mean free path and the time step is chosen to be less than the mean collision time. The computational expense corresponding to these constraints limits the application of the algorithm to microscopic systems or highly rarefied gas flows. Thus, combining DSMC with a continuum-based approach in the context of AMR is extremely attractive since it increases the applicability of the particle approach.

The DSMC/NS/AMR research is directed towards developing strongly coupled schemes for which the particle and continuum methods simultaneously evaluate different regions of the flow and continuously exchange information across an interface separating the disparate physical scales. The particle to continuum communication requires the computation of continuum quantities as averages of particle data. So, for example, the mass, momentum, and energy densities associated with a statistics cell of volume $V_c$ are defined as

\[
\begin{bmatrix}
\rho \\
p \\
e
\end{bmatrix} = \frac{1}{V} \sum \begin{bmatrix}
m \\
mv \\
\frac{1}{2}m||v||^2
\end{bmatrix},
\]

where the sum is over particles in the cell [26]. Kinetic theory dictates how continuum properties can be used to construct particle information. Particles are constructed in an interface region in the continuum near the DSMC/NS interface. Random velocities are assigned to particles from continuum data according to the Chapman-Enskog distribution [14]. These boundary particles are advanced along with the particles in the DSMC region. Flux across the continuum-discrete interface is carefully tabulated during the advance. These aggregate fluxes are then used to update the continuum region to maintain conservation.

Another interesting application of hybrid modeling techniques with AMR is shear band formation and growth in solids and granular materials. Garaizar has combined front-tracking with AMR along with a procedure for advancing the ends of the band in an elastic/plastic solid [25]. Tracking provides accurate evolution of the band when coupled to special techniques for advancing the ends of the band governed by the loss of hyperbolicity in the equations. The equations describing the antiplane shear model are hyperbolic except at some critical stress values. There, a criterion of ill-posedness and a constitutive relation are used to model shear band evolution.

AMR is useful in this application for two reasons. First, deformations are mostly uniform except in localized regions of high stress. Second, a precise description of the interaction between the hyperbolic and non-hyperbolic flow regions is needed. The interaction of relief fronts and shear bands is very complicated and potentially computationally expensive. The localization of deformation behavior indicating the onset of
shear bands allows computational effort to be localized via local grid refinement. As a result, complicated physical and numerical models can be explored in more practical situations. The information gained from this work can contribute to related simulations in a wide range of materials science problem areas.

4.5.2 Hybrid Code Requirements

Hybrid models are still very much in formative stages of development. Scientists are still pursuing very basic research into physical modeling and algorithmic design. Therefore, it is difficult to assess specific needs for integrating hybrid codes into AMR.

However, we can make two general statements regarding software and framework requirements. First, the framework must readily support new types of patch-based data structures. It would be unrealistic to expect that the standard cell-centered or edge-centered storage typically used in AMR would be sufficient for complex multiphysics applications. For example, the particles needed by the DSMC/NS method require radically different types of data structures than simple arrays. Support of such complex data is complicated by parallelization requirements, since the framework must understand how to marshal and unmarshal data for communication across processor memory spaces.

Second, the framework must be sufficiently general to allow algorithmic experimentation. Clearly, hybrid codes will require different types of algorithms than traditional hyperbolic AMR. Scientists will need an environment that enables them to experiment with various forms of coupling between physics models; therefore, the framework should facilitate rapid prototyping of new approaches and new algorithms.

4.5.3 Potential Contribution to Hybrid Applications

We are developing an AMR software framework to enable investigation into different numerical methods and design tradeoffs for hybrid applications. Without such a prototyping environment, scientists would be unable to rapidly explore new algorithms. Instead, each new approach would require an unrealistic investment in new code implementation. A well-designed framework, however, provides a solid foundation for experimentation, as much of the required software is already supplied by framework routines. The DSMC/NS application described above is being used as one proof-of-principle code to test our software design. More discussion of our proposed AMR framework can be found in Section 5.

4.6 Summary of Outstanding Research Issues

Computational scientists are now beginning to explore the extensions of AMR simulation technology into new applications domains. However, these new approaches raise a number of interesting research questions that we identified in previous sections. Here we present a brief summary of these research issues.

- How do we couple hyperbolic equations with either parabolic or elliptic equations within the AMR time refinement methodology? Coupling equations in an operator-split fashion becomes difficult because the solution of parabolic or elliptic equations requires global information at a single simulation time. However, in AMR, the various levels of the composite grid hierarchy are at different simulation times due to local time refinement. Simple linear time interpolation may not be appropriate for problems with highly nonlinear character.

- How do we efficiently solve discontinuous coefficient linear (and nonlinear) systems on a composite grid hierarchy? The linear systems of equations that arise in ParFlow, Odyssey, and ARES are difficult to solve using fast solvers such as multigrid because of highly discontinuous coefficients. Equations for more complex physical models may include tensors in place of scalars, further complicating the solution process. Many problems still require further research into efficient and accurate discretizations for the interfaces between coarse and fine grids of an AMR hierarchy.
How do we integrate ALE techniques into an AMR algorithmic framework? AMR uses Eulerian time integration methods based on computing fluxes across cell boundaries. Synchronization between levels during the standard AMR time refinement process assumes this type of flux-based conservation scheme. In contrast, ALE uses a Lagrangian moving mesh approach and is not formulated in terms of fluxes across interfaces. Combining AMR and ALE will therefore require the development of alternative conservation procedures.

How do we implement AMR in an efficient manner on large distributed memory machines such as ASCI Blue Pacific? The patterns of message sends and receives required for interprocessor communication are irregular and change during the numerical simulation. Hybrid applications require sophisticated communications support to manage the packing and unpacking of complex data structures (e.g., particles) into and out of message streams. Furthermore, operator-split solvers will create problems for balancing workloads across the processors of a large parallel machine. The data decompositions appropriate for an explicit hyperbolic advance may be very different from that needed for an implicit solve of an elliptic equation.

How do we design software frameworks to facilitate the exploration of alternative numerical methods and time integration algorithms? Although the various applications described in this section are quite different, common characteristics can be captured and exploited by a software support framework. All applications use a form of patch-based structured refinement. All have similar notions of ghost cells and nearest-neighbor communication. The primary software design issue is how to factor out these common elements and yet provide the flexibility to customize the framework for the needs of a specific application.

How do we couple different physical models for hybrid applications using AMR techniques? Proper coupling between the various mathematical models in hybrid applications is still an area of active research. Our AMR software framework will facilitate rapid prototyping of new numerical approaches and new algorithms for hybrid applications.

5 Research and Development within CASC

Structured AMR has been a highly successful simulation methodology for hyperbolic conservation laws that can be treated with explicit, conservative finite difference methods. Computational scientists are building upon the traditional structured AMR algorithm by extending its best features to address new problem areas. There are many interesting algorithmic and numerical research questions associated with such extensions. Clearly, these new approaches will require new solvers (e.g., for composite grid equations with discontinuous coefficients), new time integration methods (e.g., for tightly coupled systems of equations), and new patch-based data structures (e.g., for particles).

Building AMR applications is difficult, and a software support infrastructure will be needed to facilitate numerical and algorithmic experimentation. Such an infrastructure will enable rapid prototyping of various design alternatives by freeing scientists from low-level data structure management and other implementation details. Well-designed frameworks reduce code duplication, encourage interoperability of applications software, and simplify the learning curve for new computational methods.

After studying the future needs of AMR applications and evaluating existing AMR frameworks, we have begun within CASC the development of an AMR software framework that will be sufficiently flexible and extensible to address the identified research goals of the previous section. In the following sections, we briefly describe other AMR software efforts and outline the challenges for the CASC AMR effort.
5.1 Existing Software Frameworks for AMR

An important component of this survey was to understand existing AMR software packages and how such software technology could be applied to ASCI codes. We have evaluated three AMR frameworks and two parallelization libraries. The three AMR frameworks are BoxLib\(^4\) [19], AMR++\(^5\), and John Trangenstein's AMR infrastructure [31, 57]. The two parallelization libraries are KeLP\(^6\) [24] and DAGH\(^7\) [50]. While a detailed description of these packages is beyond the scope of this survey, in the following we briefly introduce each package and discuss the various approaches and philosophies.

BoxLib is a library of C++ classes designed to facilitate the development of block-structured finite difference algorithms for systems of partial differential equations. It is used extensively by scientists in the Center for Computational Science and Engineering (CCSE) at LBNL. BoxLib provides two basic abstractions that are common to most adaptive mesh refinement software: a Box and a FAB (Fortran Array Box). The Box represents a rectangular region in the logical index space used to define AMR patches. The FAB is an array with a fixed number of components (used to represent fluid variables such as pressure or mass) defined over the index space of a Box. BoxLib also defines numerous box calculus operations, such as union and intersection, over boxes and lists of boxes. BoxLib itself does not provide support for higher-level AMR functionality, such as time integration of the hierarchy or filling of ghost cell boundary data; these operations are managed by the application software instead of the framework. We believe that much of the software support that BoxLib currently implements at the applications level can be abstracted and packaged into the AMR support framework. Providing algorithmic support in the framework itself simplifies code development by promoting design and code reuse.

AMR++ is a collection of C++ classes for structured AMR. The software support in AMR++ consists of five principal sublibraries: (1) interpolation and projection libraries to communicate between levels of the AMR hierarchy, (2) grid objects based on Overture\(^8\) [15], a library for solving PDEs on complicated mapped domains, (3) an adaptive regridding library that manages grid generation using BoxLib, (4) an AMR solver library of templated solver objects, and (5) graphics support. The Overture grids are based on the A++ and P++ array class libraries developed by Quinlan [39, 51]. P++ supports fine-grain data parallel operations on arrays distributed across collections of processors; it automatically manages data decomposition, interprocessor communication, and synchronization. A++ and P++ are being integrated into the POOMA [4] framework.

The various levels of software infrastructure within the AMR++ and Overture frameworks are impressive and extremely useful for the types of overlapping grid computations being addressed by the Scientific Computing Group at LANL. However, we believe that the software complexity associated with these projects is largely unnecessary for our classes of applications. We aim to provide a simple and general collection of C++ class abstractions to facilitate interplay between complicated algorithmic structures (most easily represented via class derivation and composition) and efficient Fortran numerical kernels. We also feel it is important to be able to incorporate legacy Fortran numerical routines into a parallel AMR application. It would be difficult to support our general patch-based data structures in an A++ framework, since A++ only implements arrays of floating point numbers and integers and does not support more general data types (e.g., for treating particle data).

The AMR infrastructure developed by John Trangenstein at Duke University is probably the most general software framework of the three we studied. It has been applied to a variety of diverse applications, including porous media flow [30, 31], nonlinear solid mechanics [57], and shear band formation [25]. Trangenstein's framework employs many of the same "box" concepts as BoxLib; the primary differences are that Trangenstein's AMR software provides the scientist with more freedom in specifying patch-based data structures and it defines an algorithmic framework for AMR time integration that can be specialized through inheritance.

\(^4\)The BoxLib software is available from http://www.nersc.gov/research/CCSE/software/software.html.
\(^6\)KeLP is available through http://www-cse.ucsd.edu/groups/hpcl/scg/kelp.html.
\(^7\)DAGH is available from ftp://helmholtz.ph.utexas.edu/pub/parashar/DAGH/.
\(^8\)Information about Overture is available at http://www.cs.c3.lanl.gov/~benschow/Overture/Overture.html.
Indeed, the flexibility and generality offered by Trangenstein's framework has greatly influenced our own AMR framework design. We aim to build on the best features of this approach by developing a cleaner object-oriented design to allow for more general algorithmic and numerical exploration.

KeLP is not an AMR framework but rather a support library for parallelizing block-structured applications. KeLP provides powerful mechanisms that manage data decomposition and interprocessor communication for irregular, dynamic, block-structured applications running on parallel architectures. KeLP has been used to parallelize an adaptive mesh application for electronic structure calculations [38] along with more structured applications, such as multigrid [24]. The KeLP parallelization model assigns each block (or patch in an AMR application) to a single processor. Numerical computation on each patch is managed by serial Fortran, C, or C++ kernels. This is in contrast to P++, which automatically decomposes array data and requires that computational work be performed using the array operations of P++.

Like KeLP, DAGH (Distributed Adaptive Grid Hierarchy) is a data management infrastructure for parallel AMR applications. DAGH manages data decomposition using a spacefilling curve partitioning that attempts to maintain data locality vertically through the AMR hierarchy. The DAGH support software and the spacefilling curve mappings have been used in a variety of applications, ranging from relativistic and astrophysical hydrodynamics to multi-resolution spatial databases.

### 5.2 CASC Research and Development

The research issues described in Section 4.6 drive three primary design goals needed for an AMR software framework to support future ASCI and Laboratory applications. These three design goals are as follows.

- **The framework must support applications that may require multiple solution methods.** Many ASCI applications model complicated physical processes described by systems of partial differential equations nonlinearly coupled in complex ways. Often, the numerical solution of the equations is facilitated by using operator-splitting approaches by which the original system is decomposed into a collection of sub-problems, each of which may be integrated using a fundamentally different numerical method (e.g., implicit vs. explicit integration). Many interesting research questions are associated with such numerical integration algorithms independent of AMR. The application of these methods in the context of AMR exposes additional avenues of numerical algorithm research.

- **The software must run efficiently in parallel on the ASCI machines.** Proposed ASCI simulations will be extremely computationally intensive. To reap the theoretical benefits of AMR in large-scale simulations, the underlying AMR support infrastructure must parallelize efficiently and scale to large numbers of processors. Furthermore, the single-node performance of an application written within the infrastructure should be comparable to Fortran. Efficient parallel programs require efficient numerical routines, and a significant loss of performance in numerical computation is unacceptable.

- **The framework must be easily extensible to address new numerical approaches and simulation methods.** An extensible support framework will be essential as scientists begin to study increasingly complicated physical systems using AMR simulation technology. To be successful, the framework must be flexible enough to support patch-based data structures more complex than simple grids (e.g., particle chaining meshes). Furthermore, it must facilitate algorithmic exploration without requiring major modifications to the existing code framework.

Accomplishing these goals requires a careful software architecture design. Existing software packages are limited in their flexibility and support for algorithmic frameworks. ASCI applications require the investigation of new solution algorithms; thus, it is important to separate the adaptive hierarchy management from the specification of the time integration and solver algorithms.

Our proposed framework involves a substantial re-organization and extension of what we view as the most desirable features of existing software technologies. However, we will be able to borrow many ideas and some
software technology from existing frameworks. For example, our box calculus is similar to what is used in BoxLib, KeLP, and Trangenstein's framework. Our algorithmic support is an extension and generalization of what is provided by Trangenstein. Our model of patch-based parallelism and interprocessor communication is based on KeLP.

It would be unrealistic to expect current projects to abandon their existing software frameworks. Instead, our AMR software framework will provide a prototyping environment to investigate various numerical and algorithmic extensions of current AMR technology. We expect that information on solver performance, numerical methods, and parallelization approaches gained through the use of our AMR prototyping software will be disseminated to Laboratory code groups and the general AMR community. Furthermore, future code efforts at the Laboratory would be able to start with our framework as a basic foundation and thus increase code and design reuse, reducing the overall time of software development.

A Radiation Transport-Hydrodynamics Model Equations

In this Appendix, we describe the basic radiation transport and hydrodynamics equations used in the Odyssey and ARES ASCI applications. As noted earlier, the ASCI codes that we view as suitable candidates for AMR are designed for radiation transport and hydrodynamics simulations. These physical processes are represented by a coupled system of time-dependent nonlinear partial differential equations, typically. The hydrodynamics model is composed of a set of equations modeling conservation of mass, momentum, and energy. Radiation transport equations model the evolution of radiation energy (in its many forms) and the interaction of this energy with matter in the system.

A.1 Hydrodynamics

In describing the hydrodynamics equations, we follow the presentation in [9]. In particular, the equations are formulated in a Cartesian coordinate system with respect to a spatial frame of reference moving with velocity \( v_g \). If \( v_g = 0 \), this is referred to as the laboratory frame of reference and the standard form of the Eulerian equations of motion are recovered. If \( v_g = v \) (the overall flow velocity) then we recover the standard Lagrangian form of the equations. During a computation, \( v_g \) may be the grid velocity associated with a moving mesh.

The conservation of mass is represented by the equation

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v - v_g) = -\rho \nabla \cdot v_g ,
\]

where \( \rho \) is the fluid density and \( v \) is its velocity. Momentum conservation (derived from Newton's second law of motion) is modeled by

\[
\frac{\partial (\rho v^k)}{\partial t} + \nabla \cdot (\rho v^k (v - v_g)) = -\frac{\partial P}{\partial x^k} + f^k - \rho v^k \nabla \cdot v_g \]

for each coordinate direction, \( k = 1, 2, 3 \). Here, \( P \) is the fluid pressure. The body force \( f^k \) includes the influence of magnetic forces, gravity, and radiation-matter coupling. Energy conservation (derived from the first law of thermodynamics) is represented as

\[
\frac{\partial (\rho e)}{\partial t} + \nabla \cdot (\rho e (v - v_g)) = -P \nabla \cdot v - \rho e \cdot v_g + \frac{dq}{dt} .
\]

Here, \( e \) is the material internal energy density. The last term on the right-hand side is the contribution of non-adiabatic internal energy sources (see discussion of radiation transport below). The internal energy term \( \rho e \) is related to the total energy density \( E \) by the equation

\[
\rho e = E - \frac{1}{2} \rho |v|^2 .
\]
In terms of the total energy density, the energy conservation equation takes the form

$$\frac{\partial E}{\partial t} + \nabla \cdot (E (v - v_g)) = -\nabla \cdot (P v) + v \cdot f - E \nabla \cdot v_g .$$

(21)

In this form, the energy equation more clearly represents the fact that the change in total energy is equal to the rate of work done to the fluid by external mechanical processes. However, the internal energy form is usually adopted for reasons mentioned below.

Notice that the presence of the $P \nabla \cdot v$ term in the internal energy formulation implies that the equation is not in conservation form. This can present problems in the development of useful numerical methods. But, often this effect can be minimized. The total energy form, on the other hand, is in conservation form. Thus, it is possible to develop conservative numerical methods to integrate the equation. However, often it is not clear how to develop methods including additional physics, such as radiation energy, gravitational effects, and magnetic fields, for this formulation. Thus, the internal energy formulation is usually adopted since the advantages due to modeling these other physical mechanisms are considered to outweigh the strict enforcement of energy conservation in most applications.

The inviscid form of the hydrodynamic flow equations (Eqs. (17) - (19)) involve five equations and 6 variables describing the evolution of fluid density $\rho$ (1 equation), momentum $\rho v$ (3 equations), and internal energy $e$ (1 equation). The system of equations is closed by specifying a thermodynamic equation of state model $P = P(e, \rho)$. For example, the equation of state in ideal gas dynamics is $P = (\gamma - 1)\rho e$, where $\gamma$ is a ratio of specific heats [18].

Since heat conduction and viscosity are absent from the system of equations, the model includes only adiabatic processes (i.e., specific entropy is assumed to remain constant). In general, material temperature is a function of density and internal energy $T = T(\rho, e)$. The influence of all dissipative processes is contained in the $\frac{d}{dt}$ term of the internal energy equation. For non-adiabatic effects, we require additional constitutive relations allowing us to describe thermal conduction, radiation transport (e.g., opacities, emission and absorption processes), and compositional changes (e.g., ionization, dissociation, nuclear processes). Usually, the material properties describing these phenomena are modeled as functions of density $\rho$ and material temperature $T$.

Before we outline the equations describing energy transport, we mention the contribution of viscosity to the hydrodynamics equations. Viscous stresses may be due to mechanical properties of the materials involved, or the presence of molecular viscosity where the stress field is proportional to the velocity gradient. If stresses other than pressure exist in the system, then the notion of pressure is extended to include a second-rank tensor $\Sigma$ describing these effects which depend on the direction of fluid movement. The momentum equation becomes

$$\frac{\partial \rho v^k}{\partial t} + \nabla \cdot (\rho v^k (v - v_g)) = -\frac{\partial P}{\partial x^k} + \frac{\partial \Sigma_{ki}}{\partial x^i} + f^k - \rho v^k \nabla \cdot v_g .$$

and the internal energy equation becomes

$$\frac{\partial e}{\partial t} + \nabla \cdot (e (v - v_g)) = -P \nabla \cdot v + \frac{\partial v_i}{\partial x^k} \Sigma_{ik} - e \cdot v_g + \frac{dq}{dt} .$$

(22)

(23)

As before, the $\frac{dq}{dt}$ term represents energy changes due to all dissipative forces apart from viscosity.

In the standard operator splitting numerical treatment, the adiabatic energy changes are computed using the hydrodynamics equations (Eqs. (17) - (19)) and the non-adiabatic (dissipative) changes are calculated separately. The hydrodynamics energy calculation accounts for changes in internal energy density $e$ due to mechanical work associated with pressure changes once the non-adiabatic energy changes are given. Again, it is typical to consider the transport of internal energy since it is unclear how to treat $T$ as though it is advected. During the calculation of non-adiabatic energy changes (including radiation diffusion, emission, adsorption, magnetic diffusion, thermonuclear energy release, etc.) the material density and velocity are assumed to be constant functions of time.

9Note: We have invoked the Einstein summation notation for terms involving the tensor $\Sigma$. 

30
A.2 Radiation Transport and Diffusion

In the standard radiation transport model, the properties of the radiation field are characterized in terms of angular dependence and spectral distribution. That is, in the absence of interaction with matter, radiation is assumed to travel in a straight line and its flux changes only because of geometrical influences. Matter affects the evolution of radiation via emission, adsorption, and scattering. The purpose of the current section is to outline various approximations used to model radiation transport and energy interaction with matter. As in [9], we assume that the material is in thermodynamic equilibrium with itself, and is characterized by temperature $T$. The radiation temperature $T_R$ is assumed to vary significantly over length scales much larger than the atomic collisional mean free paths.

Most generally, the evolution of the radiation field is represented by the Boltzmann transport equation [40]. However, depending on the requirements and complexity of the model at hand, various levels of approximation are adopted to simplify the radiation transport description. Following [9, 40], we briefly develop the transport equation and describe these approximations and the manner in which they couple to the hydrodynamics equations. The standard numerical approach used to solve the coupled radiation and hydrodynamics processes can be the same regardless of the radiation transport model employed. Most commonly, operator splitting (see section A.1) is employed since it allows the hydrodynamics and radiation transport equations to be integrated using different numerical methods. The hydrodynamics equations are integrated (conservation of mechanical variables: $\rho, \rho v, \rho e$) with the radiation energy description assumed constant in time, and the radiation transport is calculated (changes in thermal energy) with $\rho, v$ assumed fixed.

We begin with the general radiation energy description. Let $\epsilon$ be the spectral energy; that is, $\epsilon = h\nu$ where $\nu$ is the frequency and $h$ is Planck's constant. Particle scattering is described by a probability distribution function $f(\epsilon \rightarrow \epsilon', \Omega \cdot \Omega')$ of energy and angle changes so that

$$f(\epsilon \rightarrow \epsilon', \Omega \cdot \Omega')d\epsilon' d\Omega'$$

is the probability that a particle with energy $\epsilon$ traveling in direction $\Omega$ will emerge in an infinitesimal energy interval $d\epsilon'$ about $\epsilon'$ traveling in a solid angle $d\Omega'$ about $\Omega'$. For example, in scattering collisions where exactly one particle is emitted, the function $f$ is weighted so that

$$\int d\epsilon' \int d\Omega' f(\epsilon \rightarrow \epsilon', \Omega \cdot \Omega') = 1.$$  

Assuming that there is only a single nuclide present, we can define the differential scattering cross-section $\sigma$ at a spatial position $\vec{r}$ as

$$\sigma(\vec{r}, \epsilon \rightarrow \epsilon', \Omega \cdot \Omega') = n(\vec{r}) \tilde{\sigma}(\epsilon) f(\epsilon \rightarrow \epsilon', \Omega \cdot \Omega') .$$

Here, $n(\vec{r})$ is the number of nuclei per unit volume at $\vec{r}$, and $\tilde{\sigma}$ is the microscopic cross-section (i.e., effective cross-sectional area per nucleus seen by the particles). We note that to the extent that differential cross-sections are anisotropic, they tend to be peaked in the direction of particle movement. Also, $\Omega \cdot \Omega'$ frequently determines $\epsilon'$ through momentum and energy conservation; thus, $f$ is proportional to a Dirac delta function in particle distribution, usually. Therefore, in the isotropic scattering case, the cross-section is considered to be independent of angle:

$$\sigma(\vec{r}, \epsilon \rightarrow \epsilon', \Omega \cdot \Omega') \sim \sigma(\vec{r}, \epsilon \rightarrow \epsilon') .$$

We need seven independent variables to fully describe radiation transport: spatial position $\vec{r}$ (3), particle energy $\epsilon$ (1), time $t$ (1), and angle components for direction of travel $\Omega$ (2). An especially important quantity is the particle density distribution since all quantities of interest can be determined as a function of this distribution. The particle density distribution $N$ is defined so that

$$N(\vec{r}, \Omega, \epsilon, t) dV d\epsilon d\Omega$$

is the expected number of particles in a volume $dV$ about $\vec{r}$ traveling in a cone of directions $d\Omega$ about angle $\Omega$ with energies in the interval $(\epsilon, \epsilon + d\epsilon)$ at time $t$. Changes in the particle density distribution are due to
three mechanisms primarily: (1) net streaming of particles out of $dV$ across its boundaries, (2) collisions in $dV$ causing particles to be adsorbed or scattered outside the angle-energy element $d\Omega de$, and (3) emission of particles in $dV$ from scattering, fission, or external sources within $d\Omega de$. Next we introduce quantities that are useful in describing these changes.

The specific angular intensity (or angular flux) is

$$\psi(\vec{r}, \Omega, \epsilon, t) = c N(\vec{r}, \Omega, \epsilon, t), \quad (29)$$

where $c$ is the particle speed. The angular flux represents the energy density traveling in direction $\Omega$ that crosses a surface of unit area (whose normal direction is given by $\vec{r}$) in unit time per unit solid angle (i.e., a cone of directions about $\Omega$). The product

$$\sigma(\vec{r}, \epsilon) \psi(\vec{r}, \Omega, \epsilon, t) dV de d\Omega = \sigma(\vec{r}, \epsilon) c N(\vec{r}, \Omega, \epsilon, t) dV de d\Omega \quad (30)$$

represents the total number of reactions per unit time in the phase space element $dV de d\Omega$. The distribution $\psi$ and the point function $\sigma$ are sufficient to characterize the general time-dependent transport equation:

$$\frac{\partial \psi(\vec{r}, \Omega, \epsilon, t)}{\partial t} + c \Omega \cdot \nabla \psi(\vec{r}, \Omega, \epsilon, t) + \sigma(\vec{r}, \epsilon) \psi(\vec{r}, \Omega, \epsilon, t) = q(\vec{r}, \Omega, \epsilon, t). \quad (31)$$

Here $q(\vec{r}, \Omega, \epsilon, t)$ is the emission energy density so that

$$q(\vec{r}, \Omega, \epsilon, t) d\Omega de d\Omega dt \quad (32)$$

defines the number of particles emitted from collisions and sources with directions $d\Omega$ about $\Omega$ with energies $de$ about $\epsilon$ in a volume $dV$ about $\vec{r}$ during time interval $dt$. This equation represents the fundamental conservation of angular flux expressed in terms of losses from streaming and collisions and gains from emissions. The time-independent form of the transport equation, Eq. (31), is

$$\Omega \cdot \nabla \psi(\vec{r}, \Omega, \epsilon) + \sigma(\vec{r}, \epsilon) \psi(\vec{r}, \Omega, \epsilon) = q(\vec{r}, \Omega, \epsilon). \quad (33)$$

The time-independent equation useful for two important situations, in particular. The first involves applications in which it is important to consider the uncollided flux distribution (i.e., where particles have not yet collided). In this situation, the emission density is replaced by a known source of neutrons or photons (see the Ardra discussion in Section 4.4). The second use is in computational algorithms referred to as integral transport methods.

In the remaining radiation-transport discussion, we describe three different approximations to the general radiation transport equation. First, when the radiation mean free path is much less than other significant physical length scales (making an angular-dependent description unnecessary), the multigroup radiation diffusion model is adopted. To derive this model, one integrates the transport equation over all angles (i.e., transport directions) to obtain an average angular dependence. The spectral distribution properties are maintained. Second, when the distribution of energy in the radiation spectrum can be ignored, one can integrate the multigroup equation over the energy spectrum to obtain the total energy diffusion equation. Here, an effective radiation temperature $T_R$ is introduced. Constitutive relations are needed to couple $T_R$ with the material temperature $T$ within the resulting equations. Third, when radiation and matter are very strongly coupled, a single temperature is used in the approximation and one obtains the equilibrium radiation diffusion equation. We briefly outline each of these approximations in turn.

For many problems, a detailed angular treatment of energy transport is unnecessary and a description in terms of radiation energy density and radiant energy flux is sufficient. In particular, this is the case when the radiation mean free path is much smaller than the typical length scale over which significant changes in the radiation field are observed. To define the radiation energy density in the infinitesimal range $[\epsilon, \epsilon + de]$ at $\vec{r}$ at time $t$, we integrate the energy intensity over all solid angles to obtain the scalar flux

$$\phi(\vec{r}, \epsilon, t) = \int d\Omega \psi(\vec{r}, \Omega, \epsilon, t). \quad (34)$$
Then, \( \sigma(\vec{r}, \epsilon) \phi(\vec{r}, \epsilon, t) dV d\epsilon \) is the number of reactions per unit time in the volume-energy element \( dV d\epsilon \) (see equation 30). Observe that

\[
\int d\Omega \nabla \cdot \nabla \psi dA dt d\epsilon
\]

is the number of particles with energy in the interval \((\epsilon, \epsilon + d\epsilon)\) crossing the infinitesimal surface \(dA\) in the direction of positive \(\hat{n}\) during \(dt\). Continuing with our assumption that particle interaction is independent of angle of travel, we define the radiation energy flux (or current vector) corresponding to this infinitesimal energy range as

\[
J(\vec{r}, \epsilon, t) = \int d\Omega \psi(\vec{r}, \Omega, \epsilon, t) \Omega .
\]

Then the angle-independent form of the transport equation is

\[
\frac{\partial \phi(\vec{r}, \epsilon, t)}{\partial t} + c \nabla \cdot J(\vec{r}, \epsilon, t) + \sigma(\vec{r}, \epsilon) \phi(\vec{r}, \epsilon, t) = Q(\vec{r}, \epsilon, t) .
\]

where \(Q(\vec{r}, \epsilon, t)\) is the angle-integrated emission energy density. This equation is referred to as the multigroup radiation diffusion equation. Often, Fick's approximation is adopted to simplify this equation further: \(J(\vec{r}, \epsilon, t) \approx -D(\vec{r}, \epsilon) \cdot \nabla \phi(\vec{r}, \epsilon, t)\), where \(D\) is the coefficient of radiation diffusion.

To include matter-radiation coupling, we assume local thermodynamic equilibrium holds so that Kirchhoff's law can be used. That is, the material will emit photons at a rate

\[
\rho \kappa_a(\rho, T, \epsilon) B(T, \epsilon) ,
\]

where \(B\) is the Planck function, \(\kappa_a = \lambda_a^{-1}\) is the adsorption opacity (\(\lambda_a\) is the radiation mean free path), and \(T\) is material temperature. Adsorption reduces radiation energy intensity by

\[
-\kappa_a(\rho, T, \epsilon) \rho \phi .
\]

Then, the total rate of change in the scalar flux \(\phi\) due to energy emission and adsorption is given by

\[
\kappa_a(\rho, T, \epsilon) \rho [B(T, \epsilon) - \phi] .
\]

We remark that if the radiation field is in thermodynamic equilibrium with matter, then this expression is equal to zero. Since radiation transport in this case occurs by diffusion down material temperature gradients, primarily, the multigroup diffusion approximation (including matter-radiation coupling) is often written as

\[
\frac{1}{c} \frac{\partial \phi}{\partial t} = \nabla \cdot D(\epsilon) \nabla \phi + \kappa_a \rho [4\pi B(T) - \phi] .
\]

Also, note that we have employed Fick's approximation for the energy flux: \(J(\vec{r}, \epsilon, t) \approx -D(\vec{r}, \epsilon) \cdot \nabla \phi(\vec{r}, \epsilon, t)\). Typically, the diffusion coefficient is given as \(D = \frac{\lambda}{3}\), where \(\lambda = (\kappa_{tr})^{-1}\) is the transport mean free path.

If we integrate the scalar energy flux \(\phi\) appearing in the multigroup diffusion equation (in either form: equation (A.2) or equation (41)) over all energies in the spectrum, we arrive at the radiation energy density \(\epsilon_t\) used in the total energy diffusion approximation

\[
\epsilon_t = \int_0^\infty \phi(T_R, \epsilon) d\epsilon = a T_R^4 .
\]

Then, the multigroup diffusion equation is replaced by the total energy diffusion equation

\[
\frac{1}{c} \frac{\partial (a T_R^4)}{\partial t} = \nabla \cdot (D_R \nabla (a T_R^4)) - \rho \kappa_R [(a T^4) - (a T_R^4)] ,
\]

where the radiation diffusion coefficient is \(D_R = \frac{1}{3 \kappa_R \rho}\) (\(\kappa_R\) is the Rosseland mean opacity). This approximation is useful when the electromagnetic radiation reaches thermal equilibrium on time scales much less than the characteristic time scales of the overall system dynamics.
Thermodynamic equilibrium may exist in materials when $T \neq T_R$. In this case, the exchange of energy between the radiation field and matter occurs through electron-photon interactions, primarily. If the radiation-electron coupling rates via these interactions are long compared to the electron-ion collision times in the system, then energy exchange will occur between the radiation field and the thermal energy of the material. So we need to model the coupling between $T$ and $T_R$. Then, radiation transport can be described by coupling the total energy diffusion equation, Eq. (43), to an equation describing the evolution of $T$:

$$\frac{1}{c} \frac{\partial (aT_R^4)}{\partial t} = \nabla \cdot (D_R \nabla (aT_R^4)) - \kappa_P (T,T_R) \rho (aT_R^4) + \kappa_P' (T) \rho (aT^4)$$

(44)

$$\rho C_V \frac{\partial T}{\partial t} = \kappa_P (T,T_R) \rho (aT_R^4) - \kappa_P' (T) \rho (aT^4) .$$

(45)

Here, $C_V$ is the material specific heat, and $\kappa_P, \kappa_P'$ are two forms of the Planckian opacity (see [9]). Certain equilibrium situations allow $\kappa_P (T,T_R) = \kappa_P (T) = \kappa_P' (T)$. Notice that each equation gives the rate of change of energy density $T_R^4$ or material temperature $T$ in terms of diffusion of energy density and the negative of the energy coupling terms appearing in the other equation.

Finally, when electron-radiation coupling in matter is very strong, the emission and adsorption terms in these equations cancel and the material and radiation temperatures are in thermal equilibrium at a single temperature $T$ (i.e., $T = T_R$). Adding the two equations, then taking the limit $T_R \to T$, yields

$$\frac{1}{c} \frac{\partial (aT_R^4)}{\partial t} + \rho C_V \frac{\partial T}{\partial t} = \nabla \cdot (D_R \nabla (aT_R^4)) .$$

(46)

Often, it is reasonable to define a total specific heat of radiation and matter $C_{tot} = C_V + \frac{aT^3}{p}$ so that the equation is rewritten as

$$\rho C_{tot} \frac{\partial T}{\partial t} = \nabla \cdot (D_R \nabla (aT_R^4)) .$$

(47)

This equation is often referred to as the equilibrium radiation diffusion equation.

In summary, radiation transport is described by Eq. (31). Often, it is reasonable to introduce simplifications to the general equation and consider radiation diffusion in an averaged sense as described earlier. Typically, an average radiation diffusion model involves a system of equations in which the number of equations present is equal to the number of different temperature quantities to be considered. For example, if we wish to include the effect of diffusion of material temperature in the non-equilibrium diffusion model (compare to Eqs. (45)), we obtain

$$\frac{1}{c} \frac{\partial (aT_R^4)}{\partial t} = \nabla \cdot (D_R \nabla (aT_R^4)) - \kappa_P \rho (aT_R^4) + \kappa_P' \rho (aT^4)$$

(48)

$$\rho C_V \frac{\partial T}{\partial t} = \nabla \cdot (D_e \nabla T) + \kappa_P (T,T_R) \rho (aT_R^4) - \kappa_P (T) \rho (aT^4) ,$$

(49)

where $D_e$ is the thermal conduction coefficient. Finally, if we wish to consider radiation ($T_R$), electron ($T_e$), and ion ($T_i$) temperatures independently, then we employ the system

$$\frac{1}{c} \frac{\partial (aT_R^4)}{\partial t} = \nabla \cdot (D_R \nabla (aT_R^4)) - \kappa_P \rho (aT_R^4) + \kappa_P' \rho (aT_e^4)$$

(50)

$$\rho C_{V_e} \frac{\partial T_e}{\partial t} = \nabla \cdot (D_e \nabla T_e) + \rho \frac{(C_V)_e}{c \tau_{ie}} (T_i - T_e) + \kappa_P \rho (aT_R^4) - \kappa_P (T) \rho (aT_e^4) ,$$

(51)

$$\rho C_{V_i} \frac{\partial T_i}{\partial t} = \nabla \cdot (D_i \nabla T_i) - \rho \frac{(C_V)_i}{c \tau_{ie}} (T_i - T_e) ,$$

(52)

where $D_i$ is the ion diffusion coefficient and $\tau_{ie}$ is the equilibrium time length for ions and electrons. Here, we have employed another common assumption; namely, $T$ and $T_e$ are not significantly distinguishable so that there is no need to consider $T$ explicitly.
References


35


