THE INTERNATIONAL JOURNAL of HIGH PERFORMANCE COMPUTING APPLICATIONS

Volume 27 Number 1 February 2013

Special Issue on Multiphysics simulations: Challenges and opportunities

Contents

THE INTERNATIONAL JOURNAL of HIGH PERFORMANCE COMPUTING APPLICATIONS

The International Journal of High Performance Computing Applications
27(1) 4–83

DOI: 10.1177/1094342012468181
hpc.sagepub.com
SAGE

Multiphysics simulations: Challenges and opportunities

David E Keyes[1,2], Lois C McInnes[3], Carol Woodward[4], William Gropp[5], Eric Myra[6], Michael Pernice[7], John Bell[8], Jed Brown[3], Alain Clo[1], Jeffrey Connors[4], Emil Constantinescu[3], Don Estep[9], Kate Evans[10], Charbel Farhat[11], Ammar Hakim[12], Glenn Hammond[13], Glen Hansen[14], Judith Hill[10], Tobin Isaac[15], Xiangmin Jiao[16], Kirk Jordan[17], Dinesh Kaushik[3], Efthimios Kaxiras[18], Alice Koniges[8], Kihwan Lee[19], Aaron Lott[4], Qiming Lu[20], John Magerlein[17], Reed Maxwell[21], Michael McCourt[22], Miriam Mehl[23], Roger Pawlowski[14], Amanda P Randles[18], Daniel Reynolds[24], Beatrice Rivière[25], Ulrich Rüde[26], Tim Scheibe[13], John Shadid[14], Brendan Sheehan[9], Mark Shephard[27], Andrew Siegel[3], Barry Smith[3], Xianzhu Tang[28], Cian Wilson[2] and Barbara Wohlmuth[23]

Abstract

We consider multiphysics applications from algorithmic and architectural perspectives, where "algorithmic" includes both mathematical analysis and computational complexity, and "architectural" includes both software and hardware environments. Many diverse multiphysics applications can be reduced, en route to their computational simulation, to a common algebraic coupling paradigm. Mathematical analysis of multiphysics coupling in this form is not always practical for realistic applications, but model problems representative of applications discussed herein can provide insight. A variety of software frameworks for multiphysics applications have been constructed and refined within disciplinary communities and executed on leading-edge computer systems. We examine several of these, expose some commonalities among them, and attempt to extrapolate best practices to future systems. From our study, we summarize challenges and forecast opportunities.

Keywords

Multiphysics, multimodel, multirate, multiscale, implicit and explicit algorithms, strong and weak coupling, loose and tight coupling.

[1] KAUST, Saudi Arabia
[2] Columbia University, USA
[3] Argonne National Laboratory, USA
[4] Lawrence Livermore National Laboratory, USA
[5] University of Illinois at Urbana-Champaign, USA
[6] University of Michigan, USA
[7] Idaho National Laboratory, USA
[8] Lawrence Berkeley National Laboratory, USA
[9] Colorado State University, USA
[10] Oak Ridge National Laboratory, USA
[11] Stanford University, USA
[12] Princeton Plasma Physics Laboratory, USA
[13] Pacific Northwest National Laboratory, USA
[14] Sandia National Laboratories, USA
[15] University of Texas at Austin, USA
[16] Stonybrook University, USA
[17] IBM Research Center, USA
[18] Harvard University, USA
[19] SLAC National Accelerator Laboratory, USA
[20] Fermi National Accelerator Laboratory, USA
[21] Colorado School of Mines, USA
[22] Cornell University, USA
[23] Technische Universität München, Germany
[24] Southern Methodist University, USA
[25] Rice University, USA
[26] University Erlangen-Nuremberg, Germany
[27] Rensselaer Polytechnic Institute, USA
[28] Los Alamos National Laboratory, USA

Corresponding author:
Lois Curfman McInnes, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA.
Email: curfman@mcs.anl.gov

1 Introduction

Simulations that couple multiple physical phenomena are as old as simulations themselves. However, multiphysics simulation deserves fresh assessment, in light of steadily increasing computational capability and greater aspirations for simulation in domains of scientific prediction, engineering design, and policy making. An oft-quoted motivation for extreme computing is to relax assumptions of decoupling; however, it is by no means obvious that the promises claimed for coupled multiphysics simulation will be realized in extreme-scale computational environments in the principal way by which individual codes are coupled today, namely, through divide-and-conquer operator splitting. Coupling individual simulations may introduce limitations on stability, accuracy, or robustness that are more severe than the limitations imposed by the individual components. Furthermore, the data motion and data structure conversions required to iterate between independent simulations for each component may be more costly in latency and electrical power than those of the individually tuned components. Thus, ''one plus one'' may cost significantly more than ''two'' and may be less amenable to scalable execution than expected.

The report by Brown et al. (2008) on applied mathematics at the U.S. Department of Energy (DOE) emphasizes these challenges:

> Today's problems, unlike traditional science and engineering, do not involve physical processes covered by a single traditional discipline of physics or the associated mathematics. Complex systems encountered in virtually all applications of interest to DOE involve many distinct physical processes ...
>
> ... The issue of coupling models of different events at different scales and governed by different physical laws is largely wide open and represents an enormously challenging area for future research.

In this review, we consider multiphysics applications from algorithmic and architectural perspectives, where ''algorithmic'' includes both mathematical analysis and computational complexity, and ''architectural'' includes both software and hardware environments. Many multiphysics applications can be reduced, in principle, to an algebraic paradigm whose linearization brings to bear powerful tools of analysis in which individual components (denoted here as ''uniphysics'') are represented by diagonal blocks and the multiphysics coupling between them, as off-diagonal blocks. Such analyses are not always practical for realistic applications, but model problems representative of applications discussed herein provide insight. Various software frameworks for multiphysics applications have been constructed and refined within disciplinary communities and executed on leading-edge computer systems. We discuss here some commonalities among these frameworks and extrapolate best practices to future systems.

We begin by presenting a spectrum ranging from applications that all would agree are ''multiphysics'' to others that may involve semantic stretching but that have similar structure when the discrete problem is executed on a digital computer. Our objectives are broadly to characterize the state of the art in multiphysics simulation, to illustrate opportunities for leveraging successful approaches and software across physically disparate applications, and to identify gaps in understanding that are retarding the overarching goal of uncertainty-quantified high-fidelity simulations.

Whereas multiphysics is often approached bottom-up as the assembly of individual components, we consider the complementary perspective: that problems are intrinsically coupled and that the uniphysics applications often run are idealizations made in asymptotic limits. Those limits are not always carefully examined, because their examination may be non-trivial in realistic applications; hence, the importance of gaining insight from idealized situations is heightened. Because we strongly advocate examining coupling strength before pursuing a decoupled or split strategy for solving a multiphysics problem, we propose ''coupled until proven decoupled'' as a perspective worthy of 21st-century simulation purposes and resources.

1.1 What constitutes multiphysics?

Semantically, a multiphysics system consists of more than one component governed by its own principle(s) for evolution or equilibrium, typically conservation or constitutive laws. A major classification in such systems is whether the coupling occurs in the bulk (e.g., through source terms or constitutive relations that are active in the overlapping domains of the individual components) or whether it occurs over an idealized interface that is lower dimensional or a narrow buffer zone (e.g., through boundary conditions that transmit fluxes, pressures, or displacements). Typical examples of bulk-coupled multiphysics systems with their own extensively developed literature include radiation with hydrodynamics in astrophysics (radiation hydrodynamics, or RHD), electricity and magnetism with hydrodynamics in plasma physics (magnetohydrodynamics, or MHD), and chemical reaction with transport in combustion or subsurface flows (reactive transport). Typical examples of interface-coupled multiphysics systems are ocean–atmosphere dynamics in geophysics, fluid–structure dynamics in aeroelasticity, and core–edge coupling in tokamaks. Beyond these classic multiphysics systems are many others that share important structural features.

Success in simulating forward models leads to ambitions for inverse problems, sensitivity analysis, uncertainty quantification (UQ), model-constrained optimization, and reduced-order modeling, which tend to require many forward simulations. In these advances, the physical model is augmented by variables other than the primitive quantities in which the governing equations are defined. These variables may be probability density functions, sensitivity

gradients, Lagrange multipliers, or coefficients of system-adaptive bases. Equations that govern the evolution of these auxiliary-dependent variables are often derived and solved together with some of the physical variables. When the visualization is done in situ with the simulation, additional derived quantities may be carried along. Error estimation fields in adaptive meshing applications may constitute yet more. Though the auxiliary variables may not be "physical" in the standard sense, they give the overall simulation the structure of multiphysics.

In another important class of systems that might fall under the rubric of multiphysics by virtue of being multiscale, the same component is described by more than one formulation, typically with a computationally defined boundary or transition zone between the domains of applicability. We refer, for example, to field-particle descriptions of N-body systems in celestial mechanics or molecular dynamics, in which the gravitational or electrical forces between particles that are not immediate neighbors are mediated by a field that arises from the particles themselves. Typically, each particle defines a partition of the domain into "near" and "far" for this purpose, and the decomposition is strict. Another example is provided by atomistic–continuum models of solids, such as are used in crack propagation. In this case, the atomistic and continuum models both hold in a zone of finite thickness.

Recent schemes based on projective integration keep both fine and coarse models simultaneously in the picture and pass between the different "physics" for reasons of computational complexity. Specifically, they compute as much as possible with the coarse model, which may have constitutive terms that are difficult to derive from first principles but can be computed locally in time and space by taking suitable windowed averages or moments of the dependent variables of the fine model, which are closer to first principles. In these models, the coarse-to-fine transformation is called "lifting," and the fine-to-coarse transformation is called "restriction." Lifting may mean populating an ensemble of particles according to a distribution, and restriction may mean averaging.

Still other systems may have a multiphysics character by virtue of being multirate or multiresolution. A chemical kinetics model may treat some components as being in equilibrium, idealizing a fast relaxation down to a constraint manifold on which other components vary more slowly. Some phenomena may be partitioned mathematically by projections onto wavelet bases of different frequency or wavenumber properties that are naturally treated differently.

Stretching the semantics of multiphysics still further, we may distinguish only between different mathematical formulations or even just different discretizations of what is essentially the same physical model. An example is grafting a continuum-based boundary-element model for the far field onto a finite-element model for the near field.

Systems of partial differential equations (PDEs) of different types (e.g., elliptic-parabolic, elliptic-hyperbolic, or parabolic-hyperbolic) for the same component may be thought of as multiphysics because each of the classical PDE archetypes represents a different physical phenomenon. Even a single equation with terms of different types represents a multiphysics model because each term must often be handled through separate discretization or solver methods.

All our examples so far ultimately define partitions or sequences of partial updates on dependent variables. With a slightly stronger semantic stretch, a system in which independent variable spaces are handled differently or independently may also have an algebraic character similar to a true multiphysics system. Examples include the "alternating direction implicit" method, in which physical space is decoupled, or methods in which physical and phase space or physical and frequency space variables are handled separately.

We enumerate these quotidian examples of multiscale, multirate, multilevel, and multimodel problems because they possess similarities to multiphysics problems, particularly in the time domain, that allow leveraging and insight. They also introduce data structure issues that can be treated in common with data structures for multiphysics simulations. Should any inhomogeneity in a problem be regarded as multiphysics? Not by traditional classifications, perhaps. However, all simulations are ultimately transformed to algebraic problems, which are then transformed to arithmetic operations over discrete data structures. In algebraic form and in low-level operations, inhomogeneity-induced blockings from disparate root causes look structurally similar. We seek both dividends of understanding and dividends of software reuse from these similarities. Here "multiphysics" is synecdoche for a broad class of coarsely partitioned problems.

Many algorithmic approaches to multiphysics simulations are "one-off" and do not leverage algebraic insights from related problems. Many software approaches are unpublished in code form, or are even proprietary. Approximations that go into the decoupling are inaccessible to measurement. These situations have retarded the growth of successful and reliable simulations of complex phenomena. Our efforts begin by establishing a common parlance for describing multiphysics systems that is simple and abstract enough to allow comparisons and reuse of software solutions.

1.2 Prototype algebraic forms

The two simplest systems that exhibit the crux of a multiphysics problem are the coupled equilibrium problem

$$F_1(u_1, u_2) = 0 \tag{1a}$$

$$F_2(u_1, u_2) = 0 \tag{1b}$$

and the coupled evolution problem

$$\partial_t u_1 = f_1(u_1, u_2) \tag{2a}$$

$$\partial_t u_2 = f_2(u_1, u_2) \tag{2b}$$

where the same operator notation may be overloaded in continuous or discrete settings, which are distinguished by context. When (2a)–(2b) is semi-discretized in time, the evolution problem leads to a set of problems that take the form (1a)–(1b) and are solved sequentially to obtain values of the solution $u(t_n)$ at a set of discrete times. Here u refers generically to a multiphysics solution, which has multiple components (such as those introduced in Section 1.1) indicated by subscripts $u = (u_1, \ldots, u_{N_c})$; the simplest case of $N_c = 2$ components is indicated here. Noting this, we will generically use the notation in (1a)–(1b) to refer to either a coupled equilibrium problem or a single timestep of a coupled evolution problem.

We assume initially, for convenience, that the Jacobian $J = \frac{\partial(F_1,F_2)}{\partial(u_1,u_2)}$ is diagonally dominant in some sense and that $\frac{\partial F_1}{\partial u_1}$ and $\frac{\partial F_2}{\partial u_2}$ are non-singular. These assumptions are natural in the case where the system arises from the coupling of two individually well-posed systems with legacies of being solved separately. In the equilibrium problem, we refer to F_1 and F_2 as the component *residuals*; in the evolution problem, we refer to f_1 and f_2 as the component *tendencies*.

The choice of solution approach for these coupled systems relies on a number of considerations. From a practical standpoint, existing codes for component solutions often motivate operator splitting as an expeditious route to a first multiphysics simulation capability making use of the separate components. This approach, however, may ignore strong couplings between components and give a false sense of completion. Solution approaches ensuring a tight coupling between components require smoothness, or continuity, of the nonlinear, problem-defining functions, F_i, and their derivatives. Any potential discontinuities must be identified and addressed before putting confidence in these approaches. For now, we illustrate multiphysics solution strategies within the context of the prototypical systems above and defer details until Section 3.

Classic multiphysics algorithms preserve the integrity of the two uniphysics problems, namely, solving the first equation for the first unknown, given the second unknown, and the second equation for the second unknown, given the first. This represents the reductionist approach of science, and as discussed in Section 4, software generally exists to do this. Multiphysics coupling is taken into account by iteration over the pair of problems, typically in a Gauss–Seidel manner (see Algorithm 1), linear or nonlinear, according to context. Here we employ superscripts to denote iterates.

When this iteration converges, the accuracy with which the discrete equations are solved can be improved by continuing the iterations. The largest implicit aggregate is the largest of the uniphysics problems; we refer to this iteration as "loosely coupled." A Jacobi-like iteration can be similarly defined. This further decoupling exposes more parallelism, albeit possibly at the cost of a slower convergence rate.

Algorithm 1. Gauss–Seidel multiphysics coupling.

Given initial iterate $\{u_1^0, u_2^0\}$
for $k = 1, 2, \ldots,$ (until convergence) **do**
  Solve for v in $F_1(v, u_2^{k-1}) = 0$; set $u_1^k = v$
  Solve for w in $F_2(u_1^k, w) = 0$; set $u_2^k = w$
end for

Algorithm 2. Multiphysics operator splitting.

Given initial values $\{u_1(t_0), u_2(t_0)\}$
for $n = 1, 2, \ldots, N$ **do**
  Evolve one timestep in $\partial_t u_1 + f_1(u_1, u_2(t_{n-1})) = 0$ to obtain $u_1(t_n)$
  Evolve one timestep in $\partial_t u_2 + f_2(u_1(t_n), u_2) = 0$ to obtain $u_2(t_n)$
end for

The simplest approach to the evolutionary problem likewise employs a field-by-field approach in a way that leaves a first-order-in-time splitting error in the solution. Algorithm 2 gives a high-level description of this process, which produces solution values at time nodes $t_0 < t_1 < \ldots < t_N$. Here, we use notation $u(t_0), \ldots, u(t_N)$ to denote discrete timesteps. An alternative that staggers solution values in time is also possible.

The individual component evolutions in Algorithm 2 can be implicit or explicit and performed with or without subcycles. However, there is no point in their being of high order unless a higher-order coupling scheme than this is used, such as Strang splitting (Strang, 1968) (see Section 3.2.2 for second-order or more intricate splittings) or temporal Richardson extrapolations (Richardson, 1911; Richardson and Gaunt, 1927; Gautschi, 1997) for higher-order. An inner loop may be placed inside each timestep in which the coupling variables are updated to satisfy implicit consistency downstream while still preserving the legacy software for each component. In the literature, this loosely coupled evolution is depicted by the diagram in Figure 1, where in the context of aeroelasticity, for instance, the first component consists of fluid velocity and pressure and the second component, of structural displacements. As described in more detail in Section 2, this is an interface transmission form of coupling in which structural displacements provide boundary conditions for the fluid, and fluid pressures provide boundary conditions for the structure in the context of a dynamic mesh.

If the residuals or tendencies and their derivatives are sufficiently smooth and if one is willing to write a small amount of solver code that goes beyond the legacy component codes, a good algorithm for both the equilibrium problem and the implicitly time-discretized evolution problem is Jacobian-free Newton–Krylov (JFNK) (Knoll and Keyes, 2004). Here, the problem is formulated in terms of a single residual that includes all components in the problems,

$$F(u) \equiv \begin{pmatrix} F_1(u_1, u_2) \\ F_2(u_1, u_2) \end{pmatrix} = 0 \tag{3}$$

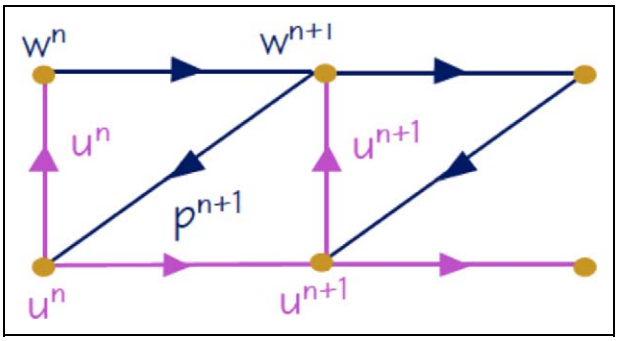


Figure 1. Illustration of loose coupling, with time advancing to the right.

where $u = (u_1, u_2)$. The basic form of Newton's method to solve equation 3, for either equilibrium or transient problems, is given by Algorithm 3 (see e.g., Kelley, 1995). Because of the inclusion of the off-diagonal blocks in the Jacobian, for example

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} \\ \frac{\partial F_2}{\partial u_1} & \frac{\partial F_2}{\partial u_2} \end{bmatrix}$$

Newton's method is regarded as being "tightly coupled." Section 3 provides more details, including the JFNK variant, as well as the additive Schwarz preconditioned inexact Newton (ASPIN) method (Cai and Keyes, 2002), which offers advantages when the types of nonlinearities in the two components are quite different.

The operator and algebraic framework described here is relevant to many divide-and-conquer strategies in that it does not "care" (except in the critical matter of devising preconditioners and nonlinear component solvers for good convergence) whether the coupled subproblems are from different equations defined over a common domain, the same equations over different subdomains, or different equations over different subdomains. The general approach involves iterative corrections within subspaces of the global problem. All the methods have in common an amenability to exploiting a "black-box" solver philosophy that amortizes existing software for individual physics components. The differences are primarily in the nesting and ordering of loops and the introduction of certain low-cost auxiliary operations that transcend the subspaces.

Not all multiphysics problems can easily or reliably be cast into these equilibrium or evolution frameworks, which are primarily useful for deterministic problems with smooth operators for linearization. In formulating multiphysics problems, modelers first apply asymptotics to triangularize or even to diagonalize the underlying Jacobian as much as possible, pruning provably insignificant dependencies, but bearing in mind the conservative rule of coupling: "guilty until proven innocent." One then applies multiscale analyses to simplify further, eliminating stiffness from irrelevant mechanisms. Because of significant timescale and resolution requirement differences, these activities often reveal justifications for splitting some

Algorithm 3. Newton's method.

Given initial iterate u^0
for $k = 1, 2, \ldots,$ (until convergence) **do**
  Solve $J(u^{k-1})\,\delta u = -F(u^{k-1})$
  Update $u^k = u^{k-1} + \delta u$
end for

physics or models from others. In these cases, an operator-splitting approach should be applied. However, caution is warranted to ensure that temporal or spatial scales do not overlap between split physics as the simulation progresses. If this happens, a more coupled approach is generally required.

1.3 Structure and motivation for this review

The previous section discussed a two-component multiphysics system at the operator and discrete level, primarily in terms of operations that are already available in simulations for the individual components, introducing generic notation that will be useful throughout the document. Generalization of the algorithms to N_c components is trivial. However, the convergence of the methods is not guaranteed.

Applications are described in terms of this notation in Section 2; model problems are treated by these methods in Section 3; and software that implements data structures and coupling to accommodate these methods is discussed in Section 4. In principle, most of the multiphysics applications can be made to exhibit the partition above, with variations on the functional forms of the cross-component dependencies.

Even when all the components entering into a multiphysics simulation are well verified, the peril of operator splitting is that systems coupled by new off-diagonal terms may admit destabilizing modes not present in any component system alone. Even when the coupling is stable in a continuous sense, the accuracy of the coupled system and the stability of the numerical discretization can be compromised by operator splitting, as shown in Section 3.4.

Couplings that are relaxed at the component level may include fluxes at interfaces between physical domains, bulk source terms, complex constitutive laws, or cross-dimensional effects. With the advent of nonlinear implicit solvers and their availability in well-established packages such as PETSc (Balay et al., 1997, 2012), SUNDIALS (Hindmarsh et al., 2005; Woodward et al., 2012), and Trilinos (Heroux et al., 2005, 2012) (see Section 4.1.3), there exists a robust alternative to operator splitting; however, preconditioning fully temporally implicit discretizations remains challenging, particularly with advanced, high-order spatial discretizations. For efficiency, it is important to understand the interaction of the many forms of errors and mechanisms of instability, as well as the relative costs of reducing them, particularly in the context of emerging

architectures, to which traditional flop-based complexity is less relevant than the complexity of transmitting or copying data.

Despite these mathematical perils, computational models of multiphysics phenomena are often approached through operator splitting, which has attractions from a software-engineering viewpoint. However, in an era of architectures that penalize data motion but allow nearly "free" flops, splitting usually results in an increase (proportional to the degree of the splitting) in the ratio of loads and stores relative to flops. That is, operator splitting shrinks the arithmetic intensity of the simulation, which is the wrong direction of code evolution. The move toward more tightly coupled approaches looms as a prime opportunity for adaptation to the exascale, with its memory bandwidth stringencies (Brown et al. 2010; Ang et al., 2012). In Sections 5 and 6 we address opportunities in multiphysics applications, propose transferable best practices, and emphasize applications at extreme scale. The topics to be addressed include physical modeling (fidelity, ranges of applicability), mathematical analysis (formulations, well-posedness in the continuum sense), numerical analysis (accuracy, stability), algorithms (complexity, optimality), software engineering (programmer productivity, reusability), and implementation on emerging architectures (performance, portability, scalability). Section 7 summarizes our conclusions. A glossary of basic multiphysics terms as used in this report is given at the end of this article.

We conclude this section by quoting from recent reports of DOE communities:

> The dominant computational solution strategy over the past 30 years has been the use of first-order-accurate operator-splitting, semi-implicit and explicit time integration methods, and decoupled nonlinear solution strategies. Such methods have not provided the stability properties needed to perform accurate simulations over the dynamical timescales of interest. Moreover, in most cases, numerical errors and means for controlling such errors are understood heuristically at best. (Simon et al., 2007)
>
> The following priority research direction [was] identified: develop scalable algorithms for non-hydrostatic atmospheric dynamics with quasi-uniform grids, implicit formulations, and adaptive and multiscale and multiphysics coupling ... Improvements in scalability alone will not be sufficient to obtain the needed throughput (the time it takes to complete a climate simulation). Obtaining the needed level of throughput will also require incorporating as much implicitness as possible ... (Washington et al., 2008)
>
> Often, scientists would ideally employ a high-order timestepping scheme and take relatively large timesteps for computational economy. However, if operator-splitting techniques are used, the low-order splitting error thwarts this objective. Moreover, computational challenges on the immediate horizon – optimization for design or control, inverse problems for parameter identification, multiphysics coupling, etc. – are most naturally tackled with fully implicit formulations. (Tang et al., 2009)
>
> The great frontier of computational physics and engineering is in the challenge posed by high-fidelity simulations of real-world systems, that is, in truly transforming computational science into a fully predictive science. Real-world systems are typically characterized by multiple, interacting physical processes ("multiphysics"), interactions that occur on a wide range of both temporal and spatial scales. (Rosner et al., 2010)

2 Practices and perils in multiphysics applications

In this section, we discuss examples of multiphysics applications selected from diverse disciplines within science and engineering. By outlining these real-world examples, we illustrate, from a practical standpoint, how multiphysics applications are assembled and used. These examples also illustrate the rich combinatorics of possible couplings among uniphysics components. In taking this approach, we provide a departure point from which to discuss issues that can be encountered by naively coupling two or more robust uniphysics codes to form a multiphysics application.

One of the most important of these issues is the coupling itself: a term that we now define and discuss further in Section 2.2.1. The concept of physical coupling within a multiphysics application is intuitive. However, upon continued discussion, the term can become increasingly vague since coupling exists at least at both a physical level and an algorithmic level. Hence, some definitions are in order. Throughout this report we refer to *strong (versus weak) coupling of physical models* as meaning that physical components of the modeled system are coupled by a strong (versus weak) interaction. Correspondingly, we refer to *tight (versus loose) coupling of numerical models* as meaning that algorithmic components are coupled by a tight (versus loose) interaction. See the glossary for further discussion.

We note that there is not a direct one-to-one correspondence between strong (weak) coupling of physical models and tight (loose) coupling of numerical models, since a tight or a loose coupling scheme may be used for strongly or weakly coupled physical models.

We also note that terminology has often been used interchangeably in the literature; for example, in fluid–structure interactions (FSIs), tight coupling schemes and strong coupling schemes have been discussed interchangeably. To illustrate the distinction, the effect of the fluid pressure on the displacement of a deformable structure and of the structural displacement on the fluid velocity are instantaneous; thus, the physical interaction is strong. However, it is possible to update the structural displacement, based on a new fluid pressure distribution, without simultaneously adjusting the fluid velocity to respect its new bounding surface. If such is the case, the numerical coupling is loose, and iteration between the two phases may be required to reduce the error of the tight interaction sufficiently. Conversely, in the tightest numerical coupling, the state variables of the two phases would be updated simultaneously, typically in an

algorithm of greater complexity. Either the tight or the loose numerical coupling could satisfy a given accuracy, and either may be more computationally efficient depending upon the problem and how it is applied.

In a reacting flow, the concentration of a chemical species may have insignificant influence on the thermodynamic state of the fluid within a broad range, and may also be insensitive to the thermodynamic state; thus, the physical interaction is weak. However, the numerical model may evolve these state variables independently (weak coupling) or simultaneously (strong coupling), although the latter effort produces little benefit in this case.

We further comment that physical interaction may be strong in one direction and weak in the other, or strong in both directions. We describe such a system as exhibiting one-way or two-way coupling, respectively. One-way coupling leads naturally in limit to a numerical system that is triangular.

We consider such coupling issues for a variety of PDE-based multiphysics applications in Section 2.1. In Section 2.2 we identify commonalities and differences across the application spectrum, and we touch on issues in algorithms and software from an applications perspective as a foundation for more detailed discussion in Sections 3 and 4.

2.1 Examples of PDE-based multiphysics applications

A broad range of multiphysics simulations are under way in the computational science community, as researchers increasingly confront questions about complex physical and engineered systems characterized by multiple interacting physical processes that have traditionally been considered separately. Sections 2.1.1–2.1.12 introduce some large-scale PDE-based multiphysics applications, including FSI, fission reactor fuel performance, reactor core modeling, crack propagation, DNA sequencing, fusion, subsurface science, hydrology, climate, RHD, geodynamics, and accelerator design. Discussion focuses on current practices and challenges in application-specific contexts.

2.1.1 Interaction of fluids and structures

Numerical simulations that model interaction between incompressible laminar flows and elastic structures require coupling a description of the fluid (typically the incompressible Navier–Stokes equations or a weakly compressible lattice-Boltzmann equation) with a description of the structures. Sample application areas for this scenario include blood flow in arteriae, veins, and heart chambers; low Mach number aerodynamics; marine propellers; and hydroelectric power plants. The unknowns of the involved equations (velocities and pressure for the fluid, displacements for the structure) are associated with different locations in the overall computational domain, resulting in a surface-coupled problem.

FSI can be simulated in at least two ways. One approach is to solve a large system of equations for all fluid and structure unknowns as a single system—typically ill-conditioned. Alternatively, in a partitioned approach, one has separate solvers for the fluid and the structure, together with a suitable coupling method. In the latter case, boundary conditions for both single-physics problems at the coupling surface have to be defined. The respective interface values are passed from one solver to the other. This approach requires mapping methods for physical variables between (in general) non-matching solver grids at coupling surfaces. Important features desired of such mappings are accuracy, consistency, and conservation of energy and momentum. Two main classes of mapping methods can be identified: interpolation methods (Farhat et al., 1998b; Jaiman et al., 2006; Scholz et al., 2006; De Boer et al., 2007; Bungartz et al., 2010), based on geometric relations between the involved grid points, and mortar methods, in which boundary conditions are formulated in weak form by using Lagrange multipliers (Farhat et al., 1998b; Baaijens, 2001; Ross, 2006; Klöppel et al., 2011).

The coupling itself can be done with different methods, leading to looser or more tightly coupled timestepping methods; see Figure 2 for two variants. The loosest coupling is a one-way coupling, where the flow solver computes a force exerted on the structure using a rigid-structure geometry, and structural movements are computed in a postprocessing-like manner based on these forces. This strategy is obviously applicable only for small and static structure deformations. The most widely used class of iteration schemes is Gauss–Seidel-like coupling iterations (see Algorithm 1), with variants ranging from a single iteration loop per timestep to repeated iteration-to-convergence within each timestep, with or without (Aitken) underrelaxation (Irons and Tuck, 1969; Wall et al., 2001; Schäfer et al., 2010), to interface quasi-Newton methods that efficiently compute approximate Newton iterations based on sensitivities resulting from Gauss–Seidel iterations (Degroote et al., 2009). To account for what are usually moderately different timescales in the fluid and the structure, a subcycling in the flow solver can be used. In the absence of turbulence, spatial scales are essentially the same throughout fluid and structure domains.

The choice of coupling implementation can lead to development of numerical instabilities in multiphysics codes. For incompressible fluids, the so-called added-mass effect induces instabilities in loosely coupled simulations and in Gauss–Seidel-like iterations: the force exerted by the fluid on a moving structure can be interpreted as a virtual added mass of the structure (see e.g., Van Brummelen, 2009). For an incompressible flow, each acceleration or deceleration of the structure causes an immediate change in this added mass (whereas the added-mass change for a compressible flow increases continuously over time). If this change is too large, both loosely and tightly coupled Gauss–Seidel-like coupling schemes become unconditionally unstable; in other words, a reduction of the timestep does not cure the instability (Van Brummelen, 2010). In the case of a massless structure, a reduction of the timestep

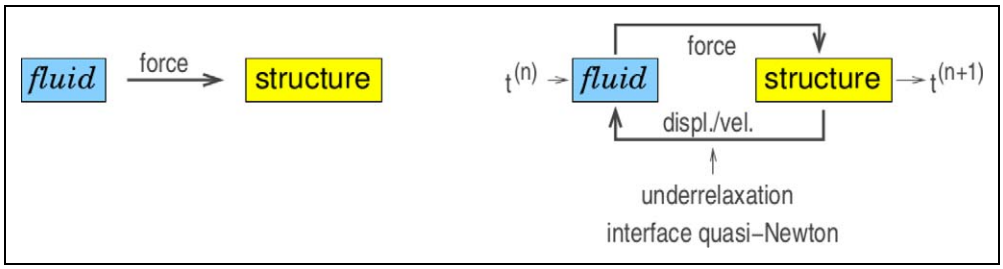


Figure 2. One-way and Gauss–Seidel-like coupling strategies for fluid–structure interaction (FSI) simulations.

even worsens instabilities (Degroote et al., 2009). Typically, only a few low-frequency Fourier modes of interface displacements or velocities are unstable (Degroote et al., 2009; Van Brummelen, 2010). Interface quasi-Newton methods rapidly capture the unstable Fourier modes and thus lead to stable coupling iterations (Degroote et al., 2009). However, standard multigrid techniques, with Gauss–Seidel-like iterations as a smoother, do not work for incompressible flows. Although they smooth high frequencies, they are unstable for low frequencies. In addition, the convergence rate for low-frequency modes does not improve on spatially coarser grids.

Since Gauss–Seidel-like coupling iterations, including the interface quasi-Newton method, are inherently sequential in execution order of flow and structure solver, and since the structure solver is usually much less costly than is the flow solver, alternative Jacobi-like methods (see Ross, 2006, for compressible flows) are desirable on massively parallel systems. These avoid processor idle time during the execution of the structural solver and the coupling numerics.

As a practical application of FSI, we consider examples in high-performance, high-fidelity analysis, which finds application in modeling the flow of airstreams over aircraft and vehicles as well as fluid flow over underwater structures. Airflow problems necessitate the solution of nonlinear compressible FSI problems. Specific examples include the parametric identification of an F-16 Block-40 fighter aircraft in clean wing configuration and subsonic, transonic, and supersonic airstreams (Farhat et al., 2003; Geuzaine et al., 2003); the aeroelastic analysis of an F/A-18 5.6 Hz limit-cycle oscillation configuration; the flutter clearance of the laminar flow wing of a supersonic business jet concept; and the aeroelastic tailoring of a Formula 1 car. Examples of incompressible FSI problems include the study of intense implosive collapses of gas-filled underwater structures and their effects on nearby structures (Farhat et al., 2010).

Many of these problems feature multiple spatial and temporal scales. In all of them, the fluid domain can be occupied by one or multiple interacting fluids, the structure can be linear or nonlinear with self-contact, and the FSI occurs at physical interfaces.

The AERO code (Geuzaine et al., 2003) is an example of a multiphysics FSI software package that considers these issues. The functional modules of the AERO code include some of the most important considerations for producing accurate models: (1) the structural analyzer AERO-S, (2) the compressible turbulent flow solver AERO-F, (3) the auxiliary module MATCHER, which enables the discretization of fluid–structure transmission conditions at nonmatching, discrete fluid–structure interfaces (Maman and Farhat, 1995), and (4) the auxiliary module, SOWER, which manages all parallel input/output (I/O) associated with this software. AERO-F can operate on unstructured body-fitted meshes as well as fixed meshes that embed discrete representations of surfaces of obstacles, around and/or within which the flow is to be computed. The body-fitted meshes and the embedded discrete surfaces can be fixed, move, and/or deform in a prescribed manner, or can be driven by interaction with the structural analyzer AERO-S. In the case of body-fitted meshes, the governing equations of fluid motion are formulated in the arbitrary Lagrangian–Eulerian (ALE) framework. In this case, large mesh motions are handled by a corotational approach, which separates the rigid and deformational components of the motion of the surface of the obstacle (Farhat et al., 2001), and robust mesh motion algorithms that are based on structural analogies (Farhat et al., 1998a). In the case of embedded surfaces that can have complex shapes and arbitrary thicknesses, the governing equations of fluid motion are formulated in the Eulerian framework, and the wall boundary or transmission conditions are treated by an embedded boundary method (Wang et al., 2011b). Both AERO-F and AERO-S feature explicit and implicit time integrators with adaptive time-stepping. Both modules are coupled by a family of partitioned analysis procedures that are loosely coupled but exhibit excellent numerical stability properties and are provably second-order time-accurate (Farhat et al., 2006). AERO-F and AERO-S communicate via runtime software channels, for example, using MPI. They exchange aerodynamic (pressure) and elastodynamic (displacement and velocity) data across non-matching, discrete fluid and structure mesh interfaces using a conservative method for the discretization of transmission conditions (Farhat et al., 1998b) and the data structures generated for this purpose by MATCHER.

2.1.2 Fission reactor fuel performance. Simulating the performance of light water nuclear reactor fuel involves complex thermomechanical processes between fuel pellets, which

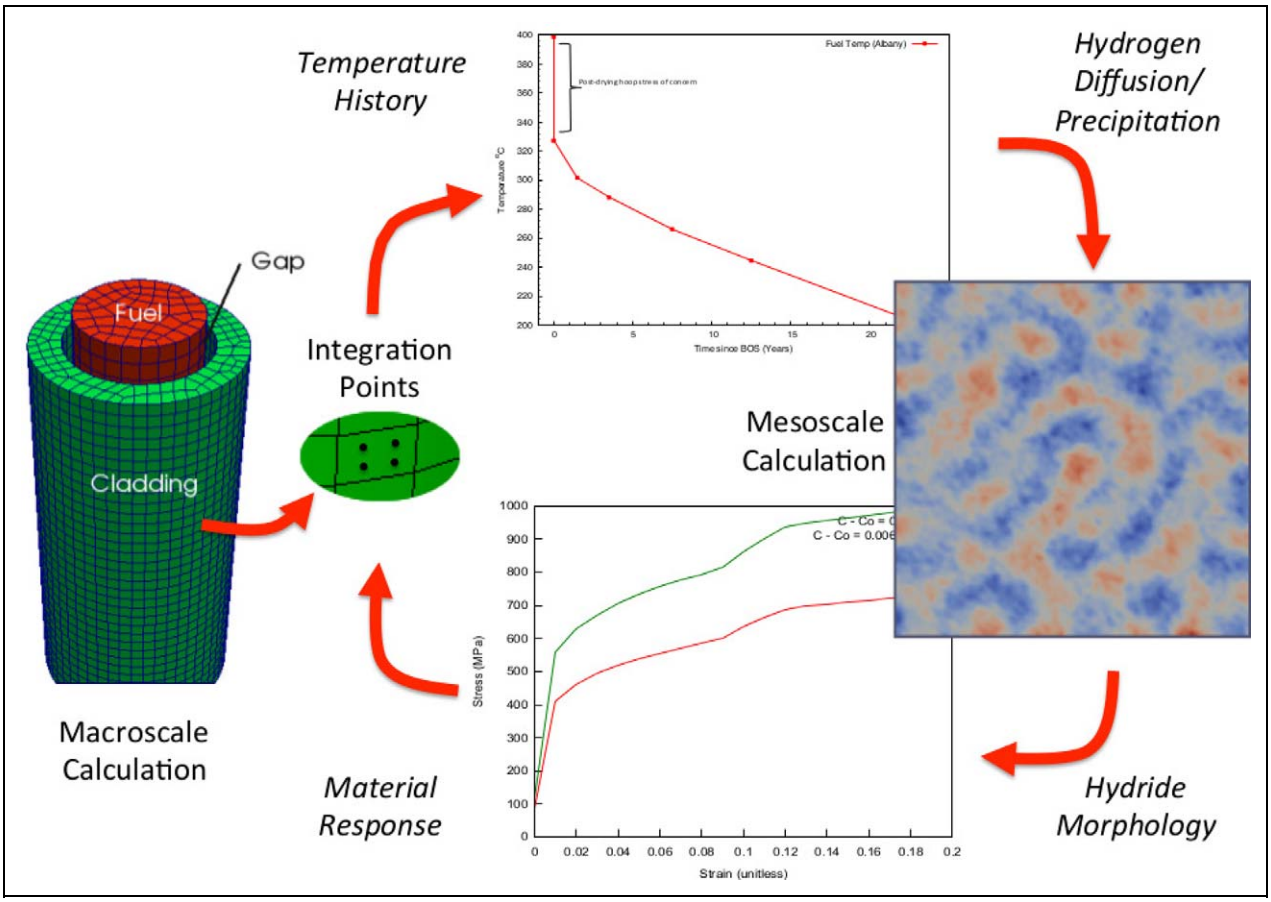


Figure 3. Illustration of a multiscale multiphysics calculation to predict the behavior of reactor fuel cladding containing hydrogen inclusions (hydrides). In this process, an initial guess of the stress state in the fuel cladding and the temperature history are calculated by using a macroscale calculation of the fuel rod. These states, defined at the integration points of the multiscale calculation, are passed to concurrent phase field mesoscale calculations in order to determine the morphology of the hydride inclusions. This morphology is then passed back to the mesoscale calculation and used to calculate the stress profile within the cladding.

are made of fissile material, and the protective cladding that surrounds the pellets (Hansen et al., 2009b). Figure 3 shows the relationship between the fuel pellet and cladding and the helium-filled gap separating them. Advancing fuel design and analysis capability requires progress both in the modeling of the fuel material behavior and in the development of a multidimensional computational framework that couples these models in a consistent, accurate, and efficient manner to describe how the fuel geometry and behavior change over the lifetime of the fuel.

The base physics begins with heat diffusion that partially drives the mechanical response of the fuel pellet and cladding system (Newman et al., 2009a). Thermomechanical response problems are often solved in an operator-splitting fashion, but this approach can result in convergence and stability issues as the pellet and cladding come into contact as the fuel ages. Newman et al. (2009b) solve the fuel performance problem in a fully coupled manner on a single mesh. Further, an oxygen diffusion equation is added since, under hyperstoichiometric conditions, both the thermal conductivity and mechanical properties of the fuel pellet are impacted by the diffusion of oxygen within the matrix. Additional material models are used to describe other details of the thermal and mechanical behavior of the fuel and cladding, such as swelling of the fuel due to the accumulation of fission products within it (Williamson and Knoll, 2009). The expansion of the pellet and, indirectly, the width of the gap separating the pellet and the cladding are functions of the thermal profile within the pellet, the age of the fuel (the degree of fission-product-driven swelling), and the creepdown of the cladding onto the outside surface of the pellet.

Given the significant radial and axial displacement of the pellet relative to the cladding, it is infeasible to mesh the helium-filled gap between the two, especially in a multiple pellet simulation. Since the gap is long and narrow, Rayleigh numbers are unlikely to be high enough within the gap to support significant convection. However, it is important to accurately model the conductive heat transfer across the gap, since the gap is one of the principal factors that influence performance of the fuel. Thus, a fully coupled thermomechanical contact model was developed to model the evolution of the gap within the fully coupled multiphysics code (Hansen, 2011).

Irradiation of the fuel and the cladding affects the thermal and mechanical properties of both. It is difficult to describe these effects on the coarse scale of the finite-element code described above. Modeling the irradiation of the material is more easily done at lower-length scales, such as the mesoscale. One might employ a phase-field simulation approach to calculate the evolution of the thermal conductivity of the fuel under irradiation. Use of this model can be challenging, however, since the characteristics of the irradiation damage change with fuel temperature. Tonks et al. (2009, 2010) couple a mesoscale solution of the Cahn–Hilliard equation to the macroscale approach to form a multiscale simulation. The mesoscale simulation requires the coupling of a temperature field from the macroscale as input to the simulation. Further, the thermal conductivity calculated at the mesoscale impacts the temperature field calculated on the macroscale. A similar approach, shown in Figure 3, may be used to assess the safety of handling used fuel after drying and storage. Here, hydrogen inclusions (hydrides), which may weaken the cladding surrounding the fuel pellets, can form during the drying of fuel. To predict cladding performance, a macroscale structural and thermal response analysis is linked to a mesoscale calculation that computes the structure of the hydride inclusions (the hydride morphology). The stress and temperature at the integration points serve as boundary conditions for the mesoscale phase field calculation hosted there; the temperature and temperature gradient in time drive the precipitation of hydrogen from solid solution; and the stress state during precipitation affects the orientation of the hydride inclusions. The resultant hydride morphology calculated at the mesoscale is then upscaled for input to the mechanical response macroscale calculation that predicts the cladding behavior under loading scenarios of potential concern.

The equations that describe reactor fuel performance are solved by using a fully coupled, JFNK approach (Gaston et al., 2009a,c; Newman et al., 2009b) (see Algorithm 3 and Section 3.1.2). A fully implicit time-integration strategy is used to support accurate time integration needed over short transient events, such as the initial warmup of the reactor and power transients during operation, and to allow long timesteps to be taken under the long quasi-static periods of normal operation. All the phenomena within the reactor fuel are tightly interrelated, particularly the mechanical aspects of the fuel. As the fuel warms up, the pellet expands, thereby narrowing the gap and increasing the pressure of the gases between the pellet and cladding. The narrowing gap increases the heat transfer efficiency across it, which cools the pellet, checking its expansion. Attempting to decouple these processes in order to simplify the solution process can result in convergence issues and even ''numerical chatter'', where the pellet and cladding chatter on incipient contact instead of arriving at an equilibrium solution (Hansen, 2011). In the gap-integration technique presented here, fluxes and tractions are exchanged between the pellet and cladding using the mortar finite-element method. The integration space and Lagrange multiplier implementation employ the Moertel package (Gee and Hansen, 2012) in Trilinos.

2.1.3 Conjugate heat transfer and neutron transport coupling in reactor cores. It is well understood that nuclear fission energy is a key component of our current and future energy needs. It is therefore urgent to develop nuclear reactors that are safe and environment-friendly. Accurately predicting neutron densities in a reactor core is a critical aspect of design and safe operation. Reactor cores are designed to balance power generation with efficient heat transfer and fuel burnup objectives, optimizing energy density relative to design limits for key global properties such as peak temperature and temperature gradient, neutron fluence, and pumping power (pressure drop).

With the availability of leadership-class supercomputers, high-fidelity simulations of different physical phenomena (such as neutron transport, coolant flow, structural mechanics, and materials science) inside complex reactor geometries appear a feasible and attractive option. However, the modeling and simulation (M&S) tools need to accurately represent the coupling among these physical processes while preserving the accuracy in each physics. The existing M&S tools are based on various approximations (to reduce the computational burden) and rely on various correlations derived from experiments that can be applied for a narrow range of design space. Therefore, what is needed is to develop a modern, integrated design tool that allows automated data transfer among different design activities, and facilitates enhanced global design optimizations. Work in the Simulation-based High-efficiency Advanced Reactor Prototyping (SHARP) project (Siegel et al., 2007; Pointer et al., 2012a) and the Center for Exascale Simulation of Advanced Reactors (CESAR) (Rosner et al., 2012) focuses on developing such tools for reactor design.

On relatively fast timescales (compared with the physics of neutronics/fuel coupling) and within certain parameter regimes, methods for computing the accurate coupling between the neutron transport and conjugate heat transfer, for fixed fuel and structural configuration, become the key problem of interest for reactor analysts. Historically, reactor design and analysis tools have been based on weak coupling between bulk fluid characterizations and few-group diffusion or nodal neutronics methods.

The SHARP and CESAR projects are investigating high-fidelity fluid/neutronic coupling methods for a range of reactor phenomena. Many of the calculations require large numbers of energy groups and are crucial for simulating various accident scenarios, such as ejected control rod and loss of coolant flow without shutdown. These accidents, which involve rapid thermal–mechanical changes in the reactor core geometry, require several minutes to hours of time data. The existing tools do model these types of problems, but confidence in the solutions is difficult to assess. Higher-fidelity solution capabilities will not only allow better understanding of the behavior of existing reactor systems but will

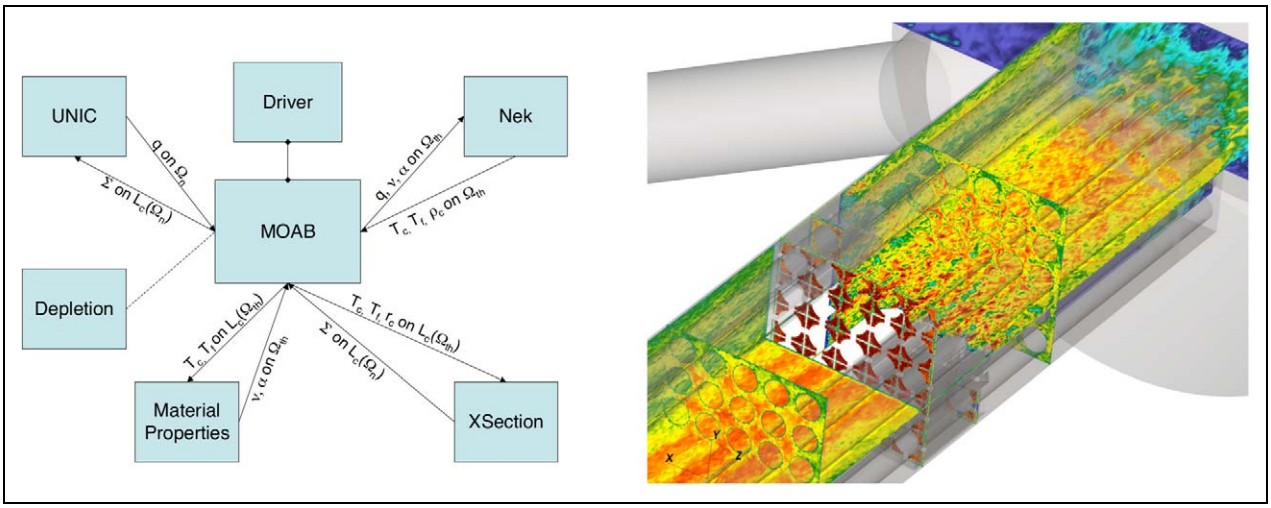


Figure 4. Illustration of multiphysics coupling in SHARP. **Left:** Schematic representation of interrelationships among the different modules of SHARP. Here Ω_n denotes the neutronics (UNIC) mesh, and Ω_{th} denotes the thermal-hydraulics (Nek) mesh (Siegel et al., 2007). **Right:** Predicted coolant velocity distribution in a light-water reactor (Pointer et al., 2012b). Evaluating reactor design performance accurately depends on resolving multiphysics coupling between neutron transport and coolant flow through the complex reactor geometries.

also predict the behavior of many newly proposed systems with as yet untested design characteristics (i.e. those without direct experimental data to back up the approximations made in the simplified modeling). High-fidelity solutions can also be used to enhance the existing simplified models by providing improved methodologies for obtaining reactivity coefficients and heat transfer coefficients.

The neutron transport equation is volumetrically coupled nonlinearly to thermal hydraulics via fuel and coolant temperature and density. Both neutronics and thermal hydraulics are coupled to the structural behaviors of fuel rods (and other components as shown in Figure 4). At present, the SHARP and CESAR projects are studying the coupling between neutronics and thermal hydraulics. The details of the discrete formulations of the governing equations are reported in companion papers (Fischer et al., 2007, 2008; Kaushik et al., 2009) and are not essential to understanding the intermodule coupling problem.

When the physics is split into separate components, these couplings take the form of specific data values that are interpolated from source to target meshes and sent between the modules through well-defined interfaces. To solve the governing equations, SHARP includes two physics modules: (1) Nek (Fischer et al., 2002, 2012), a spectral-element code that solves the three-dimensional, incompressible (Boussinesq), time-dependent Navier–Stokes equation with conjugate heat transfer on unstructured, higher-order quadrilateral meshes; and (2) UNIC (Kaushik et al., 2009; Smith et al., 2011), an unstructured, deterministic neutron transport code that incorporates parallel even parity, sweeping, and ray-tracing algorithms.

Separating physics modules into distinct components and implementing coupling as interactions between these components impose a number of requirements on the overall design of the SHARP framework, discussed in Siegel et al. (2007). SHARP couples physics components and other services through the spatial discretization or mesh abstraction provided by MOAB (Tautges et al., 2004, 2011b,c), which is introduced in Section 4.1.3.

New research in the CESAR project is building on this foundation to develop a coupled, next-generation nuclear-reactor-core simulation tool capable of efficient execution on exascale computing platforms. The approach incorporates tight coupling of extreme-fidelity models of neutron transport and conjugate heat transfer.

2.1.4 Multiscale methods in crack propagation. Most physical phenomena of interest to humankind involve a range of temporal and spatial scales. For example, the brittle fracture of solids under external stresses can lead to the failure of large structures such as bridges or ships, but originates at the atomic scale with the propagation of microcracks. Likewise, tidal currents in bays extend over many miles, but their behavior is dictated by the water viscosity, determined from the molecular-scale interactions of water molecules. A computational approach that captures the complexity of such physical systems must be able to treat all the relevant scales at the level of accuracy required for a realistic description. In this section and the next, we illustrate the issues involved in such multiscale simulations through two examples that require a broad range of scales, from the continuum to the atomistic and even to the electronic level. (Lu and Kaxiras, 2005, review similar problems and methods for their simulation.) The two examples are chosen to demonstrate both the progress that has been made and important challenges that remain in the areas of hard and soft materials.

In this section, our example involves hard materials, specifically the modeling of crack propagation and fracture of solids. Crack propagation is often the result of chemical

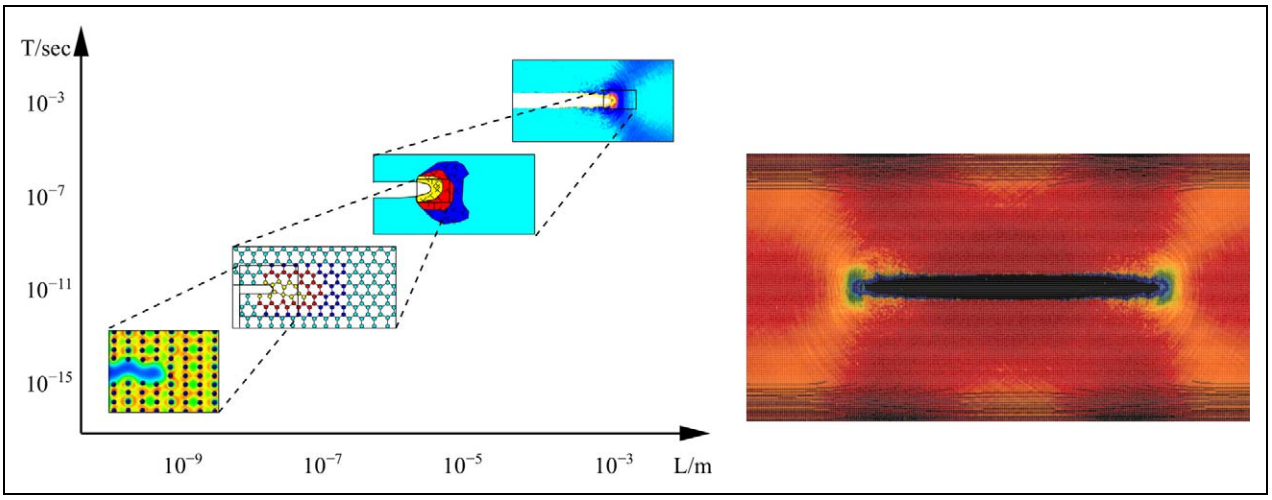


Figure 5. Illustration of the length and timescales in crack propagation. **Left:** Schematic representation of the length and timescales involved. **Right:** Multiscale simulation of crack propagation in silicon, a prototypical brittle solid, which includes all the relevant length scales; the color maps show the distribution of stress (highest at the tip of the crack), with stress waves passing seamlessly from one region to the next (Abraham et al., 1998).

impurities (corrosion), which can change the mechanical behavior of a solid from ductile (tough, difficult-to-break solids) to brittle (weak, easily breakable ones). The length scale at which this phenomenon is observed in everyday situations is roughly millimeters and above. At this scale cracks are detectable by the naked eye and, if not arrested, can lead to the failure (occasionally catastrophic) of large structures such as ships, airplanes, and bridges. The presence of chemical impurities, often in minute concentrations (a fraction of a percent or lower), changes the nature of the bonds between the atoms, which is crucial in the region near the tip of the crack where bonds are being stressed to the point of breaking. These atomic-scale changes are ultimately responsible for the macroscopic-scale change in the mechanical behavior of the solid.

A schematic breakdown of the different length and scale regimes relevant to this phenomenon is illustrated in Figure 5. The macroscopic scale (10^{-3} m or above) on which the crack propagation is of practical interest contains a huge number of atoms, of order 10^{24}, and can be handled only by continuum approaches such as elasticity theory. The response of the solid to external stresses is determined by the motion of defects such as dislocations, which provide the mechanism for deformation. Their distribution and motion at the mesoscopic scale (10^{-5} m) can be described by theories that explicitly take into account these defects and the pattern of stress they create within the solid (Peierls, 1940; Nabarro, 1947; Hirth and Lothe, 1992; Devincre and Kubin, 1997). The origin of dislocations and the nature of the interactions between them can be described only by considering their structure at the microscopic scale (10^{-7} m) in terms of atomic degrees of freedom interacting by springlike forces (Bulatov et al., 1998). Furthermore, the presence of chemical impurities and their effect on atomic bonding, and by extension of the forces that hold atoms together near the tip of a crack, must include a quantum-mechanical description of the ion–electron interactions that can capture the chemical properties of different types of atoms. This scale can be treated in the context of density functional theory using, for example, methodologies such as the Car–Parrinello approach (Car and Parrinello, 1985). A first attempt to capture all these scales for the case of fracture in silicon, a prototypical brittle solid, was the multiscale atomistic ab initio dynamics approach, which was able within the same method seamlessly to couple the quantum-mechanical scale to the classical atomistic and to the continuum (Abraham et al., 1998). A direct link between the atomistic scale and the scale of defects (dislocations, domain boundaries, etc.) dynamics was accomplished by the quasi-continuum method (Tadmor et al., 1996), which enabled realistic simulations of the large deformation of materials (Smith et al., 2001; Tadmor et al., 2002). These examples addressed, in an effective manner, many of the computational issues in the coupling of length scales for multiscale simulations of materials.

In both this hard-material example and the soft-material example, which is described in the next section, there are a number of common issues that affect the accuracy and tractability of simulations. These include boundary conditions, model stability, error propagation, and timestep limitations. These issues are discussed at the close of the next section.

2.1.5 Multiscale methods in ultrafast DNA sequencing. In this section, we examine models for soft materials by applying multiscale methods to the problem of ultrafast sequencing of DNA through electronic means, which is a concept being vigorously pursued by several experimental groups (Kasianowicz et al., 1996; Meller et al., 2000; Li et al., 2003; Storm et al., 2005; Dekker, 2007; Fyta et al., 2011). The idea is to form a small pore, roughly of the same diameter as the DNA double helix, and detect the DNA sequence of bases by measuring the tunneling current across two

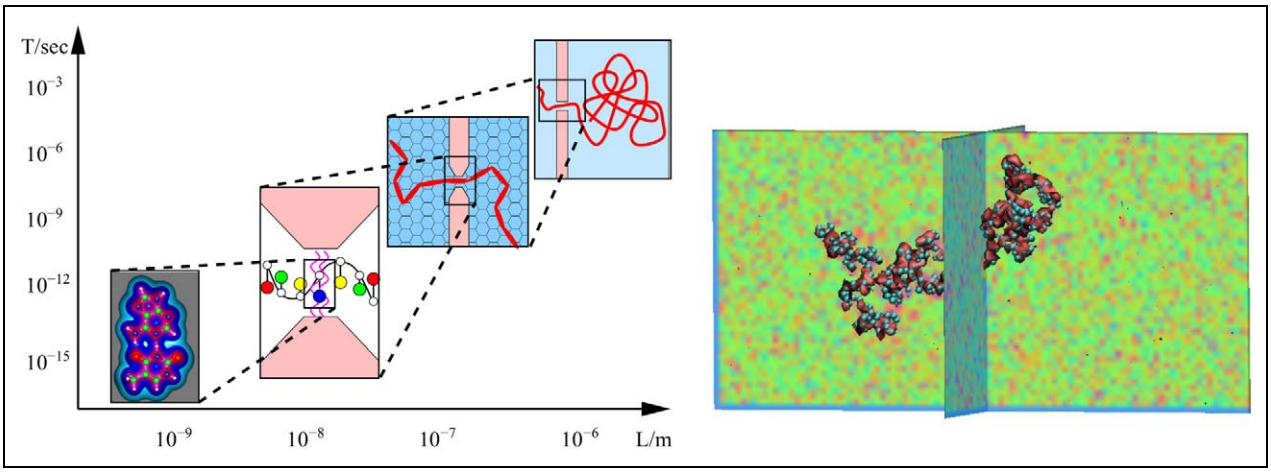


Figure 6. Illustration of the length and timescales in ultrafast DNA sequencing by electronic signals during translocation through nanopores. **Left:** Schematic representation of the process and the various scales involved. **Right:** multiscale simulation coupling the molecular scale of DNA, represented by the blue spheres at the molecular persistence-length scale, to the motion of the fluid solvent whose local velocity is given by the colored contours.

electrodes at the edges of the pore during the translocation of the biopolymer from one side of the pore to the other. Estimates of the rate at which this sequencing can be done are on the order of 10 kb per second, which translates to sequencing the entire human genome in a couple of days. While this goal has not been achieved yet, the successful demonstration of this concept has the potential to produce long-lasting changes in the way medicine is practiced.

This system is also complex, involving several scales that are shown schematically in Figure 6. At the coarsest scale, the biopolymer in solution needs to be directed toward the nanopore. The scale of the biopolymer in solution is on the order of micrometers, and the timescale involved in finding the pore is on the order of fractions of a second. At these scales, the system can be reasonably modeled as consisting of a continuous polymeric chain in a uniform solvent under the influence of an external non-uniform but continuous field that drives it toward the pore. Understanding the translocation process involves modeling the polymer at scales set by the persistence length (~ 50 nm for DNA), while including the effect of the solvent. The coupling of the molecular motion to the fluid motion has been successfully demonstrated, with the solvent described by the lattice-Boltzmann approach to modeling fluid dynamics (Benzi et al., 1992; Succi, 2001) and the polymer motion described by Newtonian dynamics under the influence of local friction from the fluid motion (Fyta et al., 2006). This lattice-Boltzmann plus molecular-dynamics (LBMD) description has successfully reproduced details of the observed translocation process in a realistic manner, as observed in experiments for both single-file and multi-file translocation. An extension of the LBMD method to blood flow in the main arteries of the heart, including the explicit dynamics of red blood cells, was able to capture both the effects of non-Newtonian blood flow on the arterial walls (the "endothelial shear stress" linked to heart disease) and the formation of correlated cell motion (Peters et al., 2010). The coupling to the atomistic and electronic scales, required to capture the whole process of electronic sequencing during translocation, remains a significant challenge. The electron current can be calculated from quantum-transport theories, once the relevant electronic states are known. To this end, detailed calculations of the electronic properties of DNA bases need to be carried out, at the level of quantum chemistry, including all the details of the atomistic structure dictated by the morphology during the translocation process. Assembling the entire picture in a coherent model is a serious challenge to current computational capabilities and is likely to require significant conceptual and algorithmic innovations in the way we handle the phase-space of complex systems.

The examples outlined in this and in the previous section illustrate several of the issues that arise in the context of multiscale simulations for hard and soft materials. The physics coupled in both cases is the motion of atoms and electrons, at the finest scale, as dictated by boundary conditions on the system at macroscopic scales, where continuum theories can be applied. The two ends of the spatial scale are intimately coupled, with the atomic motion constrained by the imposed boundary conditions and, in turn, determining the response of the system to the externally applied stress. The information passed between the scales is essentially the forces on the key components at each scale, which are derivatives of the appropriate energy functionals. These functionals have explicit dependence on degrees of freedom that are coupled across the scales.

Two key sources of errors and instabilities in these multiscale simulations are the differences in the energy functionals across two scales and the loss of information in passing from the finer to the coarser scale. The differences in energy functionals lead to the appearance of "ghost forces" since the physical quantities that underlie each

functional may have a range much larger than the boundary region between the two scales. A case in point is the range of electronic wavefunctions, which can extend over much larger distances than what is practical to capture with a multiscale method involving a quantum-mechanical description at the finest length scale (for metallic solids, these wavefunctions may extend over the entire range of the solid). The second limitation, the loss of information due to coarsening of degrees of freedom, is intrinsic to the multiscale problem and can be addressed to some extent by performing finite-temperature simulations at the atomistic scale. These, however, are both extremely computationally demanding and restricted to a single temperature value. A more general framework is required to overcome this limitation.

The most severe limitation of multiscale methods that include atomistic or electronic scales is that the timestep is dictated by the dynamics at the smallest scale, which for atomic motion is on the order of 10^{-15} s. Interesting methodologies for extending the timescale have been proposed (see e.g., Voter, 1997a,b), but they have their own limitations in the size of system that can be treated or the range by which the timescale can be extended. This issue remains one of the most challenging theoretical problems in performing multiscale simulations for materials problems that can be of direct relevance to real applications.

2.1.6 Magnetic confinement fusion. Magnetic fusion is a long-term solution for producing electrical power for the world, and the large international device ITER (2012) being constructed will produce net energy and a path to fusion energy, provided the computer modeling is accurate. Fusion computation is enormously difficult because of the wide range of temporal and spatial scales in fusion devices (Post et al., 2004; Tang et al., 2009). Consequently, the fusion community has, over decades, developed a suite of models for tokamaks (and other confinement devices).

A model that quantitatively accounts for all the interacting physical processes at work in a tokamak discharge has come to be known as a whole device model (WDM). Such a model must account for nonlinearly interacting phenomena, including sources of energy and particles into the plasma; transport of energy, particles, and momentum through the plasma; the interface between the hot central plasma and the cool edge plasma and the material walls; and the interaction of the plasma with external magnetic coils, circuits, and control systems. While work of this kind has been going on in fusion research for a long time, limitations of theory and computer power traditionally have restricted such modeling to much-simplified models for the physical processes. Including this physics in a single, integrated, multiphysics, multiscale simulation presents many challenges of physics, applied mathematics, and computer science. Beyond the mathematics and computer science required to realize the physics components in an integrated simulation, there are important mathematical issues related to model coupling and stable, accurate timestepping algorithms, as well as computer science issues of data communication and efficient management of very large-scale computer resources.

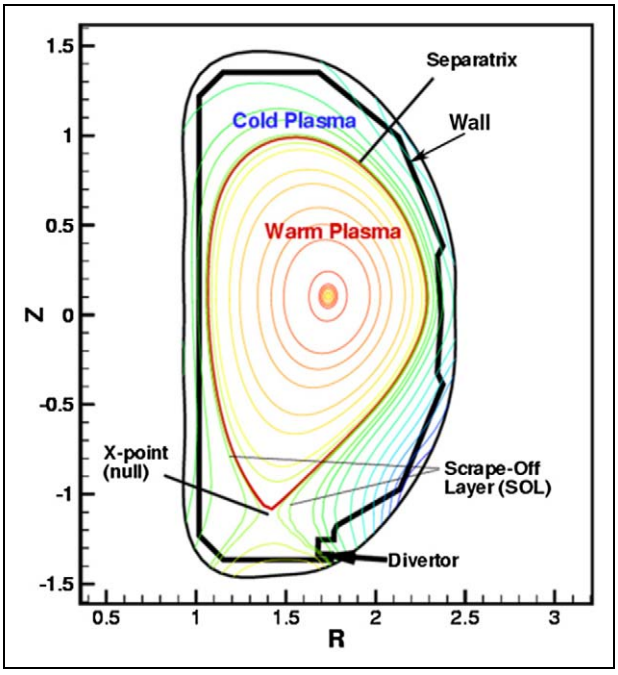


Figure 7. Schematic diagram of a tokamak cross-section. The thick black line shows the material wall. The region inside the separatrix (closed brown contour) is the core, while outside is the edge. The physics in the core, edge, and material walls occur at different timescales and include different physical phenomena. Coupled simulations are needed to self-consistently understand the physics of the tokamak in order to optimize the fusion gain.

As an example of one significant WDM project, FACETS, the Framework Application for Core-Edge Transport Simulations (Cary et al., 2009, 2011; Hakim et al., 2012), has the goal of providing modeling of a fusion device from the core to the wall. Figure 7 shows the cross-section of a tokamak plasma, which has a toroidal (donut) shape. The separatrix (so labeled) separates hot plasma in the center from the cooler, surrounding plasma. The central plasma is hot because it can move primarily only along the magnetic field lines, and those always remain in the core of the system. Thus, the plasma is never in direct contact with material walls, which would cool the plasma. In the edge region of the plasma, just outside the separatrix, the magnetic field lines (projections on a poloidal plane shown in green) wrap around the core but then terminate on the walls. This contact with the walls leads to much cooler plasma.

In the two regions, the plasma is analyzed with different approximations. In the core, the velocity distributions of the particles are nearly Maxwellian, and bulk parameters, such as temperature and density, are nearly constant along field lines, and hence, on flux surfaces, the surfaces on which the field lines remain. This constancy in flux surfaces implies that the plasma only has one-dimensional variation, on the dimension that crosses flux surfaces. In the edge, the plasma varies strongly along a field line, from the

region in the midplane near the core to the ends, which are in contact with the walls. Hence, in this region, the plasma has two-dimensional variation. Being cooler, the edge region can have a high density of atomic species, whereas in the core, the plasma is for the most part fully ionized.

Just inside the separatrix is a transition region, where the cool, two-dimensional plasma of the edge morphs into the hot, one-dimensional plasma of the core. The atomic physics processes become less important, and the character of the turbulence, which controls the plasma confinement, changes. In this region a pedestal may form. At the pedestal there can almost be a transport barrier, such that the temperature of the plasma can be very different across it. Good pedestals are needed for the beneficial H-mode of operation.

Important to the dynamics of this system is how the edge plasma interacts with the wall. The fuel for fusion is hydrogen (actually, its isotopes, deuterium and tritium). The nuclei of these atoms can diffuse into the wall and be released at a later time. These processes are important to fusion confinement devices because they act as boundary conditions on the edge plasma, determining, in part, the influx of particles.

FACETS uses a component-based approach to couple the various physics. Currently, the core, edge, and wall equations are evolved separately and coupled with an explicit coupling scheme; preliminary work on implicit coupling is promising. The coupling between the core and the edge involves a transformation of the two-dimensional edge data to a boundary condition for the one-dimensional core equations, while a reverse transformation is required to transfer fluxes from the core to the edge. The edge–wall coupling is a pointwise coupling in which each cell on the edge boundary is coupled to a wall component.

Coupled whole-device simulations of tokamaks are required in order to understand and optimize fusion gain. The current approach to coupling is essentially code-to-code coupling between existing physics components that were not originally designed for this purpose. Hence, the challenge is not only to retrofit these components to work with one another but also to design robust and accurate algorithms that evolve the coupled physics. Further, developing new WDM components that take into account emerging high-performance architectures is a significant challenge that needs to be addressed in order to satisfy the modeling requirements as ITER and other tokamaks become operational.

2.1.7 Subsurface science. Problems in hydrogeology encompass a broad range of multiphysics and multiscale issues. In this section, we discuss several solution and coupling approaches in the area of subsurface fluid flow, solute transport, and reactions.

First, we consider continuum-scale subsurface reactive transport. For several decades researchers have studied the coupling of fluid flow, multicomponent solute transport, and geochemical reactions. In the 1980s, researchers (Rubin, 1983; Cederberg et al., 1985; Lichtner, 1985) developed formulations for coupling multispecies chemistry with multicomponent transport. Yeh and Tripathi (1989) and Steefel and MacQuarrie (1996) provide overviews of these techniques, which can be divided into three general categories: (1) the global implicit approach (GIA), where solute transport and geochemical reaction are tightly coupled and solved simultaneously in a single system of nonlinear equations; (2) the sequential iteration approach (SIA), where solutes are transported independently, then coupled through geochemical reaction, with the process iterating until the global residual equations converge to a prescribed tolerance; and (3) the sequential non-iterative approach (SNIA), where the solutes are transported independently and coupled through geochemical reaction, but with the process not iterated to convergence. The SNIA approach is more commonly referred to as "operator splitting." Steefel and MacQuarrie (1996) present a concise overview of these three approaches and provide examples illustrating the trade-offs of each.

Through the years, researchers have argued in favor of SNIA or SIA over the fully coupled GIA because of GIA's excessive computing requirements (Yeh and Tripathi, 1989). However, both SIA and SNIA, with less coupling, present their own challenges regarding accuracy and performance (Valocchi and Malmstead, 1992; Barry et al., 1996; Kanney et al., 2003a,b; Carrayrou et al., 2004). SNIA is known to introduce operator-splitting error in the solution, and SIA may result in prohibitive iteration counts when geochemical species are tightly coupled to the immobile phase (e.g., strong sorption, mineral precipitation–dissolution with fast reaction kinetics). An interesting modification can be made to SIA by selectively coupling a subset of strongly coupled reacting species and solving the resulting subsystem of equations with GIA within the SIA framework (Robinson et al., 2000). This selective coupling can reduce SIA iteration counts by two orders of magnitude and model runtimes by more than one order of magnitude.

Recent advances in high-performance computing (HPC) have enabled the use of GIA to solve three-dimensional problems comprising large numbers of chemical species. The feasibility of GIA has been demonstrated on a groundwater-flow and reactive-transport problem involving tens of millions of degrees of freedom simulated on thousands of processor cores (Hammond and Lichtner, 2010). With the use of increasing numbers of processor cores (e.g., 10,000 or more), simulating these larger nonlinear systems of equations requires the development of new solver algorithms to preserve nonlinear and linear iterative solver iteration counts and to minimize communication requirements. Physics-based preconditioners have been investigated for reactive transport (Hammond et al., 2005), where the GIA Jacobian operator is split into two separate operators for global transport and local reaction. These two transport and reaction operators are then applied sequentially in the preconditioning step, much in the same way that operator splitting is applied as a solution technique. This physics-based preconditioner performs well for

subsurface problems composed of primarily aqueous reactions, but more poorly for scenarios where one would expect larger operator-splitting error (e.g., fast reaction kinetics and solute retardation through sorption and mineral precipitation).

Next, we go beyond the continuum scale and consider multiscale approaches that couple pore-scale to continuum-scale models. At the pore scale, the geometry of solid grains and pore spaces is explicitly defined, and the physics represented are Navier–Stokes equations for single-phase fluid flow, advection, and diffusion as transport processes, and various aqueous and heterogeneous reactions (both equilibrium and kinetic). Pore-scale models take various forms, including traditional mesh-based computational fluid dynamics (CFD) methods, lattice-Boltzmann methods, pore network models, and fully Lagrangian smoothed-particle-hydrodynamics (SPH) particle methods (Gingold and Monagham, 1977).

In this multiscale approach, at the continuum scale, the system is treated as a ''porous medium'' without consideration of explicit grains and pores. Continuum-scale physics includes conservation of mass combined with Darcy's law for single-phase fluid flow, mean advection, and dispersion (including diffusion and effects of unresolved velocity variations) as transport processes, and a similar set of chemical reactions. Most continuum-scale models use traditional numerical methods (finite difference, finite element, finite volume) to discretize and solve the system of PDEs, as discussed above.

Since efforts in this treatment of multiscale–multiphysics coupling in subsurface reactive-transport problems have been initiated only recently, the literature contains only a small number of examples. These use both concurrent and hierarchical approaches. In concurrent methods, different physics are solved on separate subdomains with exchange of information and boundary condition matching at the boundaries between the subdomains (or in some cases on an overlapping region). Balhoff et al. (2007) used a concurrent approach to couple a pore network model of flow within a sand-filled fracture to a continuum model of flow in the surrounding porous matrix. Tartakovsky et al. (2008) have developed a coupled pore-continuum model of a diffusion/reaction system in which SPH is used at both scales; this allows straightforward model coupling at the domain boundary but is difficult to adapt to situations in which advective flow exists. Mortar methods (Balhoff et al., 2008) also provide a means of coupling subdomains with different physics by solving a boundary-matching problem on the mortar grid. Battiato et al. (2011) used an iterative boundary-matching scheme to couple pore- and continuum-scale models of reactive transport including advective flow. Alternatively, hierarchical methods solve both models simultaneously over common domains and pass information between scales using various approaches. Tartakovsky and Scheibe (2011) have successfully applied a hierarchical method called dimension reduction with numerical closure to the simulation of mixing-controlled mineral precipitation. In this approach, conceptually similar to the heterogeneous multiscale method of E et al. (2003), macroscopic properties and states are defined in terms of pore-scale variables, and short bursts of pore-scale simulation are used to provide numerical closure for a given set of local conditions. Then, a larger time-step is taken with the macroscopic model (eliminating a significant amount of pore-scale simulation) before it is necessary to update the numerical closure to reflect new macro-scale conditions.

2.1.8 Surface and subsurface hydrology. A blueprint for modeling fully integrated surface, subsurface, and land-surface processes, originally proposed 40 years ago (Freeze and Harlan, 1969), is now becoming a reality. Although truly coupled models have appeared only recently in the literature (VanderKwaak and Loague, 2001; Panday and Huyakorn, 2004; Jones et al., 2006; Kollet and Maxwell, 2006; Qu and Duffy, 2007; Kollet and Maxwell, 2008), a growing library of models and community of modelers now contribute considerably to our understanding of the coupled terrestrial hydrologic and energy cycles. These coupled and integrated hydrologic models are being used on a range of important issues. The models have common features in that they solve some form of the nonlinear diffusion equation (e.g., the Richards equation) for subsurface water flow and some form of the shallow water equations (e.g., kinematic or diffusive wave) for surface flow. Additionally, these hydrologic models are being coupled to land-surface energy, biogeochemistry/ecology models and to atmospheric models (VanderKwaak and Loague, 2001; Panday and Huyakorn, 2004; Maxwell and Miller, 2005; Kollet and Maxwell, 2006; Maxwell et al., 2007; Kollet and Maxwell, 2008; Maxwell et al., 2011). Updated numerical and computational technologies have, in part, enabled new approaches for modeling these coupled interactions. While these models all take different numerical, discretization, and coupling approaches, they share the common goal of modeling the terrestrial hydrologic and energy cycle as an integrated system, rigorously and mathematically. Research addressing these issues encompasses a range of scales and includes a variety of processes (Ebel et al., 2009). Hydrologic simulation models break into two broad solution categories: a globally implicit formulation, and operator-splitting or Gauss–Seidel methods. Among the globally implicit hydrologic models, there are two distinct formulations for the solution of surface and subsurface flow (Ebel et al., 2009): first-order exchange (VanderKwaak and Loague, 2001; Panday and Huyakorn, 2004) and continuity of pressure (Kollet and Maxwell, 2006). Integrated hydrologic models solve equations for surface flow (generally the shallow wave equations) and subsurface flow (generally the Richards equation) simultaneously (VanderKwaak and Loague, 2001; Panday and Huyakorn, 2004; Jones et al., 2006; Kollet and Maxwell, 2006; Qu and Duffy, 2007; Kollet and Maxwell, 2008; Camporese et al., 2009; Kumar et al., 2009; Park et al., 2009b) and represent both coupling approaches and a range of numerical

implementations of the solution (finite-element methods and finite-difference/finite-volume methods). While not widespread, integrated hydrologic models have been designed to run on large-scale computing infrastructures and, in one case, demonstrate very good weak scaling out to large numbers of processors (Kollet et al., 2010).

ParFlow solves the variably saturated Richards equation in three dimensions. Overland flow and groundwater surface-water interactions are represented through a free-surface overland-flow boundary condition, which routes ponded water as via the kinematic wave equation (Kollet and Maxwell, 2006). The coupled subsurface–overland equations are solved simultaneously using a highly efficient and robust Newton–Krylov method with multigrid preconditioning, which exhibits excellent parallel scaling and allows the simulation of large-scale, highly heterogeneous problems (Ashby and Falgout, 1996; Jones and Woodward, 2001). Simultaneous solution ensures full interaction between surface and subsurface flows; this is particularly important in regions with spring-fed streams, large baseflows, and rivers that alternate between gaining and losing.

As one example of multiscale hydrological modeling, ParFlow solves the coupled water and energy balance at the land surface by incorporating a modified version of the Common Land Model (CLM3.0) (Dai et al., 2003), which calculates evaporation from the vegetation canopy and ground surface; transpiration and plant water-uptake over the soil column; snow accumulation and melting; and latent, sensible, and ground heat fluxes. Vegetation and land-energy processes are calculated at each timestep by using an operator-splitting approach that ensures full mass and energy conservation (Maxwell and Miller, 2005; Kollet and Maxwell, 2008). A fully coupled hydrologic and climate model, PF.WRF was developed by combining ParFlow with WRF (Maxwell et al., 2011). The general solution procedure begins with explicit advancement of the atmospheric code. Using the operator-splitting approach, surface fluxes that are relevant to ParFlow, such as infiltration and evaporation rates, are integrated within the atmospheric model over the ParFlow timestep. These accumulated surface fluxes are applied to ParFlow at the start of its implicit time advancement. The subsurface moisture field calculated by ParFlow is passed directly to the Noah land-surface model (Mitchell, 2005) within WRF for use in the next timestep. Noah is advanced for each WRF timestep to provide the necessary land-surface fluxes, but soil moisture values are now specified by ParFlow. ParFlow is incorporated as a subroutine in Noah, with communication over all Noah soil layers; subsurface hydrology in Noah is entirely replaced by ParFlow. The combined model is fully mass and energy conservative.

2.1.9 Climate modeling. Earth system modeling, specifically coupled global climate models (GCMs), is one of the truly definitive multiphysics applications in several respects. GCMs encompass physical behavior ranging in spatial scales from molecular dynamics to global atmospheric wave patterns. Moreover, GCMs capture climate variations covering months to 10,000 years over regional to global areas of the Earth's surface. A number of GCMs are under active development around the world, and the Community Earth System Model (CESM), which has been developed by hundreds of scientists and contributes to the Intergovernmental Panel of Climate Change Assessment Report (Solomon et al., 2007), is a good example to illustrate the multiphysics nature of GCM. In most cases, the multiphysics characteristics of the CESM apply to other GCMs.

The CESM comprises five Earth system components (atmosphere, land surface, ocean, sea ice, and ice sheets) and has provision for future components, as shown in Figure 8. These components periodically exchange data as fluxes at component boundaries through a flux coupler, which coordinates the transfer of coupled physics across model components, except for ice sheets, which currently receive one-way surface–mass balance values from the land model. Typically, coupling occurs every simulated day, although recent high-resolution coupled model development has indicated the necessity of exchanging this information more frequently (Dennis et al., 2012). The source-model flux data is interpolated to the target component by using second-order or conserved mapping weights within the coupler. Coastal "masks" which account for the boundary between land surface and ocean are created as a data file to determine whether atmosphere model grid cells couple to the land surface or ocean below and vice versa. These masks need to be created for each combination of component grids used in a coupled-model run.

Within each model component, resolved physics are represented by discretized equations referred to as the dynamical cores, or dycores. Currently, all the physics and dynamics variables exist on the scale of the three-dimensional discretized grid, although the land model performs computations using statistical representations of subgrid-scale land-use characteristics. The atmospheric component in CESM, the Community Atmosphere Model (CAM), has several dycore options that scientists can choose based on desired attributes such as computational efficiency and local conservation properties. Each model component uses a different strategy for solving the model equations. For details, the reader is referred to the model documentation (Gent et al., 2011). Unlike CAM, which has several choices for spatial discretization, the ice sheets, sea-ice, and ocean components in CESM use a finite-difference scheme on regularly spaced grids. Their time-integration methods are diverse, arising from the different equations and physical scales being treated in each component. These methods include subcycled fully explicit (sea-ice), semi-implicit to "step-over" barotropic dynamics (ocean), and fully implicit velocity with subcycled advection (ice-sheets) methods. Several of these components have developmental dycores using an unstructured grid to allow local refinement (Ringler et al., 2011) and a testbed of fully implicit algorithms to overcome stability limitations (Evans et al., 2009).

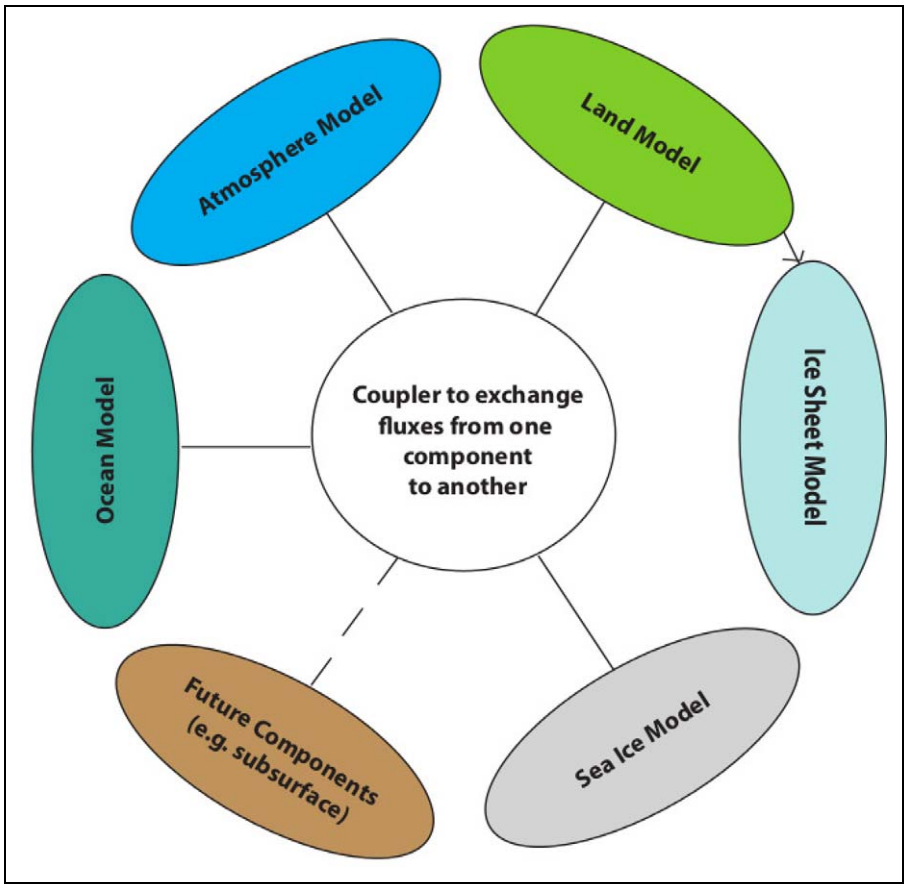


Figure 8. A schematic diagram illustrating the multi-fidelity structure of the Earth-system model. Except for the ice sheet model, all components in the model mutually couple through the flux coupler, which is shown as the hub of the diagram. The ice sheet model does not couple in the same way. It is coupled only to the land model, receiving surface-mass balance values. As is shown, this coupling is one way: the land model receives no feedback from the ice sheet model.

As mentioned above, CAM has several dycore options for scientists to select when configuring the coupled climate model for experiments. These options include a spectral-element-discretized cubed-sphere configuration, which uses a subcycled fully explicit numerical scheme. Other dycore options include an Eulerian spectral and spherical finite-volume discretization, which both use an operator-splitting, semi-implicit time-integration method to separate waves of different scales. In this case, a common splitting of advection and gravity waves adapted from the geophysical fluid dynamics community is employed because it takes advantage of a natural separation at coarse resolution, and favorable numerical properties of the resulting equations. The atmospheric dynamical core resolves horizontal waves ranging from vertically propagating gravity waves (∼200 m/s) to low-frequency global modes within the background flow (∼2–20 m/s). Vertical motions are smaller in scale and are represented by the hydrostatic approximation, which assumes that the vertical pressure gradient is largely balanced by gravity and is valid to scales down to individual cloud systems of about 100 km in extent.

Climate is uniquely challenging because the most important drivers of global climate change (chemistry, moisture, and radiation) vary over microscopic spatial scales but impact the global-scale net energy balance of the Earth system. The global-scale atmospheric flow is coupled to local chemical, cloud, and radiative forcings; and the time scale for this coupling ranges from the diurnal cycle to decades. Subgrid-scale effects are treated either as prescribed values from datasets or as separate, vertical-column, horizontally independent models that are updated as forcings to the dynamical equations. These independent models are themselves multiphysical. For example, the chemistry model in CESM (Horowitz et al., 2003) uses a split explicit–implicit solution method for the slow and fast species reactions, respectively.

2.1.10 Radiation hydrodynamics. Physical processes that involve radiation or hydrodynamics individually are part of many phenomena of everyday life and everyday scientific research. However, phenomena where radiation and hydrodynamics couple and stand together on an equal footing, where strong coupling between the two components affects the evolution of each, rarely occur terrestrially but are common astrophysically. The energetics involved explain this apparent dichotomy. In astrophysics, RHD plays a part in highly energetic phenomena, such as stellar core collapse, supernova shock propagation, supernova

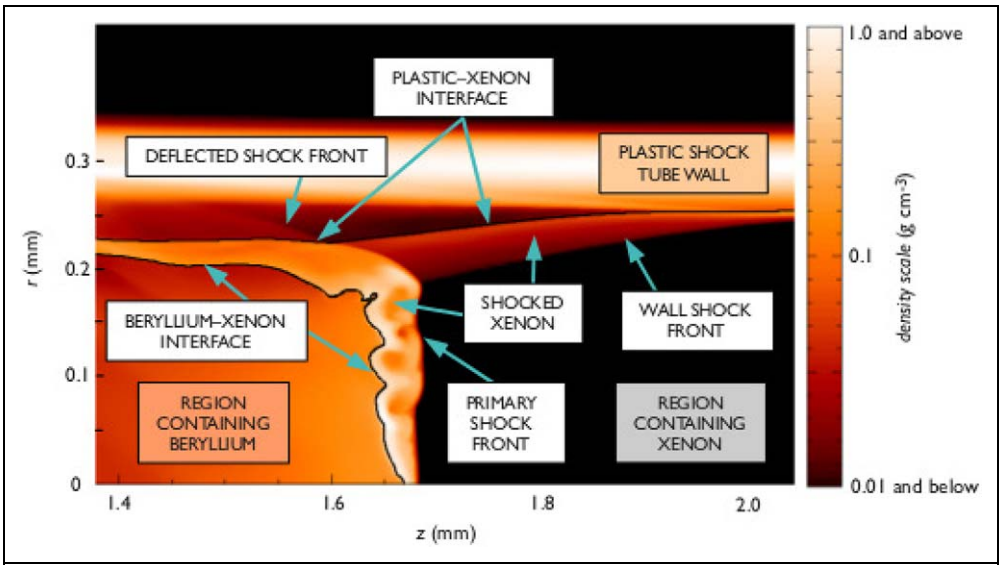


Figure 9. Interacting shock waves propagating in a simulation of a high-energy-density physics shock-tube experiment (Drake et al., 2011). The figure shows a slice representing a half cross-section of the cylindrical shock tube with the central axis at the bottom of the figure. In addition to the shock fronts, a number of material interfaces are shown, which represent boundaries between the multiple materials used in the experiment (beryllium and xenon within a polyimide-plastic shock tube). The horizontal extent of this schematic represents roughly 0.6 mm of a shock tube that is about 4.0 mm in total length.

remnants, stellar pulsations, and accretion flow. Where radiation-hydrodynamic phenomena occur terrestrially, they are restricted to high-energy density environments: inertial-confinement fusion, laboratory astrophysics, and high-temperature shock-wave and explosive phenomena.

In addition to RHD being a multiphysics field, in that it strongly couples the evolution of distinct physical components (matter and radiation), many problems in RHD introduce other multiphysics aspects. For example, a single problem can involve any number of the four fundamental interactions across multiphysics domains and can mix classical- and modern-physics paradigms (e.g., classical or relativistic space–time and gravity, quantum statistics, electroweak, and strong nuclear interactions).

Numerical RHD models are typified by a number of common features (Pomraning, 1973; Mihalas and Mihalas, 1984; Castor, 2004): conservation PDEs play a fundamental role (e.g., mass, species, energy, momentum). As a result of the energies involved, shock waves are frequently present in such systems, leading to discontinuities in physical quantities. Additionally, material interfaces, which are part of many models of terrestrial setups, introduce another form of discontinuity into the problem. Some RHD problems can be classified as reactive-flow problems, as evidenced by complex and strong coupling terms between components. Figure 9 shows an example of the complexity of these systems: interacting shock waves are propagating in a simulation of a terrestrial shock-tube experiment as part of the CRASH project (Drake et al., 2011). A beryllium foil is driven into a xenon-filled shock tube by a drive laser to speeds in excess of 100 km/s. This process generates a radiative primary shock wave, which in turn generates a wall shock as the plastic walls of the shock tube are heated and ablated.

Other complexities also are common in RHD models. Frequently, the equations for the radiation are nonlinear, requiring solution by nested iterative procedures, such as preconditioned Newton–Krylov methods. (These nonlinearities can be caused by strong coupling between matter and radiation as well as by quantum-mechanical degeneracy effects.) Coupling between physical components can be a sensitive issue because, in complex phenomena, coupling varies in strength as a function of time, position, and/or energy and thus can require different solution approaches in different situations. High dimensionality is also an issue. Proper treatment of the radiation requires solution of the Boltzmann transport equation over six-dimensional phase space as a function of time. Almost always, finite computer resources force a compromise, leading to reduced dimensionality in some fashion (Graziani, 2005). Such compromises can affect self-consistency of coupling (e.g., radiation–matter momentum exchange in flux-limited diffusion approximations).

Since numerical solution of RHD problems predates the computer age, solution techniques are many and varied. Today, however, solutions typically involve the solution of Euler's equations for the matter component. (Reynolds numbers are typically very high in such systems.) Standard methods include finite-difference and finite-volume methods. Lagrangian, Eulerian, and hybrid approaches (e.g., ALE methods) are used. Such techniques also translate to solution of the radiation (Bowers and Wilson, 1982; Bruenn, 1985; Turner and Stone, 2001; Swesty and Myra, 2009; Van der Holst et al., 2011). Here, however, finite-element methods also find use (Lewis and Miller, 1993).

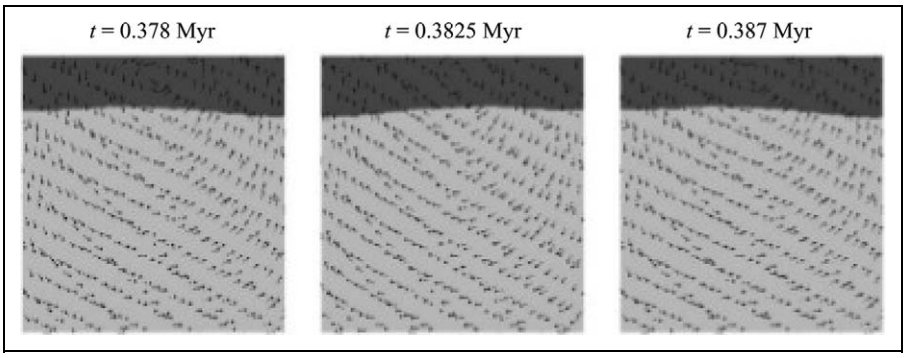


Figure 10. The "drunken sailor" instability. In a simple case of a compositionally (grey shading) driven Rayleigh–Taylor instability, an explicit coupling between the incompressible Stokes equations and the kinematic free-surface on the top boundary leads to a sloshing instability in the velocity field (vectors) (Kaus et al., 2010).

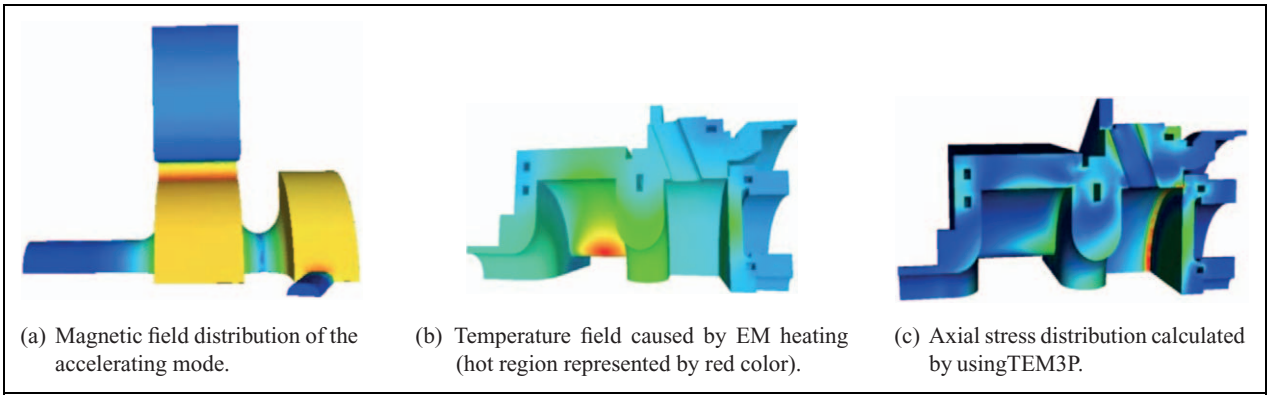
When a similar discretization is used for both matter and radiation, coupling between components reduces to a problem of obtaining consistency and accuracy of paired coupling terms. However, when different discretizations are used for matter and radiation (e.g., finite volume for matter and finite element for radiation), then accuracy of the transfer operator can become problematic. Unlike frequently encountered multiphysics couplings discussed in this report, matter–radiation coupling occurs, in some strength, over the entire physical domain of interest; it is a volumetric coupling. Methods for improving the accuracy of this coupling are an area of active research.

RHD models commonly use a mixture of explicit and implicit approaches within the context of operator splitting. Choices stem from the timescales involved. Typically, hydrodynamic timescales are much greater than radiative timescales. This situation leads naturally to use of explicit methods for the matter and implicit methods for the radiation (usually sparse iterative approaches for the latter). However, for problems with long evolutionary timescales (e.g., neutrino emission for cooling neutron stars (Reynolds et al., 2008)), implicit approaches increasingly are being used for both components.

The difficulties inherent in adequate treatment of multiphysics coupling in RHD models are illustrated by a sample list of the myriad of relevant timescales involved: acoustic, radiation-transport and radiation-source, electron–ion equilibration, electron diffusion, and viscous (if present). Arrayed in relative size, these may span a range of 10^{15}. A challenge for the application modeler is to determine which of the associated processes are to be followed in detail and which can be assumed to occur so quickly that an equilibrium approximation is adequate.

2.1.11 Geodynamics. Geodynamics is the study of dynamic processes in the Earth from crust to core. Processes range from earthquakes to plate motions, large-scale convection to microscale melt migration. Physical coupling occurs over a wide range of simulations in geodynamics. Short-term crustal behavior is generally modeled with some combination of a porous, viscous, elastic, or plastic material model (Watts, 2001). Beneath the crust, incompressible mantle flow calculations couple the advection–diffusion of energy/temperature to the Stokes equations (Schubert et al., 2001). The complexity may be increased by introducing the advection of chemical components with low or zero diffusivity, which is often achieved by using particle-based methods (Van Keken et al., 1997). Compressible flow, which normally uses an elastic approximation to filter out seismic waves and timescales, may also introduce reference equations of state derived from models of seismic wave propagation (King et al., 2010). These simulations are often multiscale, including narrow plume conduits in large domains (Tarduno et al., 2009; Davies et al., 2011), along with physical discontinuities in material properties at a variety of depths (Christensen and Yuen, 1985). At greater depths, in the outer core, models couple turbulent flow from the Navier–Stokes equations to thermal and magnetic models of rapidly convecting liquid iron (Glatzmaier and Roberts, 1995; Kuang and Bloxham, 1999).

The change of behavior and physical timescales with depth generally leads to different models being used for different shells of the Earth. Boundary conditions are then used to parameterize the fluxes between models. As an example, owing to high convective vigor in the outer core, the lower thermal boundary condition on the mantle may be treated as a Dirichlet condition (Blankenbach et al., 1989), while the motions of crustal plates may be treated as a rigid shell, with prescribed shear at the upper boundary of the mantle (Bunge et al., 1998). Depending on the application, this outer boundary may also be treated as a free surface, supporting dynamic topography, which involves coupling the solution of a kinematic boundary condition to the flow solution in the global domain. This approach potentially introduces instabilities in incompressible flows (Kaus et al., 2010), as shown in Figure 10. Smaller-scale, higher-resolution studies may occur on overlapping domains. As an example, one could focus on the production of melt in subcrustal regions, such as mid-ocean ridges or subduction zones. The differing temporal and spatial scales of pore-filling fluid and melt lead to calculations that are often multiphase, coupling Darcy's flow with Stokes flow in the mantle or poro-visco-elastic



(a) Magnetic field distribution of the accelerating mode.

(b) Temperature field caused by EM heating (hot region represented by red color).

(c) Axial stress distribution calculated by usingTEM3P.

Figure 11. Multiphysics coupling analysis of a Linac Coherent Light Source (LCLS) gun.

flow in the crust (Spiegelman, 1993a,b; Katz et al., 2007). Experimental, theoretical, and numerical work attempts to link these high-resolution studies to larger domains through parameterizations of rheology and melting (Katz et al., 2003).

2.1.12 Particle accelerator design. A key component technology in particle accelerators is the accelerator cavity, which is used to generate electromagnetic fields that control acceleration of the subatomic particles in the accelerator beam. To achieve high efficiency, a modern accelerator cavity uses superconducting devices to generate the fields that drive the cavity at its resonant frequency.

In radio-frequency (RF) cavity design, finding a sufficiently accurate resonant frequency for real geometry has become possible with the help of a higher-order, high-fidelity finite-element method (Morgan, 1996; Gupta et al., 1997; Jin, 2002; Sun et al., 2002). One of the main sources of operational failures of RF cavities is strongly related to the excessive thermal heating arising from high power or high currents in operation (Reece et al., 2007; Cheng et al., 2011). As the power and current increase for future experiments, optimizing the performance and cost-effectiveness of RF cavities becomes an increasingly important technological challenge for designing next-generation accelerators. TEM3P, a multiphysics analysis tool using finite-element methods, has been developed to capture thermal, structural, and electromagnetic effects such as cavity wall heating (Akcelik et al., 2009), structural deformations (Akcelik et al., 2008a), and Lorentz force detuning (Liepe et al., 2001). Parallel implementation of TEM3P utilizes supercomputers for large-scale computations (Lee et al., 2004; Liao et al., 2006; Li et al., 2007) with high geometric fidelity and high numerical accuracy. TEM3P solves the thermoelastic equation, in which the heat flux and radiation pressure are interpolated from the electromagnetic solution file from Omega3P (Lee et al., 2009), a parallel finite-element code for frequency domain analysis of accelerator cavities.

The design process starts with accurate electromagnetic solutions obtained from Omega3P analysis. Omega3P solves Maxwell's equations with higher-order curved elements for highly accurate solutions. TEM3P reads the electromagnetic solution file (in NCDF format), which is interpolated to produce accurate physical quantities of interest for the given multiphysics application. Heat flux is calculated for the thermal solver in TEM3P (Akcelik et al., 2009), while radiation pressure is calculated for the structural solver in TEM3P (Akcelik et al., 2008a). Interpolation of the electromagnetic fields accounts for the general case, in which the nodal points in the TEM3P mesh do not coincide with the nodal points in the Omega3P mesh. Once the desired heat flux or radiation pressure is obtained, it is applied as a boundary condition to the existing thermal model for an RF heating problem or to the existing structural model for a Lorentz Force detuning problem.

Internally, TEM3P couples its thermal solution to its structural solver. When a temperature distribution is known, thermal stresses due to the high-temperature gradient can be readily calculated from the structural solver. As shown in Figure 11b, the high-temperature gradient in the RF gun is strong enough to cause geometric distortions so that the resulting frequency shifts can be larger than the operating frequency bandwidth (Akcelik et al., 2008a).

Commonly, in multiphysics analysis, the primary source of heat flux is RF heating. In this case, an electro-thermo-structural coupling, as shown in Figure 12 is used to provide viable solutions that address frequency shift problems. TEM3P has also been applied to thermal instability problems in superconducting cavities (Akcelik et al., 2009). Thermal instability can lead to serious problems such as thermal quenching, which terminates the superconducting state of a cavity (Kim, 2003). In the face of this instability, operation is usually not possible, or only at a significantly reduced efficiency if it is possible (Lesrel et al., 1997). Superconductor materials exhibit highly nonlinear thermal characteristics, and nonlinear boundary conditions need to be applied in order to obtain an accurate solution. A robust nonlinear solver is implemented in TEM3P to address these problems arising from the nonlinear material properties.

The current version of TEM3P provides steady-state solutions for both thermal and structural problems. Multiphysics

problems with multiple spatial and temporal scales are not yet addressed in TEM3P. As designs become more concerned with fully time-dependent multiphysics coupling, further research and development become increasingly important.

Shape optimization studies have been performed with adjoint gradient information (inverse eigenvalue problems) (Akcelik et al., 2005, 2008b). Because adjoint variables enable gradient calculation at a given design point regardless of the number of design variables, achieving a near-optimal shape in a reasonable turnaround time has become feasible (Choi et al., 2008; Lee, 2010; Xiao et al., 2011). With the successful implementations of multiphysics coupling, applying its solution to optimization problems could open up new research areas. The solutions from the multiphysics coupling could provide insights in the complex physical interactions not resolvable from single-physics analysis. Thus, combining an adjoint-based gradient optimization with multiphysics coupling could potentially play an important role in achieving near-optimal solutions in multiphysics environments.

2.2 Crosscutting issues in multiphysics applications

The previous subsection contains a diverse collection of multiphysics applications, including brief descriptions of solution approaches, treatment of multiphysics coupling, and various application-specific issues. Here, we extract commonalities and differences from those descriptions. Our goal is twofold: (1) to identify issues in algorithms and software that need to be addressed through, perhaps, different approaches and more rigorous analysis, and (2) to provide pointers to related discussion in the remainder of the document. Many of the implementation issues in multiphysics applications can be classified as relating to either splitting or coupling of physical processes among or between solution modules. We consider both spatial and temporal coupling issues. We also introduce software engineering questions that touch on best practices issues as well as the cultural environment for software collaboration and the hard choices that must be made when mapping an application to specific hardware.

Many multiphysics applications owe their coupling to historical accident. Multiphysics applications have rarely been designed as a first creation. Instead, such applications typically represent the product of collaboration between two scientific groups of diverse fields who find a common interest that merits joint investigation. The result is a marriage of two diverse codes (or algorithms); common practices from a software perspective are discussed in Section 4.1.1. The driving force is the new capability that this union makes available; the drive toward an answer trumps the drive toward an *accurate* answer or a computationally efficient simulation. In spite of valuable scientific advances that may be made, time is rarely made for mathematical rigor. Thus, it is common that, for example, two well-constructed second-order accurate codes are coupled by a quick ad hoc scheme that may be, at best, first order. In this report, our goal is to suggest approaches for more formal analysis of spatial and temporal coupling methods in such areas as accuracy and order of convergence. Of special importance are interfaces between subdomains of a problem, volumetric interfaces, and temporal interfaces.

2.2.1 Choices and challenges in coupling algorithms. Spatial interfaces remain a primary challenge, especially between subdomains using different computational approaches and discretizations (e.g., an interface between two subdomains, one discretized by finite elements and the other by finite volumes). Many commonly used interpolation approaches are non-commutative and thus can lead to non-conservation of supposedly conserved quantities. A number of applications we have described use mortar methods to bridge this difficulty (see Sections 2.1.1, 2.1.2, and 2.1.7). Methods for coupling multiphysics components in space are discussed in Section 3.2.1.

The collection of applications described in Section 2.1 suggests that explicit methods are still prevalent across many fields, especially for "production" applications. This situation is understandable: compared with implicit approaches, explicit methods are easier to implement, maintain, and enhance from a software-engineering perspective, as discussed in Section 4.1.1. Maintaining the stability of an explicit method mandates that the programmer understand at least some the important timescales involved in the problem. Implicit methods are free of splitting errors that are introduced by operator splitting, and may allow the use of much larger timesteps than an explicit method. However, when using an implicit method, timesteps must be carefully selected to control at least local time-integration error in order to avoid "silently" producing inaccurate answers by potentially taking large timesteps over important processes having fast timescales. This is of special concern when, as occurs frequently, the equations being solved are nonlinear. (Nonlinearities can occur in nearly all the application examples above, but notably those described in Sections 2.1.1, 2.1.3, 2.1.6, 2.1.7, 2.1.8, 2.1.9, 2.1.10, and 2.1.12.) In summary, the explicit–implicit debate becomes one of accuracy versus efficiency. Given the increasing use of implicit methods, it is necessary to ask how we should approach judging the accuracy of a method. For example, it is easy to integrate $y'(t) = y(t)^2$ from $y(0) = 1$ in a stable way that completely misses the singularity at $t = 1$ and continues smoothly with unbounded error into $t > 1$. How do we know we are not skipping over physics? What is the collection of metrics we can use to provide confidence in our answers? Which integrated metrics are of greatest value? Which pointwise metrics are reliable?

A number of the applications discussed in this report have timescales for important physical processes that may vary by many orders of magnitude. (See Sections 2.1.4, 2.1.5, 2.1.9, 2.1.10, and 2.1.11.) The challenge for the model builder is to have a clear picture of what quantities need to be tracked in the problem being solved. Such insight helps to determine

which of the associated processes need to be followed in detail and which can be assumed to occur too quickly for their evolutionary details to be important. An additional challenge is to write that intelligence into the code itself. Section 3.2.2 discusses methods for coupling multiphysics components in time, including implicit–explicit (IMEX) and adaptive techniques, which can help address such multiphysics challenges in temporal coupling. Additional opportunities are considered in Section 5.

Since diverse timescales are common in multiphysics applications, it is worth asking whether an approach using UQ methods may yield insight into the relation between timestep selection and results obtained; UQ issues are discussed in Section 3.5.

A related issue involves quantifying certain cultural knowledge that allows one to "cheat" a little on constraints when quantities of interest are not affected. An example is the careful violation of the Courant–Friedrichs–Levy (CFL) condition in explicit codes.

Another interesting challenge is spatial coupling of the discrete to the continuum. This issue arises in problems in which the macroscopic evolution of the system relies on detailed knowledge of phenomena occurring on a microscopic level. Examples include the modeling of crack propagation and the ultrafast sequencing of DNA discussed in Sections 2.1.4 and 2.1.5. An additional class of problems in this vein is non-equilibrium fluid flow: adequate description of state variables in the fluid may rely upon, for example, the result of a simulation at the molecular, atomic, or even electronic or nuclear scale. Problems in RHD (see Section 2.1.10) can fall into this class.

Discrete-to-continuum coupling may also arise at interfaces between diverse materials where, in one material, a microscopic model is required for a satisfactory physical description, while in the other material, a classical continuum description is adequate. Whenever these sorts of couplings arise, one or more additional mesoscale descriptions that link the two extremes has found use. (Section 2.1.4 describes an application that benefits from this approach.) We discuss discrete–continuum coupling more fully in Section 3.3.

Any need for adaptivity in algorithms throughout a given multiphysics simulation also presents challenges. As an example, consider a multiphysics application in which a key piece of science changes in the middle of the model's evolution (e.g., a rate may become much larger or smaller as a result of some change in a linked quantity). This situation often leads to the component equations having a changed character. For example, what might have previously been a parabolic problem might now take on hyperbolic behavior. This circumstance may require a change of solution methods (introduced in Section 3.1), anywhere from the preconditioner to the whole solver approach. Equally extreme are phase transitions: solution methods are likely to be different for a subdomain now containing a solid phase instead of a gaseous phase. These issues, as well as related challenges in supporting changes in modeling and algorithms over the lifetimes of multiphysics applications, are considered from a software-design perspective in Section 4.

2.2.2 Software engineering. The current status of multiphysics software is discussed in Section 4.1, including current practices for multiphysics coupling in production codes, common needs, and examples of libraries, frameworks, and tools that support multiphysics simulations in HPC environments. The development of large-scale multiphysics applications raises many of the same issues as does development of any large-scale application. One of the most fundamental issues is the "buy" versus "build" choice. In the multiphysics world, however, this question takes on increased importance because of the added complexity. The "build" approach frequently requires large collaborations of diverse expertise. However, the sources where one may "buy" are often similarly scattered among disciplines and institutions. Neither approach alone guarantees robust, accurate code. Risk mitigation also plays a role in this choice. On the one hand, third-party code is not designed with the end-user's specific purpose in mind. Learning the code (and perhaps learning that the code is not suitable) can be time-consuming. Even if such a third-party code is suitable, additional investment is required to maintain multiple, interacting software packages. A decision to "build" can be equally risky. In addition to the obvious issues of funding, assembling, and directing a collaborative team, and then having the team deliver on time, there is an ever-present danger that the drive for science leads to compromises in code accuracy, especially where the subtle coupling of multiphysics models occurs. Challenges, near-term needs, and risks in multiphysics software are discussed in Sections 4.2 and 4.3, while longer-term opportunities and insertion paths are considered in Sections 5 and 6.

2.2.3 Analysis and verification. Analysis and verification of coupled models involve issues that span both algorithms and software and have been only partially addressed by the community overall. Sections 3.4 and 3.5 discuss error estimation and UQ for multiphysics problems from an applied mathematics perspective, while Section 4.2 considers issues in software testing. Verification of coupled codes is challenging. Unit tests do not test the synergistic effects of coupling, which is the very aspect of simulation that probably requires the greatest testing. In addition, large multiphysics codes have too many coupling combinations to allow practical coverage of the code space by any conceivable test suite. Furthermore, standard reference problems are lacking. Given this situation, verification tests tend to devolve into regression tests, whose value further devolves when existing physics is changed or new physics is introduced.

3 Algorithms for multiphysics coupling

In this section, we discuss methods for formulating and solving multiphysics problems. The formulation of a

multiphysics problem is highly dependent on the nature of the problem and how the component physics fields are coupled in space and time. In some circumstances, coupling between continuum and discrete models is required. Examples include heat conduction problems that require calculation of temperature-dependent thermal conductivity from first principles, and plasticity problems where empirical stress–strain relationships are not available.

This section also addresses error estimation and UQ for multiphysics problems. The complexity of multiphysics problems makes the assessment of errors and uncertainties both problematic and essential. We note that in practical terms it is often difficult or impossible to obtain solutions of multiscale, multiphysics models that are asymptotically accurate uniformly in space and time. Hence, the fruits of classical numerical analysis may be unobtainable in the study of multiphysics models. Further, while UQ is a relatively young field with many foundational research problems remaining to be tackled, UQ for multiphysics models is simply nascent.

We begin with a general discussion of solver methods in Section 3.1. These algorithms form the building blocks of many multiphysics codes that run on the largest currently available HPC systems. This is followed by a discussion of strategies for coupling physics components in space (Section 3.2.1) and in time (Section 3.2.2). The special case of coupling continuum and discrete models is discussed in Section 3.3. We then discuss error estimation in Section 3.4 and UQ in Section 3.5.

3.1 Solver methods

While multiphysics problems (1a)–(1b) and (2a)–(2b) are inevitably nonlinear, most solution methods require solving a series of linear subproblems. Here we briefly summarize currently available techniques for solving systems of linear equations, followed by a review of methods for nonlinear problems. Related issues in software are discussed in Section 4.

3.1.1 Methods for systems of linear equations. Consider the system of linear equations

$$Au = f \qquad A \in \mathrm{R}^{N_d \times N_d} \quad u, f \in \mathrm{R}^{N_d} \tag{4}$$

where N_d is the number of degrees of freedom. Equation (4) can represent an entire multiphysics problem or some subproblem encountered in a multiphysics solution algorithm. Methods for solving (4) fall into two categories: direct and iterative methods. Direct methods generally rely on some canonical decomposition of A and scale poorly (in both storage and operations) when applied to large problems, even when sparsity in A is effectively exploited. For many years, researchers relied on *stationary* iterative methods, such as classical Jacobi, Gauss–Seidel, or successive overrelaxation methods. While effectively alleviating the large storage requirements of direct methods and being easy to implement, stationary methods lack robustness and generally do not converge fast enough to be considered practical for realistic multiphysics problems. Current best practices for the efficient solving of (4) are based almost exclusively on multigrid methods and preconditioned Krylov subspace methods.

Multigrid methods. Systems of linear equations (4) usually must be solved many times in a multiphysics application. Consequently, linear solvers are often bottlenecks in multiphysics applications. This situation places a premium on algorithms that have optimal arithmetic complexity; ideally, solutions to (4) should be computed with $\mathcal{O}(N_d)$ floating-point operations. Multigrid methods (Brandt, 1973; 2002; Trottenberg et al., 2001) achieve this objective for a large class of problems.

Multigrid methods are based on the observation that different components of error can be more efficiently represented and eliminated on grids of different resolutions. This idea can then be applied recursively to develop solution algorithms that utilize a grid hierarchy to achieve high efficiency. The key ingredients are an inexpensive smoothing method to apply on each level in the grid hierarchy (typically a stationary method), transfer operators to move data between successive grid levels, and a schedule for visiting each grid level in the hierarchy, resulting in the well-known V-cycle, W-cycle, or full multigrid cycle.

Developed in the context of elliptic and parabolic PDEs that typically enjoy smooth solutions, multigrid methods have demonstrated near-optimal $\mathcal{O}(N_d \log N_d)$ scaling on such problems on up to hundreds of thousands of processors on modern parallel architectures (Bergen et al., 2006; Baker et al., 2012; Gmeiner et al., 2012). Variations for anisotropic problems and problems with discontinuities have also been developed. While the early development of multigrid methods focused on constructing coarser problems through reference to the underlying grid, the fundamental properties of smoothing and grid transfer operations were soon abstracted and combined with coarsening strategies that are adaptively based on analysis of the matrix entries. This research led to the class of *algebraic* multigrid methods, which offer the potential for near-optimal black-box solvers for large-scale problems without any reference to an underlying mesh structure. Current research in multigrid methods strives to improve the scalability of coarsening strategies (Henson and Yang, 2000; Gee et al., 2006). High-performance implementations of both geometric and algebraic multigrid methods are available in numerous software libraries (Heroux et al., 2005; Falgout et al., 2011; Balay et al., 2012).

Krylov subspace methods. Let $u_0 \in \mathrm{R}^{N_d}$ be an initial approximation to the solution of (4), let $r_0 = f - Au_0$ be the initial residual, and for each integer $k \geq 1$ let

$$\mathcal{K}_k(r_0, A) = \mathrm{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$$

be the Krylov subspace generated by r_0 and A. Consider first the case where A in (4) is symmetric and positive definite. In this case, A induces an inner product $\langle u, v \rangle_A = u^{\mathrm{T}}Av$

and corresponding proper norm $\| u \|_A = \langle u, u \rangle_A$. For symmetric positive definite A, the conjugate-gradient (CG) method (Hestenes and Stiefel, 1952) generates a sequence $u_k \in u_0 + \mathcal{K}_k(r_0, A)$ of approximations to the solution $u^\star$ of (4) such that $\| u_k - u^\star \|_A$ is minimized over $u_0 + \mathcal{K}_k(r_0, A)$ and the residuals $r_k = f - Au_k$ are orthogonal to $\mathcal{K}_k(r_0, A)$. Because of these optimality properties, CG is guaranteed to converge in at most N_d iterations in exact arithmetic. Remarkably, these optimality properties can be achieved by using simple, three-term recurrence relations and a small number of auxiliary vectors. Hence, CG was initially seen as a direct method with low storage that could compete with factorization-based direct solution methods. It was soon realized, however, that the key orthogonality relationships could not be maintained with finite precision, and interest in CG waned until the early 1970s. At that time, it was pointed out (Reid, 1971) that CG could still be used effectively as an iterative method. Ideas for accelerating convergence using preconditioning appeared soon afterward (Axelsson, 1972; Concus et al, 1976). See Golub and O'Leary (1989) for an early history, and Freund et al. (1992) and Saad and Van der Vorst (2000) for more complete recent overviews of Krylov subspace methods.

When paired with a suitable preconditioner, CG is a robust and efficient method for solving (4) when A is symmetric positive definite. In multiphysics problems, however, few circumstances are encountered where A is symmetric positive definite. Numerous generalizations of CG to non-symmetric problems that seek to exploit the optimality properties of CG have been proposed. Two principal approaches have been taken. One is to maintain $r_k \perp \mathcal{K}_k(r_0, A)$ through an explicit Gram–Schmidt orthogonalization process. This approach constructs an orthonormal basis of $\mathcal{K}_k(r_0, A)$ and leads to the Arnoldi family of iterative methods, of which the generalized minimum residual (GMRES) method (Saad and Schultz, 1986) is the best-known variant. The main disadvantages of this approach are the storage requirements for the basis of $\mathcal{K}_k(r_0, A)$ and growth in the operation count for incrementally constructing that basis. Practical approaches to dealing with this difficulty amount to setting a maximum dimension of the Krylov subspace. Alternatively, a three-term recurrence relationship can be used. This approach is unable to maintain orthogonality, but it is inexpensive and can often be effective when paired with a suitable preconditioner. This strategy leads to the Lanczos-based family of methods. BiCGStab (Van der Vorst, 1992) is perhaps the most widely used method in this class. For general non-symmetric A, maintaining orthogonality while using short recurrence relationships is not possible (Faber and Manteuffe, 1984). Flexible Krylov variants (see e.g., Saad, 1993; Golub and Ye, 1999; Szyld and Voge, 2001; Vogel, 2007; Chen et al., 2012), where the preconditioner may change across iterations, offer promise for multiphysics scenarios. High-performance implementations of Krylov subspace methods are widely available (Balay et al., 2012; Falgout et al., 2011; Heroux et al., 2005).

Preconditioning. The key observations for Krylov subspace methods are that $\{\mathcal{K}_k(r_0, A)\}_{k=1}^{N_d}$ is a nested sequence of subspaces that approximates the range of A and that the efficiency of a Krylov subspace method is determined by how well (4) is represented by its restriction to a low-dimensional Krylov subspace. Preconditioning is a technique for transforming (4) to an equivalent system where a lower-dimensional Krylov subspace provides a better approximation. This amounts to the introduction of a preconditioner $P \approx A^{-1}$. Typical formulations include left, right, and split preconditioning:

$$\begin{aligned} PAu &= Pf, && \text{(left)} \\ APP^{-1}u &= f, && \text{(right)} \\ P_1 A P_2 P_2^{-1} u &= P_1 f, && \text{(split)} \end{aligned}$$

Observe, in particular, that for left preconditioning, the sequence of approximations $\{u_k\}$ now resides in the Krylov subspaces $\{\mathcal{K}_k(Pr_0, PA)\}_{k=1}^{N_d}$. To illustrate with an extreme case, if $P = A^{-1}$, then $Pr_0 = u^\star - u_0$ (i.e. the initial error), and the Krylov subspace method converges in one iteration. Thus, the art in selecting a preconditioner is managing the trade-off between the cost of applying the preconditioner and the reduction in the number of required iterations.

Many strategies are available for constructing a preconditioner. Early efforts were based on the notion of incomplete factorizations (Meijerink and Van der Vorst, 1977), and much research into black-box, algebraic preconditioning techniques is built around this idea. Using a stationary iterative method as a preconditioner was first explored by Axelsson (1972), and preconditioning with a multigrid method was first explored by Kettler (1982).

Parallel computing introduces new requirements for preconditioners that have a high ratio of computation to communication. A popular preconditioning approach is to use sparse direct solvers or incomplete factorizations in serial on each parallel process, in which each process solves only the portion of the overall problem that resides on its subdomain. Since these methods follow from the natural parallel decomposition of a problem into tasks, they are referred to as *domain decomposition* methods (Smith et al., 1996; Toselli and Widlund, 2005). Such solvers make good reuse of existing serial solver infrastructure, can attain excellent flop performance, work well for problems posed on unstructured meshes, and naturally couple multiphysics processes tightly within each solve.

Consider the full computational domain Ω as being given by a set of p non-overlapping subdomains $\Omega_1, \ldots, \Omega_p$; that is, $\Omega = \cup_{i=1}^{p} \Omega_i$. Then we can partition the unknowns of (4) into two subsets: the first consists of unknowns belonging to a single subdomain, u_I, and the second consists of unknowns that lie on inter-subdomain

boundaries and couple subdomains together, u_Γ. Applying this partition to the matrix A as well results in a block linear system of the form

$$\begin{bmatrix} A_{I,I} & A_{I,\Gamma} \\ A_{\Gamma,I} & A_{\Gamma,\Gamma} \end{bmatrix} \begin{bmatrix} u_I \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_I \\ f_\Gamma \end{bmatrix} \tag{5}$$

where the matrix $A_{I,I}$ is block diagonal. We further define restriction operators R_i that take a full set of unknowns, u, and return only the set of unknowns on the interior of Ω_i for each $i = 1, \ldots, p$.

The simplest domain decomposition method is *block Jacobi*, where all the interprocessor couplings are ignored (that is, Γ is empty) and (5) is solved by using only the decoupled subdomain solves

$$u^{k+1} = u^k + \left(\sum_{i=1}^{p} R_i^T A_i^{-1} R_i \right) (f - Au^k) \tag{6}$$

where A_i is a block of $A_{I,I}$ belonging to Ω_i. A block Gauss–Seidel strategy, which sacrifices a lower degree of parallelism for faster convergence, can also be formulated, provided some coloring of the subdomains $\{\Omega_i\}_{i=1}^p$ is available. For problems with indefinite or singular A_i, artificial Dirichlet, Neumann, or Robin boundary conditions are typically employed at interprocessor interfaces. Because of the resulting decoupling between each subdomain, these methods do not perform well at large scales, especially for problems whose Green's functions have global support, for example elliptic and parabolic PDE problems. As a result, various methods have been proposed to increase the coupling between subdomains and attain improved scaling performance.

Additive Schwarz methods (Dryja and Widlund, 1987; Cai, 1989; Chan and Zou, 1994; Smith et al., 1996) increase coupling by extending each subdomain to overlap with neighboring subdomains, increasing the size of each A_i matrix and redefining each R_i appropriately, then using (6). Extensions of this method include modification of the prolongation matrix R_i^T (Cai and Sarkis, 1999) and inclusion of coarse-grid solves (Bramble et al., 1990). A multiplicative variant, using the same components of the block Jacobi strategy above, is also possible.

Other classes of domain decomposition methods (e.g., *FETI* (Farhat and Rouz, 1991), *BDD* (Mandel, 1993) and *BDDC* (Dohrmann, 2003)) instead increase subdomain coupling by using a non-empty interface set Γ and following a Schur complement-like formulation to solve the full problem

$$u^{k+1} = u^k + \begin{bmatrix} I & -A_{I,I}^{-1} A_{I,\Gamma} \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{I,I}^{-1} & 0 \\ 0 & S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{\Gamma,I} A_{I,I}^{-1} & I \end{bmatrix} (f - Au^k)$$

where $S = A_{\Gamma,\Gamma} - A_{\Gamma,I} A_{I,I}^{-1} A_{I,\Gamma}$ is the Schur complement matrix. Primary differences in methods following this approach lie in their choice of artificial boundary conditions at interior subdomain interfaces and how much of Γ they include, typically choosing only a small subset of the interface problem within S^{-1} at each iteration. Extensive support for constructing domain-decomposed preconditioners is available in high-performance libraries (Balay et al., 2012; Heroux et al., 2005).

3.1.2 Methods for systems of nonlinear equations. Perhaps the simplest and most robust approach for solving the systems of nonlinear equations (3) is the fixed-point iteration, also known as the Picard or nonlinear Richardson iteration. These methods are performed through reformulation of the root-finding problem (3) into a fixed-point problem $u = G(u)$, through definition of a fixed-point iteration function, for example

$$G(u) := u - \alpha F(u) \tag{7}$$

where $\alpha > 0$ is a fixed-point damping parameter that is typically chosen to be less than one. Fixed-point methods then proceed through the iteration

$$u^{k+1} = G(u^k) \tag{8}$$

with the goal that $\| u^{k+1} - u^k \| < \epsilon$. A particular attraction of the fixed-point iteration is that if the iteration function is a contraction, that is, if there exists some $\gamma \in (0, 1)$ such that

$$\| G(u) - G(v) \| \leq \gamma \| u - v \|$$

for all vectors u and v in a closed set containing the fixed-point solution $u^\star$, the fixed-point iteration is guaranteed to converge based on the Banach fixed-point theorem (also called the contraction mapping theorem (Banach, 1922; Granas and Dugundji, 2003)). This convergence is typically linear, however, and can result in slow convergence even when starting with a good initial guess at the solution.

Fixed-point iterations can converge slowly, in particular when the contraction constant γ is close to one. Newton's method (Algorithm 3) offers faster (up to quadratic) convergence. Direct computation of δu in Algorithm 3 may be prohibitively expensive for large-scale problems. *Inexact* Newton methods (Dembo et al., 1982) generalize Algorithm 3 by allowing computation of δu with an iterative method, requiring only that $\| J(u^{k-1})\delta u + F(u^{k-1}) \| < \epsilon_k$ for some carefully selected set of tolerances $\{\epsilon_k\}$; see Eisenstat and Walker (1994, 1996) for theoretical details. Newton–Krylov methods are important variants of inexact Newton methods in which δu is computed with a Krylov subspace method. This choice is advantageous because the only information required by the Krylov subspace method is a method for computing Jacobian–vector products $J(u^k)v$. A key consequence of this reliance on only matrix–vector products is that if these directional derivatives may be approximated, the Jacobian matrix $J(u^k)$ is never itself needed. The JFNK method exploits this approach, using a finite-difference approximation to these products

$$J(u^k)v \approx \frac{F(u^k + \sigma v) - F(u^k)}{\sigma} \tag{9}$$

where σ is a carefully chosen differencing parameter (Brown, 1987; Kelley, 1995) and F is sufficiently

smooth. This important specialization facilitates use of an inexact Newton method by eliminating the need to identify and implement the Jacobian. Of course, efficiency of JFNK depends critically on preconditioning the inner Krylov subspace method, and the need to track changes in the Jacobian places a premium on preconditioners with low setup cost; see Anderson et al. (1999), Gropp et al. (2000), Pernice and Tocci (2001), Chacón and Knoll (2003), Knoll and Keyes (2004), Mousseau (2004), Knoll et al. (2005), Reynolds et al. (2006) and Yeckel et al. (2009). Implementations of inexact Newton methods are available in several high-performance software libraries (Balay et al., 2012; Heroux et al, 2005; Hindmarsh et al., 2005).

Fixed-point and inexact Newton methods can be used to solve multiphysics problems in a fully coupled manner, but they can also be used to implement multiphysics coupling strategies such as Algorithms 1 and 2. With an implicit method for one of the components, one can directly eliminate it in a linear or nonlinear Schur complement formulation. The direct elimination process for u_1 in the first equation of the equilibrium system (1a), given u_2, can be symbolically denoted

$$u_1 = G(u_2)$$

with which the second equation (1b) is well defined as

$$F_2(G(u_2), u_2) = 0$$

and each iteration thereof requires subiterations to solve (in principle to a high tolerance) the first equation. Unless the first system is much smaller or easier than the second (a special case of merit), this is not likely to be an efficient algorithm, but it may have robustness advantages.

If the problem is linear

$$\begin{aligned} F_1(u_1, u_2) &= f_1 - A_{11}u_1 - A_{12}u_2 \\ F_2(u_1, u_2) &= f_2 - A_{21}u_1 - A_{22}u_2 \end{aligned} \tag{10}$$

then $F_2(G(u_2), u_2) = 0$ involves the traditional Schur complement

$$S = A_{22} - A_{21}A_{11}^{-1}A_{12}$$

If the problem is nonlinear and if Newton's method is used in the outer iteration, the Jacobian

$$\frac{dF_2}{du_2} = \frac{\partial F_2}{\partial u_1}\frac{\partial G}{\partial u_2} + \frac{\partial F_2}{\partial u_2}$$

is, to within a sign, the same Schur complement.

Similar procedures can be defined for the evolution problem, which, when each phase is implicit, becomes a modified rootfinding problem on each timestep with the Jacobian augmented with an identity or mass matrix.

If the types of nonlinearities in the two components are different, a better method may be nonlinear Schwarz or ASPIN (Cai and Keyes, 2002). In nonlinear Schwarz, one solves component subproblems (by Newton or any other means) for componentwise corrections

$$\begin{aligned} F_1(u_1^{k-1} + \delta u_1, u_2^{k-1}) &= 0 \\ F_2(u_1^{k-1}, u_2^{k-1} + \delta u_2) &= 0 \end{aligned}$$

and uses these legacy componentwise procedures implicitly to define modified residual functions of the two fields

$$\begin{aligned} G_1(u_1, u_2) &\equiv \delta u_1 \\ G_2(u_1, u_2) &\equiv \delta u_2 \end{aligned}$$

One then solves the modified rootfinding problem

$$\begin{cases} G_1(u_1, u_2) = 0 \\ G_2(u_1, u_2) = 0 \end{cases}$$

by any means. If one uses Algorithm 3 to solve the modified problem, the Jacobian is

$$\begin{bmatrix} \frac{\partial G_1}{\partial u_1} & \frac{\partial G_1}{\partial u_2} \\ \frac{\partial G_2}{\partial u_1} & \frac{\partial G_2}{\partial u_2} \end{bmatrix} \approx \begin{bmatrix} I & \left(\frac{\partial F_1}{\partial u_1}\right)^{-1}\frac{\partial F_1}{\partial u_2} \\ \left(\frac{\partial F_2}{\partial u_2}\right)^{-1}\frac{\partial F_2}{\partial u_1} & I \end{bmatrix} \tag{11}$$

which clearly shows the impact of the non-dimensionalized cross-coupling in the off-diagonals, since the diagonal blocks constitute the identity. (This makes sense since the componentwise rootfinding problems are already solved at this outer level.) In practice, the outer Newton method must converge in a few steps for ASPIN to be worthwhile, since the inner iterations can be expensive. In nonlinear Schwarz, the partitioning of the global unknowns into u_1 and u_2 need not be along purely physical lines, as they would be if they came from a pair of legacy codes. The global variables can be partitioned into overlapping subsets, and the decomposition can be applied recursively to obtain a large number of small Newton problems. The key to the convenience of nonlinear Schwarz is that the outer iteration is Jacobian-free and the inner iterations may involve legacy solvers. (This is also true for JFNK, where the inner preconditioner may involve legacy solvers.)

Unless linear preconditioning of the global system is required to complement what is in effect nonlinear preconditioning through the inner solves, only distributed vector operations are needed to migrate legacy codes requiring coupling to this framework.

Error bounds on linear and nonlinear block Gauss–Seidel solutions of coupled multiphysics problems are derived in Whiteley et al. (2011), wherein it is shown how the norm of the product of the off-diagonal blocks in (11) enters the analysis. Consider the linear case (10) for which the Jacobian of the physics-blocked preconditioned system is

$$\begin{bmatrix} I & A_{11}^{-1}A_{12} \\ A_{22}^{-1}A_{21} & I \end{bmatrix}$$

Let the product of the coupling blocks be defined by the square matrix $C \equiv A_{11}^{-1}A_{12}A_{22}^{-1}A_{21}$, the norm of which may be bounded as $||C|| \leq ||A_{11}^{-1}||\,||A_{12}||\,||A_{22}^{-1}||\,||A_{21}||$. Whiteley et al. (2011) show that, provided $||C|| \leq 1$, any linear functional of the solution (u_1, u_2) of (10) solved by

physics-blocked Gauss–Seidel (Algorithm 1) satisfies conveniently computable bounds in terms of residuals of the individual physics blocks of (10). The cost of evaluating the bounds involves the action of the inverse uniphysics operators A_{11}^{-1} and A_{22}^{-1} on vectors coming from the uniphysics residuals and the dual vectors defining the linear functionals of interest. The extra work required to evaluate the bounds provides confidence that block Gauss–Seidel has been iterated enough to produce sufficiently accurate outputs of interest, be they point values, averages, or fluxes, etc. The required actions of the inverse uniphysics operators are a by-product of an implicit method for each phase. The nonlinear case is similar, except that the Jacobian matrices from each physics phase that comprise C may change on each block Gauss–Seidel iteration.

Design patterns for multiphysics preconditioning. Preconditioning is essential for efficiency when a Krylov subspace method is used. While there has been considerable success in developing black-box, algebraic strategies for preconditioning linear systems (4), preconditioners for multiphysics problems generally need to be designed by hand, using building blocks described above for implementation. A few strategies for doing so are described next. A more complete discussion can be found in Knoll and Keyes (2004).

A simple strategy for constructing a multiphysics preconditioner is to use a Picard linearization to construct an approximate Jacobian for use in a JFNK solution strategy. Another strategy is to mimic operator-splitting strategies to construct an approximate inverse through sequential application of single physics solvers. This can be especially useful when the individual physics components have no knowledge of each other but can solve appropriately posed linear subproblems. Another approach is to use the multiphysics application itself as an approximate solution strategy. In this context, we can think of the outer Krylov or Newton iteration as acting to accelerate and/or stabilize the existing solution method. This idea first appeared in Wigton et al. (1985). The multiphysics application has to satisfy certain properties for this idea to be used; in particular, the ability to calculate a clean, full residual and calculating corrections rather than full updates to solution variables are needed.

A widely used approach is to use a block-diagonal approximation of the Jacobian of the system, which can be effective in many circumstances. This approach ignores the off-diagonal coupling, however, and analysis of the problem can help to identify the strongly coupled physics. Improved performance can generally be achieved by capturing the strong couplings with the preconditioner and leaving the weak couplings to be resolved by the outer Krylov/Newton iterations. Such approaches generally lead to identification of a Schur complement that embodies the important coupling, and their success relies on judicious approximation of the Schur complement (Chacón et al., 2002; Elman et al., 2003).

3.2 Continuum–continuum coupling

Multiphysics problems require the computation of interacting fields that may feature different scales in time and/or space and may additionally be defined over different spatial regions. Mathematical modeling of such complex systems can lead to large systems of PDEs, possibly with different structures on different domains. Roughly speaking, two approaches are used. In a single-domain approach, the coupled system is considered globally, and the state variables are defined over the entire spatial domain of the problem; a simple example is a thermomechanical system whose solution consists of the temperature and displacements. Alternatively, different sets of state variables and/or models may be considered on different spatial or physics domains. A simple example for different spatial domains is FSI, where the fluid velocity and pressure are defined in the fluid domain and displacements are defined in the solid domain.

In general, single-domain approaches can be conceptually simple and straightforward. However, single-domain models may lack flexibility; solution of the single domain problem may be expensive; and, more importantly, generalization to more complicated models may not be possible. For example, no single-domain approaches exist for coupling multiphase, multicomponent flow in porous media with multicomponent free flow, whereas transfer conditions based on mechanical, chemical, and thermal equilibrium can be derived for a model posed on two domains (Mosthaf et al. 2011). On the other hand, the divide-and-conquer approach of multidomain models allows the use of different numerical strategies tuned to the physics of each component. The price of such flexibility is the need to formulate suitable transfer conditions, in both model formulation and model discretization, which guarantee equilibrium and well-posedness of the coupled system, and to devise efficient solution techniques that are robust, stable, and accurate.

Multidomain approaches require coupling of continuum models in space and, for time-dependent problems, in time. This coupling must respect the mathematical formulation of the coupled problem and must account for different meshes and discretizations used in each domain. Coupling components that are allowed to proceed at their own timescales can be particularly problematic. The following subsections describe techniques currently being used in several selected multiphysics applications. We note that, currently, highly scalable algorithms for these operations are not readily available.

3.2.1 Methods for coupling multiphysics components in space. Multiphysics applications require some means to exchange information between physics components. Preserving important properties, such as conservation or order of accuracy, is a principal concern. In the simplest case, all the physics components are defined on the same physical domain Ω and use the same mesh Ω^h. In this situation, the coupling is defined by the physical model, and no error is

incurred in transferring data between meshes. Numerically, some additional interpolations may be needed to align variables with different centerings or to achieve some consistency between different discretization schemes. In addition, the resolution must be fine enough to meet the requirements of all the physics components that share the mesh. More often, each physics component uses a mesh that is tailored to best represent the physics. These meshes can either fully or partially overlap and can also introduce different discrete representations of the geometry of Ω. (A similar need is encountered for arbitrary Lagrangian-Eulerian (ALE) methods, where periodic mesh remapping is needed.) Moreover, different physics components can be defined on distinct physical domains Ω_i (where i indexes the physics components) and are coupled across lower-dimensional interfaces $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j,\ i \neq j$, between the domains, which may also evolve in time. In this case, the meshes $\Omega_i^{h_i}$ and $\Omega_j^{h_j}$ may or may not match along the interface. When the restrictions $\Gamma_{ij}^{h_i}$ and $\Gamma_{ij}^{h_j}$ of the meshes to the common interface Γ_{ij} do not match, information transfer across the interface must account for these differences in the discrete representation of Γ_{ij}. In some cases the picture can be further complicated when Γ_{ij} is determined as part of the calculation. The rest of this section provides a brief overview of methods currently used by multiphysics applications, and Section 4.2.3 considers related software issues.

At its core, transferring discrete fields between different meshes is a form of interpolation that is determined by how the field is represented on the source and target meshes. Search algorithms that determine the support of the interpolation on the source mesh are critical for constructing efficient transfer operations. Tree structures such as octrees, alternating digital trees, and k-d trees are used when the points to be evaluated are randomly distributed. Mesh topological adjacencies are used when the points are ordered such that the next point is from a nearby element. When the geometries of the discrete domains do not match, a destination point may lie outside the discrete source domain. In this case extrapolation can be used; ideally, information about the physical domain and boundary conditions will inform the extrapolation process.

In more complex cases, the transfer process must satisfy various kinds of local and global constraints. Examples include matching the integral of the solution and guaranteeing a divergence-free property (Bochev and Shashkov, 2005). In many circumstances, local conservation constraints are imposed. For a mesh transfer $\mathcal{T}$ to be conservative in a region $\mathcal{R} \subset \Omega$, it must satisfy

$$\int_{\mathcal{R}} \mathcal{T}f = \int_{\mathcal{R}} f$$

where f is the source grid data. Enforcement in a discrete sense can be represented as

$$\int_{\mathcal{R}} \phi_i \mathcal{T}f = \int_{\mathcal{R}} \phi_i f$$

where $\{\phi_i\}$ is a set of test functions. For interpolation, they are simply delta functions at points on the destination mesh. If f is represented in a Galerkin formulation, $\{\phi_i\}$ is the basis that is used on the destination mesh. Because f and ϕ_i are only piecewise smooth on source or target meshes, the product $\phi_i f$ is in general smooth only within the intersection of source and target elements. Therefore, using quadrature rules over source or target elements to compute $\int \phi_i f dx$ may violate the regularity assumption of quadrature rules, and large errors and even instabilities can result from computing these integrals using quadrature rules based only on the source or destination mesh (Jiao and Heath, 2004a; Jaiman et al., 2011b). Knowledge of the complete intersection of the source and target meshes can then be used together with appropriate quadrature rules to construct $\mathcal{T}$ (Grandy, 1999; Jiao and Heath, 2004a,b,c; Solin et al., 2010; Jaiman et al., 2011a).

Transferring information across a shared interface between two domains depends on the smoothness of solutions on either side of the interface, discretization schemes, and alignment of the two meshes on either side of the interface as well. When the meshes align at the interface, direct transfer can be used effectively. For an illustration, consider coupling Navier–Stokes flow on the surface and Darcy flow in subsurface porous media (Figure 12) (Chidyagwai and Rivière, 2010). The free-flow region is denoted Ω_1, and the porous medium is represented as several layers (regions A, B, C) with different permeability and faults (region D). The interface conditions enforce continuity of the normal component of the velocity across the interface, a force balance at the interface, and the Beavers–Joseph–Saffman model relating the tangential free-flow velocity to the shear stress at the interface (Girault and Riviére, 2009). Let a_{NS} represent the bilinear form for the Navier–Stokes flow discretized with a continuous finite-element method, and a_D represent the bilinear form for the Darcy equations discretized with a discontinuous Galerkin method to accurately capture irregularities in the subsurface region. A grid is used that covers the entire domain but captures the interface between the two models. This means that the bilinear form a_Γ for the interface coupling conditions across the interface can easily be constructed by simply using values of the solution obtained at the interface. The fully coupled problem is then given by

$$a_{NS}(u, v) + a_D(u, v) + a_\Gamma(u, v) = L(v) \tag{12}$$

where v is drawn from suitable test spaces and L is a forcing term.

A fully coupled solution method is the most stable and accurate approach to solving (12). However, this leads to a large system of nonlinear equations that may be difficult to precondition and expensive to solve. One way to lower the cost of solving (12) is to use a two-grid technique (Chidyagwai and Rivière, 2011). In the first step, (12) is solved on a coarse mesh with resolution H; in the second

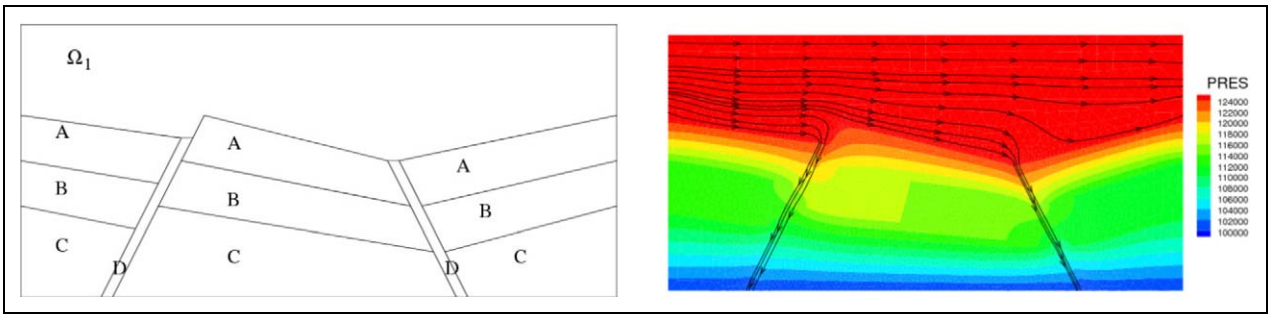


Figure 12. Computational domain (left) and pressure and velocity contours (right) for coupling of Navier–Stokes flow on the surface and Darcy flow in subsurface porous media.

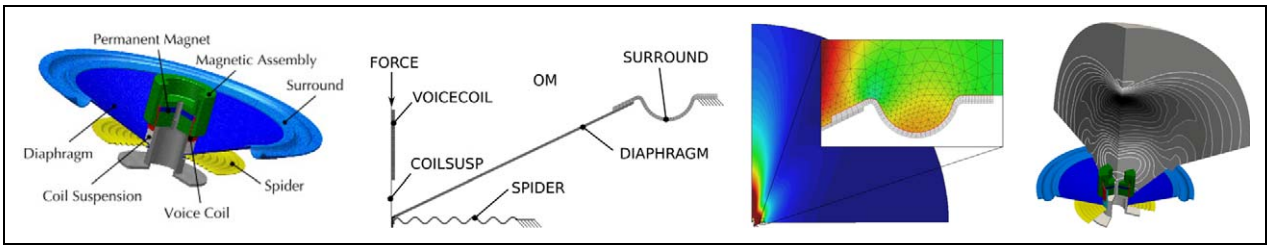


Figure 13. Schematic view of the geometric and acoustic pressure field in acoustic-structure interaction for a three-dimensional loudspeaker geometry.

step, decoupled "single physics" problems are solved on a finer mesh, with resolution $h = H^2$. Boundary data along the interface for the decoupled problems is obtained from the coupled coarse mesh solution. This approach is nearly an order of magnitude faster and produces solutions with accuracy comparable to the fully coupled approach, but it is less stable and accurate. The solution obtained with this algorithm (Figure 12) shows smooth velocity contours, even into the fault regions. For details, see Chidyagwai and Rivière (2011).

The mortar finite-element method is another approach for treating interface conditions (Bernardi et al., 1993, 1994). It is especially effective when the meshes on the two sides of the interface do not match and when some quantity (such as flux or stress) is needed at the interface. The mortar finite-element method yields a variationally consistent formulation and provides an abstract framework for analyzing the weakly coupled discrete model. The algebraic problem can then be stated as a saddle point or, equivalently, as a constrained system where static condensation of the Lagrange multiplier is realized and the coupling conditions are imposed on the spaces. Mortar formulations are widely used in the context of nonlinear contact problems in structural mechanics and replace simple node-to-node or node-to-segment penalty approaches; see, for example, Wohlmuth (2011). The surface traction plays the role of the Lagrange multiplier, which has to satisfy a sign condition in the normal direction, which is due to the non-penetration condition, and the friction law, which combines normal and tangential surface forces. The inequality conditions can be reformulated as a weakly consistent saddle point system in terms of a nonlinear complementarity function. Then semi-smooth Newton methods provide a superlinear solver, and an efficient algorithmic implementation can be based on the choice of biorthogonal basis functions, resulting in a diagonal mass matrix.

Figure 13 illustrates the acoustic-structure interaction for a three-dimensional loudspeaker geometry; see Flemisch et al. (2010). The problem has been formulated in a weakly consistent mortar setting, and highly non-matching meshes are applied for the numerical simulation.

Of crucial importance for the stability of the method is a uniform inf–sup condition for the pairing between dual and primal variables. In order to obtain optimal a priori error results, the discrete Lagrange multiplier space must additionally satisfy a best-approximation property. With these two ingredients, one can easily construct locally defined higher-order methods. Moreover, the influence of curvilinear interfaces can be analyzed within the framework of variational crimes, and the weak coupling concept can be easily combined with blending or isogeometric finite elements.

Although mortar techniques were originally designed for non-overlapping domain decomposition methods, the idea of weakly coupled subdomain formulations can be used to design two-scale overlapping subdomain approaches. Globally, a simple, coarse-scale discrete model is used, whereas a more sophisticated model is applied locally on a small patch. Different equations, mesh sizes, and geometries can be considered. Here, one has to distinguish between one- and bidirectional couplings. One-directional coupling can be

regarded as a postprocess and allows locally for a higher resolution of the numerical solution. The possibly nonlinear system has the structure of a lower triangular block system and thus can easily be solved in terms of a Gauss–Seidel-type solver, possibly in combination with massive parallel algebraic multigrid or domain decomposition techniques. For bidirectional coupling, iteration between the small-patch and coarse-scale models is required to obtain a physically correct solution.

3.2.2 Methods for coupling multiphysics components in time. Coupling time-dependent continuum models depends on the temporal resolution requirements of the physics components. Typically, each component of the solution is allowed to evolve governed by its own physics. However, there can be wide disparity in the timescales of the constituent physics components, which may in turn be quite different from the timescale of the problem being solved. Replacing instantaneous interactions between different physics with discrete interactions can introduce subtle stability and accuracy issues. Likewise, the multiphysics problem may conserve some global quantity (such as energy), but this property may not be preserved by the simulation.

In evolutionary problems (2a)–(2b), the spatial discretization leads to a system of ordinary differential equations (ODEs) or differential algebraic equations (DAEs) in general; that is, ODEs possibly with algebraic constraints. A timestepping approach is then used to advance the solution in time. This strategy is called the *method of lines*. For exposition brevity, we initially focus on ODEs, with initial conditions $u(t_0)$ and computing values of the solution $\{u(t_n)\}_{n=1}^{N_t}$ at discrete times $t_1 < \ldots < t_{N_t}$.

In an *explicit method*, future states are given by a formula involving known computed quantities that can be evaluated directly. The simplest example is forward Euler:

$$u(t_{n+1}) = u(t_n) + \Delta t f(u(t_n))$$

where $\Delta t = t_{n+1} - t_n$. Explicit methods are inexpensive per iteration because they require a fixed number of function evaluations to complete the step; however, they typically have severe stability restrictions and, therefore, are suited only for non-stiff problems.

A *fully implicit method* advances the solution in time by using current information and an inversion process with no explicit steps. The simplest example is backward Euler:

$$u(t_{n+1}) = u(t_n) + \Delta t f(u(t_{n+1}))$$

Fully implicit methods are relatively expensive because of the need to solve (non)linear systems at each step; however, they have favorable stability properties (Shampine and Gear, 1979; Higham and Trefethen, 1993). Inclusion of explicit elements gives rise to semi-implicit methods (Zhong, 1996; Giraldo, 2005).

Spatial discretization of multiphysics problems (2a)–(2b) may result in problems in which the tendency $f(u)$ has components with widely different dynamics, often classified as stiff and non-stiff (Ascher et al., 1997; Verwer and Sommeijer, 2004; Shampine et al., 2006; Hundsdorfer and Ruuth, 2007; Constantinescu and Sandu, 2010b; Cionnors and Miloua, 2011). This case, called *additive splitting*, generally corresponds to problems in which the physics components act in the same spatial domain, such as reactive transport or RHD (Ruuth, 1995; Giraldo et al., 2003; Estep et al., 2008a; Giraldo and Restelli, 2009; Giraldo et al., 2010; Durran and Blossey, 2012). When a partitioned time-integration method is used with physics components defined on the same spatial region, some work must be done to synchronize the timestepping in each physics component. When one component has a much faster timescale relative to another, a simple strategy is to use an integer multiple of the faster timescale for the slow timescale. In *component partitioning* some solution components correspond to a stiff subproblem, and the others, to a non-stiff one; examples are FSI and flow with different speeds (Constantinescu and Sandu, 2007; Constantinescu et al., 2008; Sandu and Constantinescu, 2009). Component partitioning can be used, for instance, if physics components are defined on different spatial domains; the same subcycling idea can work, but some protocol for communicating information across the interface is needed to ensure accuracy and stability. This leads to the following two conceptual types of partitioning:

$$f(u) = f_1(u) + f_2(u) \qquad \text{additive partitioning} \qquad (13)$$

$$f(u) = \begin{bmatrix} f_1(u_1, u_2) \\ f_2(u_1, u_2) \end{bmatrix} \qquad \text{component partitioning} \qquad (14)$$

Simple transformations can be used to cast one type of splitting into the other (Ascher et al., 1997; Constantinescu and Sandu, 2010b). We will henceforth use form (13) in which f_1 is relatively non-stiff and its presence points one to use an explicit integrator for efficiency, whereas f_2 is relatively stiff and requires an implicit integrator for stability reasons. For instance, in (13), f_1 can incorporate all slow transport components and f_2 can incorporate all stiff kinetic terms, and with (14), u_1 may represent coarse-grid points and u_2, finely refined states or structure and fluid states, respectively. Partitioned integrators were introduced to solve such problems efficiently.

Partitioned methods use time integrators with different properties or timesteps for each (physics) component. They allow black-box implementation of the physics components and different timesteps in each physics component. Nevertheless, this formalism allows one to rigorously analyze the stability and accuracy of the compound scheme. Partitioned methods can be tightly coupled (in time); one simple example is the forward–backward Euler method:

$$u(t_{n+1}) = u(t_n) + \Delta t f_1(u(t_n)) + \Delta t f_2(u(t_{n+1}))$$

Strang splitting methods (Strang, 1968) are also particular cases of (loosely coupled) partitioned schemes:

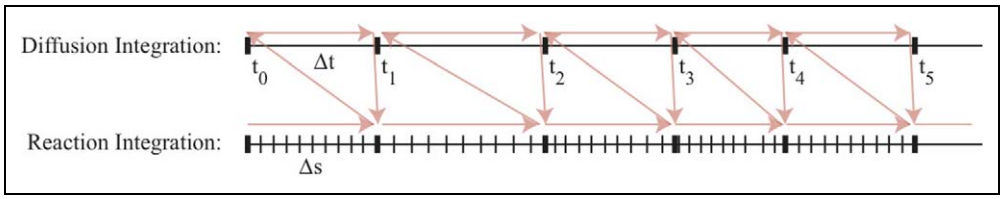


Figure 14. Illustration of multirate operator-splitting time discretization.

$$u_\star = u(t_n) + \frac{\Delta t}{2} f_2(u_\star)$$
$$u_{\star\star} = u_\star + \Delta t f_1(u_\star)$$
$$u(t_{n+1}) = u_{\star\star} + \frac{\Delta t}{2} f_2(u_{\star\star})$$

The caveats of such a strategy include order reduction and low accuracy as illustrated in Knoll et al. (2003). Also, reduced order or even loss of consistency may occur at boundaries for additive splitting methods. One solution to restore consistency in this case is to apply *intermediate boundary conditions* that are specific to each substep in the method (LeVeque and Oliger, 1983; Connors et al., 2012). The temporal splitting error can be fairly easily avoided in tightly coupled schemes (Kennedy and Carpenter, 2003; Constantinescu and Sandu, 2010b). Other tightly coupled strategies involve embedding explicit evaluations inside an implicit solver (Kadioglu et al., 2010b).

We differentiate two cases that have a more practical flavor: multirate and IMEX. Multirate methods are partitioned methods that use the same scheme in each partition but with different timesteps (Gear and Wells, 1984; Bartel and Günther, 2002; Logg, 2003a,b, 2004, 2006; Constantinescu and Sandu, 2007; Savcenco et al., 2007; Constantinescu et al., 2008; Sandu and Constantinescu, 2009); this goes back to subcycling. Multirate schemes are also used to address accuracy considerations. One instance is applications that employ temporal and spatial adaptive grids. Another instance is shown in Figure 14, which illustrates a multirate scheme that uses smaller timesteps for the reaction component in a reaction–diffusion problem.

An IMEX method is a partitioned method that uses an implicit integrator on one partition and an explicit integrator on the other. Therefore, an optimal trade-off between efficiency and stability becomes possible (Crouzeix, 1980; Kennedy and Carpenter, 2003; Constantinescu and Sandu, 2010b; Filbet and Jin, 2010). Moreover, it is generally desirable to use an explicit integrator when nonlinear or non-differentiable spatial discretizations are used, such as shock resolving schemes (LeVeque, 2002). In this case IMEX alleviates the difficulties posed by such a problem on the nonlinear solver, which is applied only on the stiff component. Furthermore, IMEX methods generalize fully implicit $[f_1 \equiv 0, f \to f_2]$, and fully explicit $[f_2 \equiv 0, f \to f_1]$ methods, and can provide the best balance between accuracy and efficiency. In both cases the temporal splitting error can be made of the same order as the schemes used separately; however, stiffness leakage can be a concern (Kennedy and Carpenter, 2003). Partitioned methods based on the classical linear multistep (Ascher et al., 1995; Ascher and Petzold, 1998; Hundsdorfer and Ruuth, 2007), Runge-Kutta (Ascher et al., 1997; Kennedy and Carpenter, 2003; Verwer and Sommeijer, 2004; Constantinescu and Sandu, 2007) and extrapolation methods (Gragg and Stetter, 1964; Constantinescu and Sandu, 2010b) have been proposed and successfully applied in practice (see e.g., Butcher, 2008). In Figure 15 we illustrate the temporal convergence rate for a two-dimensional Euler rising-bubble problem by using high accuracy in space (Giraldo and Restelli, 2009) and different IMEX time-integration schemes. The splitting is based on linearizing the PDE: extracting the stiff wave components, L, and then integrating (2a)–(2b) as

$$\partial_t u = (f(u) - Lu) + Lu$$

where the first term is treated explicitly and the last linear term, implicitly. The efficiency advantage of IMEX schemes increases with temporal discretization order. Given a timestep selected for accuracy, in terms where the timestep is below the stability limit, explicit integration is employed for efficiency, and in terms where the stability is limiting, implicit integration is employed.

Interest in IMEX schemes is far from limited to multiphysics problems. An early use of IMEX temporal integration, with consideration of stability and numerical efficiency, was in solid mechanics and heat transfer applications (Hughes and Liu, 1978b,a). A common source of localized stiffness apart from multiphysics problems is stiffness induced by adaptive meshes, for example in boundary layers in fluid mechanics or electromagnetism. In Kanevsky et al. (2007), IMEX methods are introduced for this purpose, and implicitness is employed in the diagonal terms only of the highly refined portions of the domain, in pursuit of further efficiency. Truly multiphysics applications of IMEX methods now abound. An algorithmic variant that exploits the JFNK method for the implicit part and fully eliminates splitting error is employed by Kadioglu et al. (2010a,b) in RHD problems. In these studies, a second-order IMEX technique resolves the explicitly treated non-stiff hydrodynamics subsystem inside every nonlinearly implicit Newton step of the radiation subsystem. This causes each Newton correction of the implicit block to be felt immediately by the unknowns governed by the explicit block; however, there is no additional software or data structure complexity for the implicit solver.

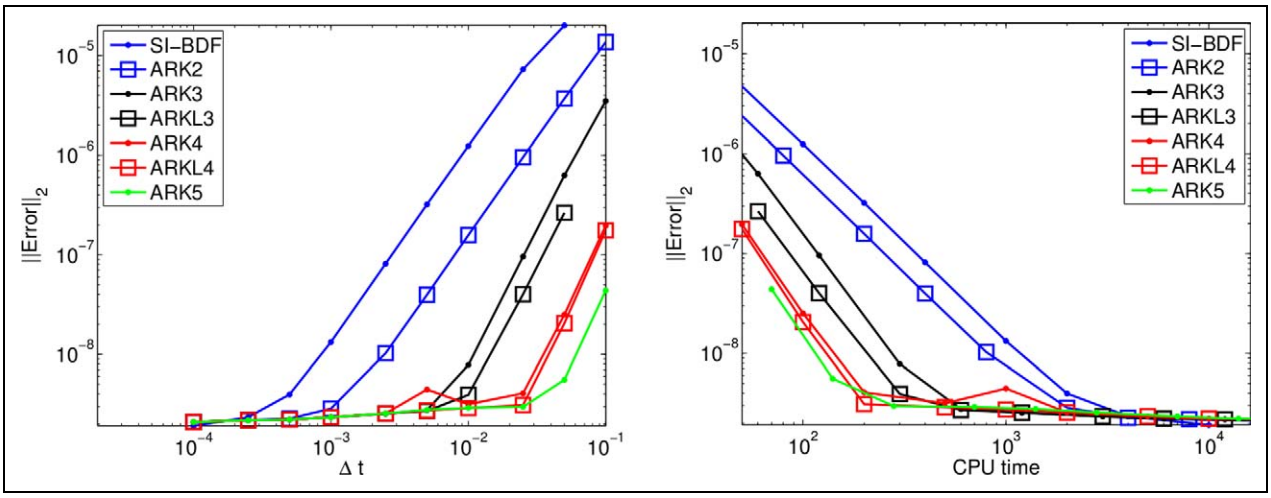


Figure 15. Global convergence and timing of implicit–explicit (IMEX) schemes for a two-dimensional Euler rising-bubble test problem. Method SI-BDF is second order, and the rest have the order suffixed to their name.

Related to the self-consistent IMEX method enabled by the use of JFNK for the outer implicit part is the concept of "implicit balancing" introduced in Knoll et al. (2003), in which in multirate, multicomponent problems, all nonlinear couplings between the components are evaluated at the same point in time, rather than just semi-implicitly. Semi-implicitness is sufficient to preserve stability, but its accuracy properties are limiting. As illustrated for radiation diffusion, MHD, and shallow-water models, implicit balancing permits second-order schemes to attain true second-order temporal accuracy, as opposed to first-order accuracy with semi-implicitness only.

Stability considerations for partitioned methods. Partitioned methods use multiple time integrators with well-known stability properties when applied to the compound problem (2a)–(2b). When applied on partitions, however, the stability properties of individual methods do not transfer directly to the stability of the compound problem. Consider two integrators that are stable when applied separately on each partition but where the resulting compound method may lead to unstable overall computations (Frank et al., 1997; Ropp et al., 2004; Ropp and Shadid, 2009; Constantinescu and Sandu, 2010b). Such a situation is illustrated in Figure 16 for extrapolated IMEX schemes, where the dependence between the implicit and explicit stability regions is clearly visible. Note that larger stability regions for the implicit method correspond to smaller stability regions for the extrapolated explicit methods. This behavior must be taken into account when selecting an extrapolated IMEX scheme for multiphysics problems. If the implicit term is used to integrate symmetric discretizations of hyperbolic operators that lead to pure imaginary eigenvalues, an A-stable (see e.g., Lambert, 1991) implicit method is needed, and one may expect a penalty on the explicit step size because the explicit stability region is smaller (Method A). Discretizations of parabolic operators that lead to negative real eigenvalues can use an $A(\alpha)$-stable implicit method for $\alpha < 90$, so the larger stability region for the explicit operator allows larger stable explicit steps (Method B). These examples tacitly assume that the implicit and explicit operators are simultaneously diagonalizable, which happens almost exclusively only for scalar problems; however, in practice, this analysis typically holds for vector-valued problems, especially when stiff components are present (Frank et al., 1997).

Stability properties of partitioned timestepping methods are also affected by the treatment of interfaces. This can be illustrated by a simple model of heat conduction through two adjacent materials in subdomains Ω_1 and Ω_2 that are coupled across their shared and rigid interface Γ through a jump condition:

$$\begin{cases} \frac{\partial u_i}{\partial t} - \nu_i \Delta u_i = f_i, & x \in \Omega_i \\ -\nu_i \nabla u_i \cdot \mathrm{n}_i = \kappa(u_i - u_j), & x \in \Gamma \\ u_i(x,0) = u_i^0(x), & x \in \Omega_i \\ u_i = g_i, & x \in \partial\Omega_i \backslash \Gamma \end{cases} \tag{15}$$

for $i,j = 1,2,\ i \neq j$. Three time-integration strategies can readily be described. The fully coupled approach solves equation (15) with u_1 and u_2 and all terms evaluated at the same time level. This method is unconditionally stable, conserves energy, and converges optimally under standard smoothness assumptions. A second scheme is to again keep u_1 and u_2 at the same time level but allow the transmission conditions on Γ to lag in time. This IMEX scheme decouples the subdomain solves but is stable only when κ is small compared with ν_1 and ν_2, together with a CFL-like stability restriction on the timestep. Here the trade-off is between the need for smaller timesteps and the reduced cost of solving easier problems in each subdomain. A third partitioned scheme uses interface data at the previous timestep from the other subdomain. Since each subdomain requires data from the other subdomain, this data-passing approach is less flexible but is stable when κ is large compared with ν_1 and ν_2. The data-passing approach also does not

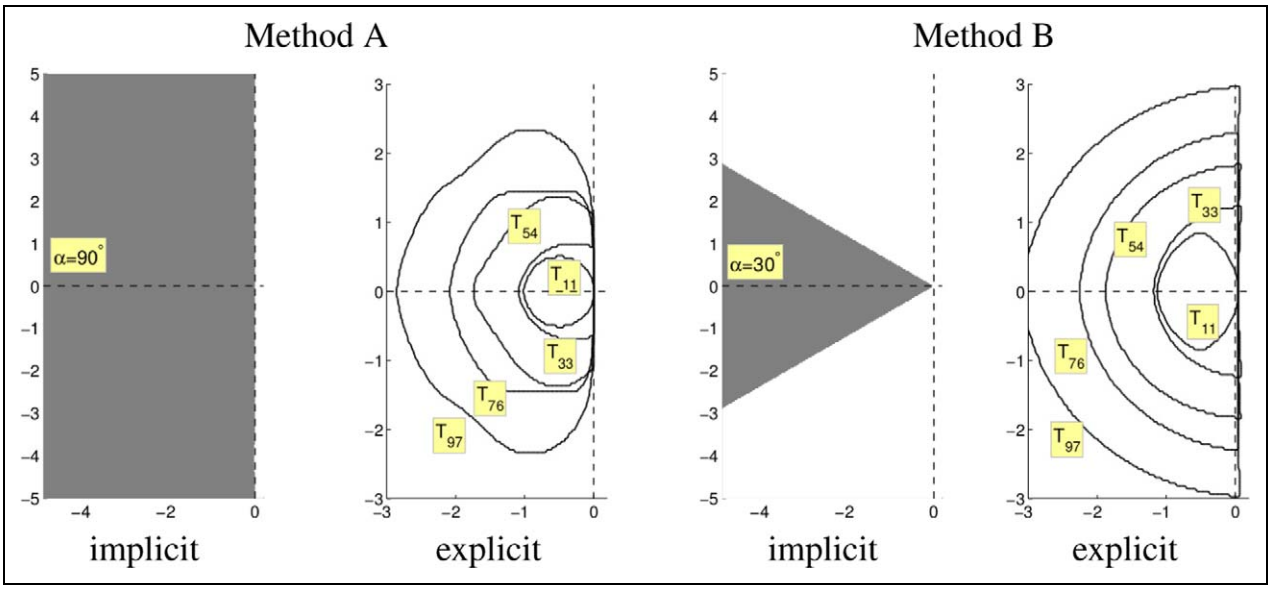


Figure 16. Stability regions of implicit–explicit (IMEX) pairs for extrapolated IMEX schemes. The implicit methods are $A(\alpha)$-stable for different values of α. The explicit stability regions indicate different orders of extrapolation. The stability region of the IMEX scheme contains the cross-product of the regions for these pairs. This figure is adapted from Constantinescu and Sandu (2010a); 2010 Society for Industrial and Applied Mathematics. Reprinted with permission. All rights reserved.

have a stability constraint on the timestep; see Connors et al. (2009) for details and analysis. Partitioned time-integration methods are often used in a sequential fashion, as in Algorithm 2, first advancing a single physics component in time, followed by updating information on an interface, advancing a second physics component in time, and finally updating interface information from the second component. The strategy is often referred to as staggered computation, and the techniques used to update data on the interface play a crucial role in the stability of this approach, irrespective of the time-integration methods used for the component physics. In particular, there can be a time lag in the interface data that leads to an imbalance in the interface transmission conditions, even for strategies that are formally second-order accurate in time. For example, in FSI, a small error in the structural displacements may result in large errors in the fluid pressure. The stability and accuracy can be improved by using the combined interface boundary condition (CIBC) (Jaiman et al., 2011b), which solves the transformed conditions for the interface quantities based on their space derivatives and time derivatives. The CIBC technique allows for the construction of a staggered coupling procedure with a similar order of accuracy and stability as tightly coupled computations for some classes of fluid–structure coupling problems.

Despite the issues associated with partitioned and mixed approaches to multiphysics time integration, these elements can be combined to produce stable, accurate, and efficient simulations. Such an approach is illustrated by hierarchical adaptive mesh refinement (AMR), which we discuss in the context of lean, premixed turbulent combustion (Day and Bell, 2000). This is a demanding multiscale simulation involving fluid flow, chemical reactions, and heat transfer to resolve spatial scales that span two orders of magnitude and timescales that span three orders of magnitude. A finite-volume formulation is used to enforce crucial conservation properties. The AMR strategy is tailored to the mathematical structure of the problem to exploit the relationship between the scales and physical processes in the problem. The fluid equations are advanced in time by using a fractional-step projection formulation, whose elements include explicit advection (via a Godunov scheme), implicit diffusion (via a Crank–Nicolson predictor–corrector scheme), Strang splitting for the chemical reactions, and two discrete projection steps to enforce a modified, divergence-free constraint.

AMR introduces two complexities. The first is the need to compute discrete differential operators where the resolution changes; this computation is done through a simple direct interpolation of data at a coarse resolution to a finer resolution. Local piecewise quadratic interpolation of coarse data is used to align the coarse data with fine data and to calculate fine-scale fluxes at the interface (Almgren et al., 1998). (Local piecewise linear interpolation can also be used (Ewing et al., 1991). Failure to adequately address this issue can lead to non-physical artifacts in the solution (Choi et al., 2004).) In addition, a re-fluxing procedure (Berger and Colella, 1989) is employed to enforce conservation and free-stream preservation at these changes in resolution. The second complexity is local subcycling in time: a global timestep on the entire locally refined grid can be inefficient and will lead to a wide range of CFL numbers across the grid, compromising the accuracy of the Godunov advection scheme. In order to handle this, timesteps are chosen in each refinement region to maintain a CFL number close to one. Practically speaking, this means that a grid

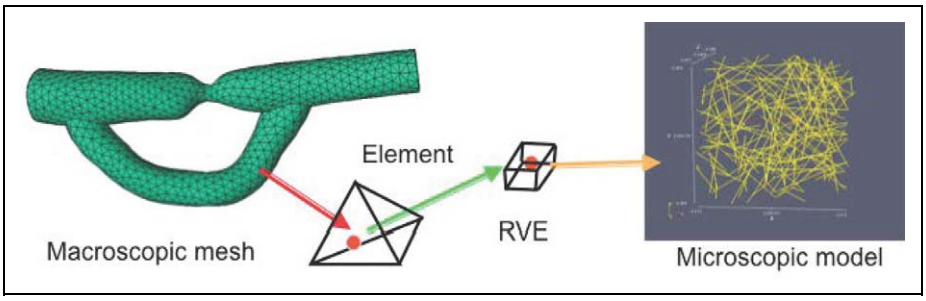


Figure 17. Two-scale model for soft, fibrous material. This diagram indicates the transition from the element level of a macroscopic mesh to a representative volume element (RVE) and then to a microscopic model (Luo et al., 2009).

with resolution h requires twice as many timesteps as the next-coarser region with resolution $2h$ (assuming the refinement ratio is 2). Overall, this AMR scheme is second-order accurate in time and space; see Pember et al. (1998) and Almgren et al. (1998) for details.

We note that it is difficult to devise partitioned time-integration schemes that are better than second-order accurate in time. Spectral deferred correction (SDC) (Frank and Ueberhuber, 1997; Dutt et al., 2000) is one potential way to improve time accuracy. Starting with a provisional solution obtained with a simple, low-order method, SDC generates a sequence of corrections that can lead to higher-order accuracy in the solution. Specifically, one writes the solution to the problem in terms of a Picard integral equation and evaluates the integral using interpolated values of the provisional solution. Spectral accuracy of the integral can be obtained by using Gaussian quadrature nodes in the quadrature. In principle, the order of accuracy can be increased up to the order of accuracy of the quadrature used to evaluate the Picard integral. Variations of SDC have also been developed for semi-implicit time-integration methods (Minion, 2003, 2004).

3.3 Continuum–discrete coupling

The development of methods to couple models across multiple physical scales is a fairly new area of investigation that has grown rapidly in the past decade. Much of the initial multiscale-methods development has focused on the linking of distinctly different spatial scales, in which the coarse scale is represented by a PDE model and the fine scale by either a discrete model (e.g., atomistic) or a fine-scale heterogeneous continuum model (e.g., composite materials in which the microstructural constituents are modeled). A key feature of these problems is the existence of a strong scale separation that greatly influences the method developed. The majority of the scale-linking methods fit into one of two classes: hierarchic (also called information-passing) or concurrent methods (Fish, 2010).

In hierarchic methods information is "upscaled" from the fine-scale model to the coarse scale to provide model information (e.g., constitutive relations) at a point in the coarse scale. Figure 17 graphically depicts this process for the case where a finite-element mesh is used to discretize the macroscale continuum equations. As part of the construction of the element stiffness matrix, the "force/deformation" relationship at the numerical integration points in the elements needs to be evaluated. In this two-scale model, the "force/deformation" relationship is determined through the analysis of a model defined on a representative volume element (RVE) composed of one-dimensional fiber elements (Luo et al., 2009). There is a long history of development of homogenization methods (Benssousan et al., 1978; Sanchez-Palencia, 1980) to link continuum models between two scales. Recently these methods have been generalized to link from atomistic to continuum (Li et al., 2008).

In concurrent methods the different models exist concurrently over different portions of the domain. Concurrent methods are frequently used for problems where an atomistic representation is needed in some small fraction of the domain, while the majority of the model can use a much coarser representation. Among the most mature of these methods is the quasicontinuum method (Tadmor et al., 1996; Knap and Ortiz, 2001; Miller and Tadmor, 2002) in which atomistic mechanics are applied in critical regions. In non-critical regions entire groups of atoms are approximated in terms of a small number of representative atoms with specific assumptions applied on the variation of fields over the group of atoms in much the same way as a field is approximated in terms of shape functions over a mesh of elements. Other concurrent methods employ a continuum PDE for the coarsest scale (Belytschko and Xiao, 2003; Wagner and Liu, 2003; Xiao and Belystschko, 2004; Fish et al., 2007). The key component of these methods is the linkage of the atomistic to the continuum. Methods that attempt to link through just the interface between the portions of the domain with different models have not produced good results. Therefore, methods are needed that employ some form of overlap region in which both the continuum and atomistic models exist and are blended from one model to the other. Figure 18 shows a simple graphic of this idea, with a strictly atomistic region to the right; an overlap region in the middle where both atomistic and continuum models interact; and a continuum region to the left that, in this example, includes some dummy atoms that follow the continuum and support finalization of the transition from atomistic to continuum. A key function of the overlap region is supporting the transition of atomistic

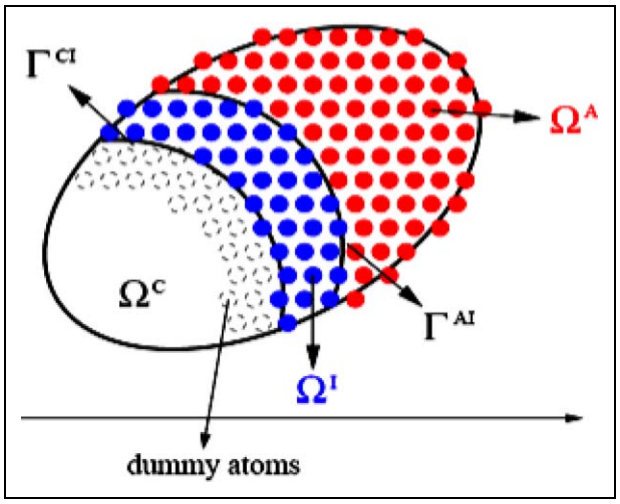


Figure 18. Atomistic-to-continuum concurrent method with overlap region.

quantities to continuum quantities, which must be done with care in the zero temperature case and has proved difficult in the finite-temperature case since a method is needed to determine what portion of the energy content goes into thermal loads and what goes into mechanical loads.

In many problems the areas in which the fine-scale models are needed are not known a priori. To deal with such situations, adaptive multiscale methods are being developed (Oden et al., 2006; Nuggehally et al., 2007; Bauman et al., 2008). Figure 19 shows the use of an adaptive concurrent atomistic–continuum method to study the growth and interactions of dislocations around nanovoids (Fish et al., 2007; Nuggehally et al., 2007).

An alternative scale-linking approach is the heterogeneous multiscale method (E et al., 2007), which defines compression operators to relate discrete-to-continuum scales and reconstruction operators to relate continuum-to-discrete scales. In addition to dealing with the complexities of linking the 10 orders of magnitude of spatial scales when linking atomistic to continuum, the timescales that must be bridged can be as high as 15 orders of magnitude. The same methods used in multiphysics time-coupling are considered in multiscale time-coupling. Moreover, new methods such as the equation-free multiscale method (Kevrekidis et al., 2003) have been developed that link statistically averaged fine time- and space-scale realizations to the coarse time and space scales.

3.4 Error estimation

Reliable error estimation for multiphysics problems is difficult. Because a priori bounds must account for the largest possible error across a large class of solutions, they do not yield sharp estimates and are useful primarily for characterizing the general behavior of the error, such as order of accuracy. In addition, most a priori estimates are asymptotic and are useful primarily for relatively well-resolved problems having smooth solutions. In this regime, a priori bounds can be useful for constructing error estimates based on extrapolation. For example, the AMR algorithm described in Section 3.2.2 uses Richardson extrapolation to determine where additional spatial resolution is needed. Multiphysics applications that use finite-difference or finite-volume discretization can use similar extrapolation-based techniques to estimate error.

In contrast, a posteriori analysis focuses on the error in a particular computed solution. Methods for directly estimating the error in a finite-element solution include residual-based estimators that construct estimates based on element residuals, hierarchical estimators that project the residual onto some larger finite-element space, and recovery-type estimators that focus on ways to reconstruct the solution gradient (Ainsworth and Oden, 1997). Generally, these estimators are more reliable for smooth, well-resolved solutions. Extending these ideas to develop robust and accurate error estimators for coarse meshes and non-smooth solutions is an active area of research (Cai and Zhang, 2009, 2010a,b).

Alternatively, the error in some scalar model response quantity $\mathcal{Q}$ may be of interest. In this case, duality and adjoint operators can be used to obtain very accurate estimates of the error in $\mathcal{Q}$ (Estep et al., 2000; Becker and Rannacher, 2001; Oden and Prudhomme, 2002; Cao and Petzold, 2006). This approach requires solving a related problem involving the adjoint operator together with a projection of the adjoint solution in the direction of the residual. For time-dependent problems, solving a problem involving the adjoint operator entails integrating backwards in time to the initial state at every timestep. When the problem is nonlinear, some application state must be saved during the forward time integration in order to construct the adjoint operator during the backward time integration. Storage and data management requirements can make duality-based a posteriori error estimation impractical. Also, because of the way that multiphysics applications are constructed from components, global adjoints and residuals are often not available. Consequently these duality-based a posteriori techniques are not widely used.

Even if full adjoints and residuals were available, the divide-and-conquer approach commonly encountered in multiphysics applications presents special difficulties for error estimation. Significant extensions and alterations of the standard approaches are required that are well developed for single-physics problems. The key features required to obtain accurate error estimates for multiphysics problems include the following:

- Estimating the effects of errors passed between physical components;
- Estimating the effects of processing transferred information, for example, upscaling, downscaling, and changing discretizations;
- Quantifying the effects of multiphysics solution algorithms on stability by identifying perturbations to adjoint operators;

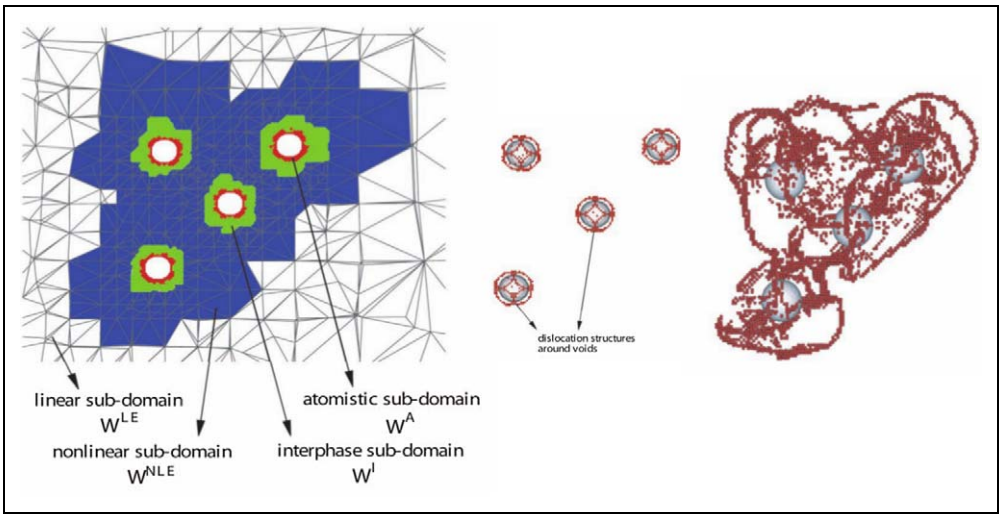


Figure 19. Four-void multiscale example. **Left:** Models adaptively selected at step 20. **Center:** Dislocation structure around voids at step 20. **Right:** Evolved dislocation structure at step 28.

- Estimating the effects of using a finite number of iterations in the solution when an iterative solver is used.

The following subsections illustrate some of these issues. In this discussion we collectively refer to these strategies as multiscale, multiphysics, multidiscretization operator decomposition (M^3OD), which encompasses multiphysics methods already discussed to this point:

- Operator splitting, or partitioned methods;
- Decoupling "loosely coupled" systems that are related through parameter passing;
- Coupled systems in which upscaling and downscaling operators are introduced;
- Incomplete iteration in block or nonlinear solvers;
- Different discretizations for different components, or in different regions or at different times in the same equation.

While these methods or issues are very different, they all share the common feature that they involve some substantial alteration of the system of differential operators representing the components of the underlying system. These modifications may arise from formulation and/or discretization and must be accounted for in order to achieve robust and reliable error estimation for multiphysics problems.

This common feature to all M^3OD approaches results in a common issue: M^3OD discretizes the instantaneous interactions between the component physics in the multiphysics model.

One potential consequence is the possibility of new forms of instability, some of which may be subtle. Another consequence is that as information is exchanged between physical components, numerical error is also exchanged, and the error in one component may well pollute the solution of another component. We illustrate the range of methods and effects using four common examples.

3.4.1 Operator splitting for reaction–diffusion equations. A classic example of M^3OD is operator splitting for a reaction–diffusion equation. Accuracy considerations dictate the use of relatively small steps to integrate a fast reaction component. On the other hand, stability considerations over moderate to long time intervals suggest the use of implicit, dissipative numerical methods for integrating diffusion problems. Such methods are expensive to use per step, but relatively large steps can be used on a purely dissipative problem. If the reaction and diffusion components are integrated together, then the small steps required for accurate resolution of the reaction lead to an expensive computation.

In multirate operator splitting, the reaction and diffusion components are integrated independently inside each time interval of a discretization of time and "synchronized" in some fashion only at the nodes of the interval. The reaction component is often integrated by using significantly smaller substeps (e.g., 10^{-5} times smaller is not uncommon) than those used to integrate the diffusion component, which can lead to tremendous computational savings.

We illustrate some possible effects for multirate operator splitting using the well-known Brusselator problem

$$\begin{cases} \frac{\partial u_1}{\partial t} - \nu_1 \Delta u_1 = 0.6 - 2u_1 + u_1^2 u_2 \\ \frac{\partial u_2}{\partial t} - \nu_2 \Delta u_2 = 2u_1 - u_1^2 u_2 \\ \text{suitable initial and boundary conditions} \end{cases}$$

with $\nu_1 = \nu_2 = 0.025$, over the one-dimensional spatial domain $[0, 1]$. We discretize using:

- a second-order, piecewise linear finite-element method in space with 500 elements;
- a standard first-order multirate splitting scheme;
- the implicit second-order trapezoidal rule with a timestep of 0.2 for the diffusion;
- first-order implicit Euler with a timestep of 0.004 for the reaction.

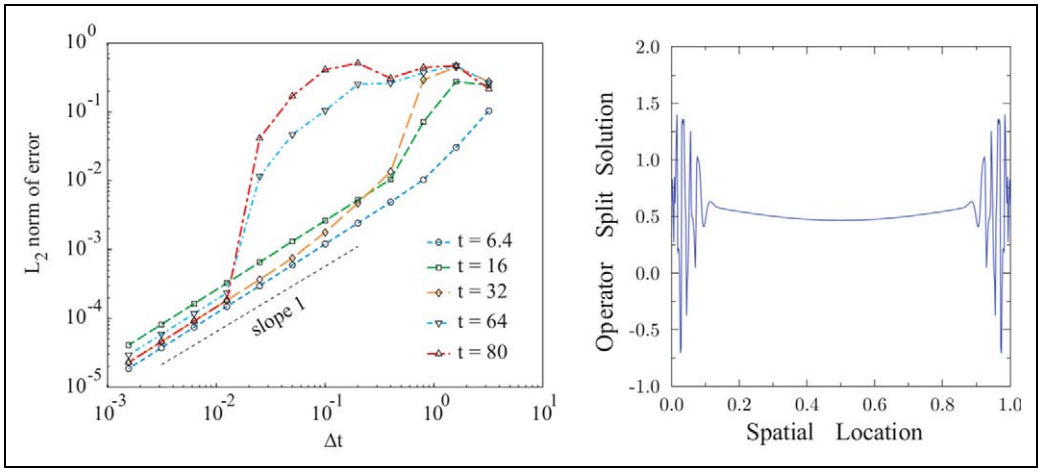


Figure 20. Left: On moderate to long time intervals, there is a critical timestep above which convergence fails. **Right:** A typical unstable solution. See Estep et al. (2008a) for details; 2008 Society for Industrial and Applied Mathematics. Reprinted with permission. All rights reserved.

In particular, the resolution is sufficient to ensure that spatial discretization errors are negligible and that the reaction and diffusion components are each solved with reasonable accuracy. Overall, the method is first-order accurate in time. See Estep et al. (2008a) for details.

We show the results in Figure 20. On the left, we plot the error versus timestep for different final times. We see that on sufficiently long time intervals, there is a critical timestep above which convergence fails. On the right, we plot a typical long time unstable solution. Note that the oscillations are *not* on the scale of the spatial discretization.

This example illustrates well the fundamental issue arising in solutions using M^{3}OD. When divorced from the stabilizing effect of diffusion, the solution of an unstable reaction component is free to grow too rapidly. In this case, the reaction component is not constrained by boundary conditions imposed on the solution of the diffusion component, and consequently its values near the boundaries grow far too rapidly. This coupling either needs to be compensated for by applying an L-stable (see e.g., Lambert, 1991; Ascher and Petzold, 1998) diffusion solver in the diffusion solve (Ropp et al., 2004; Ropp and Shadid, 2009), or the coupling needs to be implicit for large timesteps.

3.4.2 Operator splitting for advection–diffusion equations. When splitting an advection operator from a diffusion operator, a subtle inconsistency may occur at boundaries. The ensuing error will pollute both the advection and diffusion components of the solution. In some cases, the inconsistency may occur only if advection is subcycled relative to diffusion.

Consider the advection–diffusion problem

$$\begin{cases} \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, & x \in (x_L, x_R) \times (0, T] \\ u(x_L, t) = g_L(t), & t \in (0, T] \\ u(x_R, t) = g_R(t), & t \in (0, T] \\ \text{suitable initial conditions} \end{cases}$$

where $a > 0$ and $\nu > 0$ are constants. An example of an operator-split calculation is to resolve advection by using a step of Lax–Wendroff between times t^n and $t^n + \Delta t = t^{n+1}$, then using a centered scheme with a Crank–Nicholson timestep for diffusion. We illustrate this using finite difference methods at grid points $x_i = x_L + i\Delta x$ for $i = 0, 1, \ldots, M+1$ with $x_{M+1} = x_R$. At a grid point x_i and time level t^n, denote the advection degrees of freedom as v_i^n and the diffusion degrees of freedom as w_i^n, with the operator-split solution degrees of freedom being denoted by u_i^n. The discrete algorithm is defined over a timestep by setting $v_i^n = u_i^n$ for $i = 0, \ldots, M+1$, then updating the advection degrees of freedom:

$$\begin{cases} v_i^{n+1} = v_i^n - \frac{a\Delta t}{2\Delta x}(v_{i+1}^n - v_{i-1}^n) + \frac{a^2\Delta t^2}{2\Delta x^2}(v_{i+1}^n - 2v_i^n + v_{i-1}^n) \\ v_0^{n+1} = g_L(t^{n+1}) \qquad\qquad i = 1, \ldots, M \\ v_{M+1}^{n+1} = g_R(t^{n+1}) \end{cases}$$

Alternatively, the right boundary value could be extrapolated since $a > 0$. The updated advection data is used to initialize the diffusion step by setting $w_i^n = v_i^{n+1}$ for $i = 1, \ldots, M$, with boundary values $w_0^n = g_L(t^n)$ and $w_{M+1}^n = g_R(t^n)$. The diffusion update is

$$\begin{cases} w_i^{n+1} = w_i^n + \nu\frac{\Delta t}{\Delta x^2}(w_{i+1}^{n+1/2} - 2w_i^{n+1/2} + w_{i-1}^{n+1/2}), \quad i = 1, \ldots, M \\ w_0^{n+1} = g_L(t^{n+1}) \\ w_{M+1}^{n+1} = g_R(t^{n+1}) \end{cases}$$

where $w_i^{n+1/2} = (w_i^{n+1} + w_i^n)/2$. The operator-split approximation is then $u_i^{n+1} = w_i^{n+1}$, for $i = 0, \ldots, M+1$.

The truncation error for this method is found in the usual way by inserting $u(x_i, t^j)$ for u_i^j and $j = n, n+1$. Away from boundaries ($1 < i < M$), the method is consistent, with order $\Delta t^2 + \Delta x^2$. Near the boundaries, however, an inconsistency has been introduced because the boundary data used in the scheme does not generally evolve in a way that is compatible with the split operators. Indeed, one may verify directly that the local truncation error is on the order of $\frac{\Delta t}{\Delta x^2}$ at x_i in the case where $i = 1$. Special *intermediate*

boundary values may be derived to amend this issue. See Connors et al. (2012) for examples, a posteriori error analysis, and other details.

3.4.3 Iterative solution of parabolic problems coupled through a common boundary. Operator splitting is an extreme form of M^{3}OD because it is a blunt factorization of the diffusion–reaction or advection–reaction differential operator whose effects cannot be repaired by, for example, iteration. We next discuss a less severe example of M^{3}OD that nonetheless presents some difficulties and puzzling behavior.

We consider a system of parabolic problems, where each equation is posed on a different domain but the domains share a common boundary Γ. In contrast to equation (15), we impose continuity of state and normal flux across Γ:

$$\begin{cases} \frac{\partial u_1}{\partial t} - \nabla \cdot (\nu_1 \nabla u_1) = f_1, & x \in \Omega_1 \\ \frac{\partial u_2}{\partial t} - \nabla \cdot (\nu_2 \nabla u_2) = f_2, & x \in \Omega_2 \\ u_1 = u_2, & x \in \Gamma \\ \nu_1 \nabla u_1 \cdot \mathbf{n}_1 = \nu_2 \nabla u_2 \cdot \mathbf{n}_2, & x \in \Gamma \\ \text{suitable boundary conditions on the other boundaries} \\ \text{suitable initial conditions} \end{cases}$$

where $\mathbf{n}_i$ is the outward unit normal to Ω_i along Γ for $i = 1, 2$. This example arises in conjugate heat transfer, representing the exchange of heat between two abutting solids with different temperatures. It is a subsystem in the exchange of heat between a solid and an encompassing fluid. This system also arises in other contexts, for example the core–edge coupling in models of a fusion reactor.

Such a system may be posed as a large, fully implicit system of equations. However, it is common practice to use some form of iterative timestep method. The different components often use different spatial and temporal discretizations and may even be solved with different codes.

A common approach combines Algorithm 1, as an inner iteration, with Algorithm 2 and is specified in Algorithm 4. The solves that are performed on each subdomain may use subcycling in time; recall that we use $u_1(t_n)$ and $u_2(t_n)$ to denote the nodal values of the numerical solutions at N_t time nodes $t_0 = 0 < t_1 < t_2 < \ldots < t_{N_t}$. Of course, we must discretize in space to create full discretizations. Many possible variations of this approach exist.

The number of specified iterations in each step is an important factor. This approach can be interpreted as a fixed-point iteration, and if the fixed-point iteration converges, then by iterating sufficiently, this discretization converges to the fully implicit discretization. On the other hand, this approach often is implemented with only one iteration. Doing so may have significant effects on accuracy. More worrisome is that if the fixed-point iteration actually fails to converge, using only one or a few iterations will fail to reveal this fact. The numerical results are questionable in this circumstance.

To illustrate some of the behavior that can arise in this approach, we consider simple linear heat equations with

Algorithm 4.

Given initial values $u_i(t_0)$, $i = 1, 2$:
for $n = 1, \ldots, N_t$ do
  Set $u_i^0(t_n) = u_i(t_{n-1})$, $i = 1, 2$
  for $k = 1, 2, \ldots,$ (until convergence) do
    Solve for $u_1^k(t_n)$ with $u_1^k(t_n) = u_2^{k-1}(t_n)$ on Γ
    Solve for $u_2^k(t_n)$ with $\nu_2 \nabla u_2^k(t_n) \cdot \mathbf{n}_2 = \nu_1 \nabla u_1^k(t_n) \cdot \mathbf{n}_1$ on Γ
  end for
  Set $u_1(t_n) = u_1^k(t_n)$ and $u_2(t_n) = u_2^k(t_n)$
end for

diffusion coefficients $\nu_1 = 1$ and $\nu_2 = 1/2$ on domains $\Omega_1 = [0, 1]$ and $\Omega_2 = [1, 2]$, respectively, coupled by imposing continuity of state and normal flux at $\Gamma_{12} = \{1\}$. The solution should converge to a steady state. We solve using Algorithm 4 with continuous second-order finite-element methods in space with $\Delta x = 0.01$ and implicit Euler with $\Delta t = 10^{-3}$ for each component. Each component is solved sufficiently accurately for the component errors to be negligible.

We plot the total error of the approximation in Figure 21. We see the expected increase during the initial transient. However, the increase in error that occurs over longer time intervals is not expected, since the solution should be converging to a steady state. We can estimate the error that arises from incomplete iteration at each step (we use only one iteration) and the error in the numerical flux that is used in the solution of u_2. Note that u_1 produces a numerical flux that is only first order. We see that the error from incomplete iteration is significant during the initial rapid transient but becomes unimportant over long time intervals. However, the error due to transferring a numerical flux increases as time passes, and becomes the dominant feature. In fact, the pollution from transferring the numerical flux causes a loss of order in spatial accuracy. Analyses of the errors, as well as examples in two dimensions with both matching and non-matching grids along Γ, are given in Estep et al. (2008b, 2009).

3.4.4 Solution of systems of elliptic equations with independent discretizations. In the next example, drawn from Carey et al. (2009), we consider an even more nominally benign form of M^{3}OD, which nonetheless produces interesting behavior. We consider a two-component, "one-way" coupled system.

$$\begin{cases} -\Delta u_1 = \sin(4\pi x)\sin(\pi y) & x \in \Omega \\ -\Delta u_2 = b \cdot \nabla u_1 & x \in \Omega \\ u_1 = u_2 = 0 & x \in \partial\Omega \end{cases} \qquad b = \frac{2}{\pi} \begin{pmatrix} 25\sin(4\pi x) \\ \sin(\pi x) \end{pmatrix}$$

Here $\Omega = [0, 1] \times [0, 1]$. The "lower triangular" form of this system means that we can solve it as a coupled system or we can solve the first equation and then use the solution to generate the parameters for the second problem, as in Algorithm 5. In either case, we use independent discretizations of the two problems.

Solutions obtained by using a standard piecewise-linear, continuous finite-element method on uniform, highly refined meshes and Algorithm 5 are shown in Figure 22.

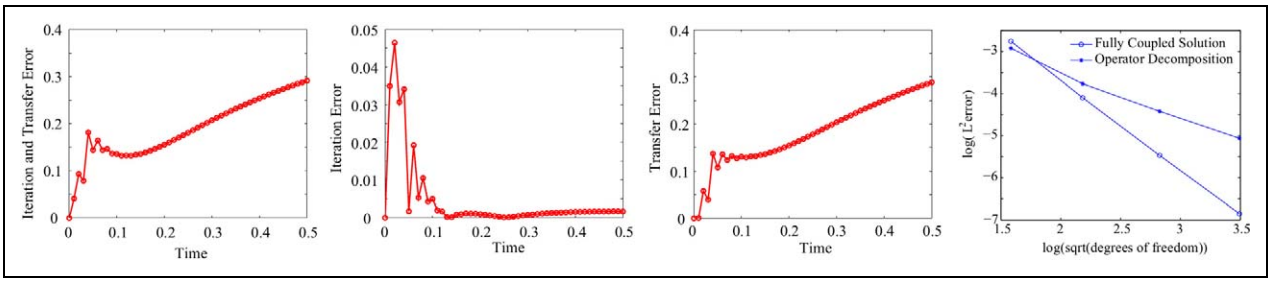


Figure 21. Results of the Gauss–Seidel algorithm for coupled parabolic problems. The component discretization errors are negligible. **Left:** the total error. **Second from left:** the contribution from incomplete iteration on each step. **Second from right:** the contribution from the passing numerical fluxes. **Right:** approximation showing loss of an order of accuracy in space as a result of the error in the numerical flux.

Algorithm 5.

Construct discretizations for the two equations
Compute the numerical solution u_1 of the first equation
Project u_1 onto the mesh for the second equation
Compute the numerical solution u_2 of the second equation

Using uniform meshes, we evaluate the standard a posteriori error estimate for the second component problem and the quantity of interest $u_2(0.25, 0.25)$, ignoring any effect arising from error in the solution of the first component. We estimate the solution error to be ≈ 0.0042. However, the true error is ≈ 0.0048, giving a discrepancy of ≈ 0.0006 ($\approx 13\%$) in the estimate. This is significant for these kinds of accurate a posteriori error estimates since the solutions are smooth.

If we adapt the mesh for the solution of the second component based on the a posteriori error estimate of the error in that component while neglecting the effects of the decomposition (Figure 23), the discrepancy becomes alarmingly worse. For example, we can refine the mesh until the estimate of the error in the second component is ≈ 0.0001. However, we find that the true error is ≈ 0.2244.

3.5 Uncertainty quantification (UQ)

While UQ is a relatively young field with many foundational research problems remaining to be tackled, UQ for multiphysics models is in its infancy. UQ is a process for determining how random variation in model inputs, such as initial/boundary conditions or parameters in material properties or constitutive laws, affects the results of models and simulations. UQ is important for using multiphysics models in engineering and scientific practice because of the complicated systems they attempt to model. In particular, multiphysics systems present severe difficulties for experimental observation. Consequently, mathematical models are heavily used to supplement the meager availability of experimental data and observations. UQ can be used to characterize and reduce uncertainties in multiphysics applications as part of a process of continuous quantifiable improvement of simulation accuracy.

Random variation in model parameters can arise from a number of sources, including measurement error and natural variability. When sufficient data is available, this randomness can be characterized in the form of probability distributions, and the random parameter is treated stochastically. Such uncertainties are generally termed *aleatory*. On the other hand, *epistemic* uncertainties generally reflect a lack of knowledge, and probabilistic treatment is not always appropriate. In practice, the effects of epistemic uncertainty can be analyzed by assuming a uniform distribution with suitable lower and upper bounds, or alternatively through non-probabilistic approaches such as interval analysis or evidence theory. When both aleatory and epistemic uncertainties are present, hierarchical sampling techniques such as second-order probability methods can be used to obtain an "envelope" of probability distributions for the output response.

In either case, one of the principal concerns of UQ is to construct probability distributions of the model responses. Doing so requires multiple deterministic runs of the multiphysics model using samples drawn from the input distribution. Current UQ practice is dominated by "black-box" non-intrusive strategies that require no modification of the application. Monte Carlo sampling and more efficient variations, such as Latin hypercube sampling, are most widely used. These approaches can require thousands of deterministic realizations to obtain the output probability distribution, especially when a large number of uncertain parameters must be accounted for. This is the so-called curse of dimensionality and is exacerbated in multiphysics simulations, where the number of parameters involved increases significantly with each physics model incorporated into the system. One way to address the expense of these runs is to construct either an emulator or a reduced-order model of the original simulator and then use this faster model in the Monte Carlo runs (Oakley, 2004). With this method, one must account for the uncertainty in the emulator as well as that of the computer model. Methods for constructing emulators include those based on interpolations or Gaussian-process approximations (Sacks et al., 1989), multivariant regressions (Box and Draper, 2007), and neural networks (Hayken, 1998). In developing reduced-order models, the goal is to identify a subspace of the state solution space in which the dominant physics reside, then project the original

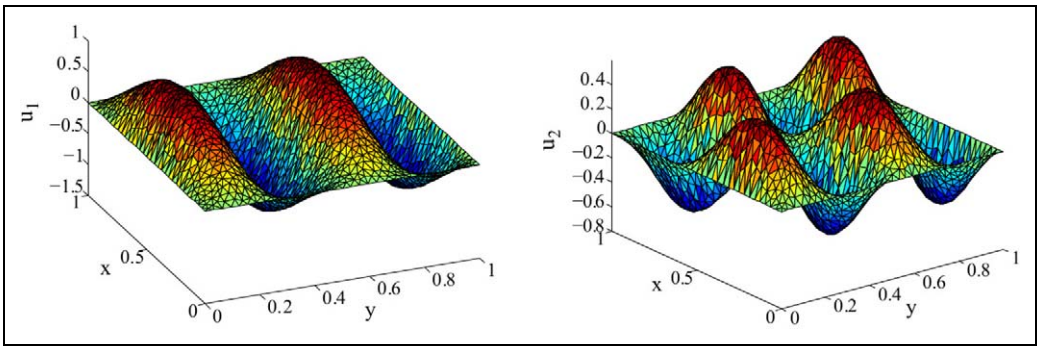


Figure 22. Solutions of the component problems of the elliptic system computed on uniform meshes. See Carey et al. (2009) for details; 2009 Society for Industrial and Applied Mathematics. Reprinted with permission. All rights reserved.

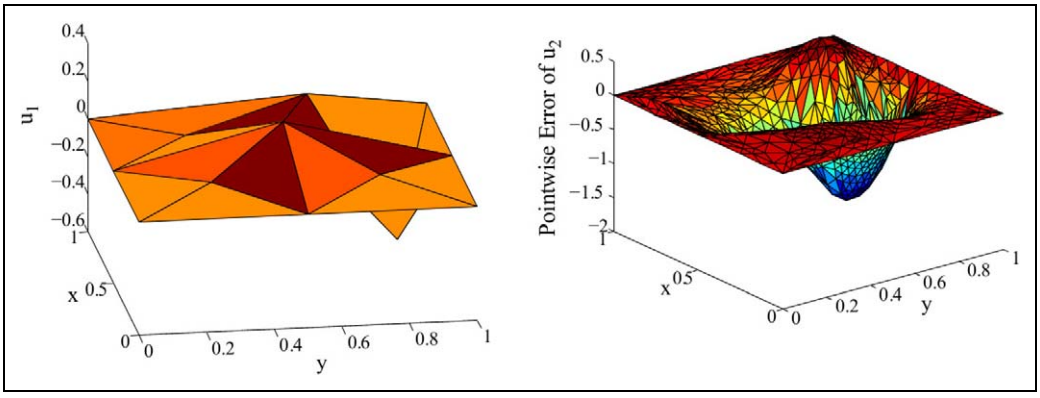


Figure 23:. Results obtained after refining the mesh for the second component so that the a posteriori error estimate of the error only in the second component is less than 0.001. The mesh for the first component remains coarse; consequently, the error in the first component becomes relatively much larger. See Carey et al. (2009) for details; 2009 Society for Industrial and Applied Mathematics. Reprinted with permission. All rights reserved.

governing equations onto this subspace (Adams et al., 2012). While much work has been done on linear, steady-state problems, the field of reduced-order modeling is very active for nonlinear and time-varying problems.

Another way the curse of dimensionality can be mitigated is by using spectral polynomial chaos expansions (PCE) (Ghanem and Spanos, 1991). While PCE can dramatically lower the cost of calculating the output probability distribution, lack of smoothness in the random parameters can impede PCE convergence, and it can be very difficult or impossible to derive the system of equations for the expansion coefficients. PCE also requires a complete rewrite of the application. In order to avoid having to rewrite the code, this method is best brought in at the start of code development; software solutions are available to generate the system of equations for the expansion coefficients (Pawlowski et al., 2012a,b). Stochastic collocation (Xiu, 2010) is a non-intrusive alternative to PCE that is still much less expensive than Monte Carlo sampling.

One approach to the multiphysics challenges for UQ is to address the multiphysics simulation as a hierarchy of models, addressing single components, then pairs, and working up to the full simulation (Adams et al., 2012). This approach allows leverage of knowledge from one stage to the next and can provide greater understanding of model coverage and correctness. The complexity of a typical multiphysics model presents significant challenges. On the one hand, the complex behavior makes it difficult to obtain informed, high-fidelity statistical models of the system (when subject to random data and parameters), so that the black-box approach to UQ becomes problematic. On the other hand, propagation of random uncertainty through a multiphysics model encounters all the difficulties that have been outlined for the numerical solution. In addition, significant "stochastic feedback" often occurs between components of a multiphysics model. A useful source for further information can be found in Adams et al. (2012).

4 Multiphysics software

Software is the practical means through which collaboration on multiphysics algorithms and applications occurs; collaboration is unavoidable because the full scope of required functionality is broader than any single person or team can deeply understand. The following quote from Steve Jobs (1997) truly resonates for multiphysics:

> The way you get programmer productivity is by eliminating lines of code you have to write.

We could expand this by saying that the only way the multiphysics research community can address increasing challenges in modeling and algorithms is by making it possible to combine and leverage diverse code for the various phases of simulation. Software for multiphysics presents a host of difficulties beyond those faced in single-physics contexts because of the compounded complexities of code interactions. Section 2.2 introduced issues from the perspective of scientific application teams, who face daunting challenges both in building application-specific infrastructure and in incorporating libraries, frameworks, and tools that are under development by many groups at universities and research laboratories. Section 4.1 expands on these issues with a closer look at the current status of multiphysics software, emphasizing capabilities needed for HPC environments. Section 4.2 discusses challenges in designing software for multiphysics algorithms as well as some possible approaches to help manage complexity and change. Section 4.3 considers practical difficulties in collaborative research software.

4.1 Status of software for multiphysics

We begin by surveying current multiphysics software practices. This information provides a background for discussing common needs. We then present a few examples of HPC multiphysics software infrastructure.

4.1.1 Current practices. Current practices for multiphysics coupling in production codes, such those introduced in Section 2, must strike a balance among performance, software reuse, numerical accuracy, and robustness, while also addressing ease-of-use for developers at all levels of the multiphysics software stack. Exactly which practices and approaches are favorable to these goals is a subject of controversy and continued research. However, some common choices exist.

The importance of software reuse has strongly favored operator-splitting coupling methodologies, possibly at the expense of numerical accuracy. Typical workhorse codes in key physics areas such as CFD, particle transport, atomistic methods, and structural mechanics can easily have hundreds of developer-years behind them, including extensive test suites, comprehensive experience with the strengths and limitations of the underlying numerical methods, active user communities, and highly optimized kernels. Since most projects using these codes are driven by application goals (scientific or engineering discovery), any significant restructuring of the codes, let alone a complete rewrite, is extremely risky for application scientists, and often completely infeasible with current software technology. Moreover, many of these applications were not originally written with extensibility in mind and depend on outdated constructs and practices in software design. Thus, their integration into a more modern, modular, component-based software architecture is often more problematic than may initially appear to be the case.

Therefore, it is fair to say that the lowest threshold to early multiphysics results involves treating the coupling variables as I/O to these legacy codes. This approach allows some initial exploration into basic coupled phenomena with very limited modification of the original codes. We also note that higher than first-order accuracy may be achieved within this framework through various iterative techniques, as introduced in Section 3, albeit possibly at a higher performance cost than other choices.

While this approach sounds relatively straightforward, non-trivial implementation issues remain, particularly on HPC architectures, which beget another set of fundamental choices. These typically fall into two general classes.

The *service-oriented architecture* approach (e.g., the fusion IPS framework (Elwasif et al., 2010)), ensures the maximum independence of each code component by basing coupling on I/O rather than variable/grid-based data formats. The key idea is that interacting with each code at the I/O data level is more flexible and is independent of the internals of the codes to be coupled. Currently file I/O is commonly used for data transfer, but it will be necessary to move toward in-memory transfers for scalability.

A standard for the meaning of the I/O format (e.g., which variables are located where on which mesh type) is still required for the coupling to be numerically consistent, but this approach avoids any direct "contact" with the internal data structures of each component. Note that, while not required, this approach can be naturally carried out in a parallel environment when each component runs as its own executable on a disjoint set of MPI processes. This approach can be implemented on top of a higher-level I/O library.

In the *remote procedure call* approach, components share information by direct method calls based on a defined abstraction of the information that the components must share. This approach does not imply that the internal data representations of each code are directly modified to accommodate the coupling; rather, more typically, interface routines are used to transmit information between the components for loading into the internal representations of the communicating components. Clearly, the more common the internal representations used by the components, the more efficient, in terms of memory and translation, the information communication process. This approach is more commonly followed with existing HPC codes and is more naturally implemented as a single MPI executable with either disjoint or overlapping processor configurations. When linking independently developed components with different internal data representations, this approach must employ a *lingua franca* perspective in which the information passed in the calls can be easily extracted from, and placed into, the internal data structures of the individual components. Such an approach was adopted by the ITAPS SciDAC center for dealing with unstructured meshes and the field (solution) information defined over those meshes

(Chand et al., 2008; Devine et al., 2009; Ollivier-Gooch et al., 2010).

We note that there is no superior multiphysics software solution in general; the best approach for a particular circumstance depends on application-specific requirements. Usually, though, service-oriented architecture approaches will require much less development work but offer less flexibility and opportunity for performance optimization, in terms of both memory and execution time.

Increasingly, the commercial simulation community is endeavoring to provide "coupling toolkits" to ease the programming burden of building multiphysics simulations. Some of these toolkits, such as MpCCI (2012), claim broad penetration and adoption in the commercial modeling world. Their basic philosophy is to allow users to specify their coupling problem at a high level of abstraction by providing the I/O variables and a description of the orchestration (time sequence of couplings). Such tools are focused on a loose coupling approach, providing the adapter code that can interpolate variables between different simulation packages. For this approach to work, the list of codes that can be coupled is hardwired into the toolkit, something that probably makes more sense in the commercial world where a larger fraction of the community uses a smaller number of codes, and thus it is feasible to support only the most popular ones. Limitations of this approach include little or no accommodation of parallel computing resources, particularly at the very high end, and lack of flexibility in potential candidate code components.

A number of projects have explored implicit coupling based on the JFNK method, introduced in Section 3. This approach requires each of the N_c code components to return a solution residual, $F_i(u)$, $i = 1, \ldots, N_c$. The approach can be taken even one step further, with a broader goal of blending the best of both worlds by building in the option to employ an operator-splitting algorithm in cases where a tightly coupled approach is not needed and to employ an operator-splitting solution as a preconditioner when a fully implicit solution is deemed necessary. Section 4.1.3 discusses software to facilitate this approach.

For all these techniques, it is important at the implementation level to categorize the range of coupling types that one encounters across applications (see e.g., Larson, 2009a,b). The frequency and volume of data movement depend greatly on these categories and go a long way in determining the difficulty of implementation or the other strategies described above.

4.1.2 Common needs. From this brief survey of current practices in production multiphysics codes, we identify common needs for HPC multiphysics software to achieve reasonable performance, robustness, portability, and flexibility, as well as to address the expectations of software developers and end-users. These needs fall into several categories that are introduced briefly below and discussed in more depth in Section 4.2. Many of these needs also exist for good-quality HPC uniphysics software. However, while it may be possible to "get away with" some poor software engineering practices in a uniphysics context, addressing these issues is imperative for effective code sharing for multiphysics applications.

- *Interfaces with appropriate levels of encapsulation and granularity while permitting adequate performance.* In designing a multiphysics software package, it is critical to determine interfaces with appropriate levels of encapsulation. If too much is encapsulated, the interface is inflexible, and users cannot access needed data. If too little, objects can easily be used incorrectly, and functionality will not be great enough to encourage adoption. See Section 4.2.1.
- *Common infrastructure and ability to exploit commonality among different physics operators.* Multiphysics software for HPC should also facilitate access to a broad set of common infrastructures that may be used by the physics codes being coupled to ease the coupling process. A clear example is mesh management (Kirk et al., 2006; Glass et al., 2011; Tautges et al., 2011c), where the use of a common mesh and field database can allow multiple codes to access a mesh in a common way and to exchange data relating to that mesh. Common verification support, including order-of-accuracy verification for multiphysics simulations, would also be of great use. Other important types of infrastructure include coupling technology, such as parallel communication schedulers, interpolation, and mortar methods to exchange data on different meshes; linear, nonlinear, and timestepping solvers that support multiphysics algorithms; automatic differentiation technology for derivative computations; error estimation and UQ software; and tools for data management, visualization, and I/O. A multiphysics software package might also help developers exploit commonality among different physics operators such as those in different types of convection-dominated systems. See Sections 4.2.2 through 4.2.4. A test-driven design that helps verify correct operation is important to manage the difficulties of incorporating such a broad range of tools that are developed by diverse multi-institutional teams and constantly changing. See Section 4.3.1.
- *Strong focus on meeting practical user expectations such as multilanguage support, ease of building, documentation, assurance of future support, and transparent open-source licensing.* In addition to functional requirements, multiphysics software must meet the practical needs of software developers and users if it is to be widely adopted. Some multiphysics software supports only C++, but Fortran and C interfaces are also necessary to accommodate the majority of physics codes, and Python is increasingly popular. Software must be sensitive to global scope issues (including symbol names and file usage), must easily integrate with

applications, and must support all relevant hardware. Because many approaches to multiphysics simulations involve combining software from several different sources, clear interface specifications are needed. Related needs are good practices for software maintenance and documentation, including tutorial material and clear examples. See Section 4.3.

- *Extensibility to the largest current and anticipated future problems and to future extreme-scale systems.* Multiphysics approaches and algorithms that can be extended to very large problems on future extreme-scale architectures are required. We note that scalability by itself is not necessarily a useful measure of performance and that some very inefficient codes may exhibit relatively good scaling. For exascale architectures, memory use must be minimized. This circumstance may make it attractive to develop custom variants of implicit methods, and to run one type of physics on some nodes and other types of physics on other nodes, requiring careful attention to load-balancing issues. See Sections 4.2.5 and 5.1.

A longer-term and ambitious goal might be to allow domain experts who do not have extensive knowledge of multiphysics coupling and possibly have little advanced HPC expertise to write robust, coupled, parallel simulation codes that exhibit reasonably good performance.

4.1.3 Software successes. The development of multiphysics software is becoming increasingly widespread. Several commercial analysis packages contain the word ''multiphysics'' in their description, and Section 2 mentions some application-specific research software. Below we list a few frameworks, libraries, and other tools that incorporate general-purpose, reusable algorithmic infrastructure in support of parallel PDE-based multiphysics applications, and aim to address the needs identified above. Of course, many additional projects address important aspects of multiphysics simulations. Such projects include approaches that facilitate working at higher levels of abstraction (Sahu et al., 1999; Bulatewicz and Cuny, 2006; Ford et al., 2006; McInnes et al., 2006; DeVito et al., 2011), as well as software such as Cactus (Goodale et al., 2003), Coupler (Liu and Sosonkina, 2011), DDEMA (Michopoulos et al., 2003), FLASH (Dubey et al., 2009), InterComm (Sussman, 2006), Jade (Hegewald et al., 2008), Karma (Mahadevan et al., 2009), MCT (Larson et al., 2005), MUSE (Zwart et al., 2008), Overlink (Grandy, 2004), Overture (Henshaw, 2002), PreCICE (Gatzhammer et al., 2010), Sierra (Stewart and Edwards, 2004), and XpressSpace (Zhang et al., 2011a).

BoxLib. BoxLib (Bell et al., 2012) is a software framework designed to facilitate the development of parallel, block-structured AMR applications. The block-structured approach to AMR used in BoxLib is based on representing the solution on a sequence of nested grids with successively finer resolutions. The data at a given level of resolution is organized into a collection of logically rectangular grid patches, each containing a large number of points. The aggregation of grid patches at a level, referred to as a MultiFab, is the fundamental parallel abstraction used in BoxLib. Coarse-grained parallelism is expressed by distribution of the grid patches in a MultiFab to nodes of the system. Operations at the node level are then specified in terms of iterators with an owner-computes rule. BoxLib supports a hybrid parallelization model in which OpenMP directives are used within a node to thread operations on a grid patch. A number of applications have been implemented within the BoxLib framework. These include LMC for low Mach number combustion (Day and Bell, 2000), CASTRO for RHD (Almgren et al., 2010; Zhang et al., 2011b), MAESTRO for low Mach number astrophysical convection (Almgren et al., 2006a,b, 2008; Zingale et al., 2009; Nonaka et al., 2010), PMAMR for subsurface flow (Pau et al., 2009) and NYX for computational cosmology (Almgren et al., 2012).

Chombo. The Chombo library (Colella et al., 2000) provides a set of tools for implementing finite-volume methods for the solution of PDEs on block-structured, adaptively refined rectangular grids. Both elliptic and time-dependent modules are included. Chombo supports calculations in complex geometries with both embedded boundaries and mapped grids, as well as coupling with fast particle methods. Parallelization is currently achieved through a highly scalable domain decomposition strategy involving the distribution of rectangular grid blocks across processors (Van Straalen et al., 2011). Chombo has been used in a variety of multiphysics contexts, including biological flow, compressible gas dynamics, hydrology (Ateljevich et al., 2010), ice sheet modeling (Cornford et al., 2012), and plasma physics (Dorr et al., 2010). Of particular use in these multiphysics simulations is Chombo's open-source framework for temporal and spatial discretizations, which enables the scalable integration of multiple physical models, including external solvers, specialized physics codes, and custom algorithms.

Escript/Finley. Escript (Gross et al., 2007; Escript/Finley, 2012), is a Python-based programming tool for mathematical modeling based on nonlinear, time-dependent PDEs. Escript has been designed to give modelers an easy-to-use environment for developing and running complex and coupled models without accessing the underlying data structures directly. Escript employs the Finley finite-element method solver library. The code has been efficiently parallelized using MPI, OpenMP, and hybrid modes. Applications include modeling Earth mantel convection, earthquakes, porous media flow, reactive transport, plate subduction, and tsunamis.

Illinois Rocstar. Illinois Rocstar (2002) is a general-purpose numerical solver for fully coupled, time-dependent fluid, structure, and combustion problems that grew out of the work in the Center for Simulation of

Advanced Rockets (CSAR) (Heath et al., 2010). Rocstar was designed from the beginning to be fully parallel and is routinely run on systems with over 1000 processors. The Rocstar framework is built around Roccom, which manages the communication among the components. These include modules for timestepping, mesh modification, I/O, and performance profiling. While originally developed for solid propellant rocket motors, Rocstar has been applied to a variety of different problems, including simulations of the noise from helicopter blades and the effect of wind on tall buildings.

LIME. The Lightweight Integrating Multiphysics Environment (LIME) is a software package for coupling multiphysics simulation codes (Schmidt et al., 2010; Pawlowski et al., 2011). It is used when application codes for solving different parts of a multiphysics problem need to be coupled together in either a loose or a tight sense. LIME supports a number of coupling modes, ranging from simple, nonlinear block Gauss–Seidel and nonlinear block Jacobi to fully coupled Newton-based methods. In addition to the solution drivers, LIME provides an abstract interface to application physics codes based on the Thyra: :Model Evaluator in Trilinos. LIME has been successfully used on the BRISC project (Schmidt et al., 2010), to integrate six separate codes to assemble a virtual nuclear reactor model; the various couplings in the aggregate system include nonlinear block Gauss–Seidel, nonlinear Schur (nonlinear elimination), and JFNK. LIME is currently used in the Consortium for Advanced Simulation of Light Water Reactors (CASL) Project (Kothe et al., 2012).

MOAB. MOAB (Tautges et al., 2004, 2011b,c) is a library for query and modification of structured and unstructured meshes and field data associated with the meshes. MOAB provides a solution transfer capability for multiphysics approaches in which different physics are solved over the same volumetric domain, with each physics implemented on its own mesh and no assumptions made about mesh nesting or the two physics using the same element type or discretization. MOAB's solution transfer tool (Tautges and Caceres, 2009) breaks the process into four stages: initialization, point location, interpolation, and conservation/normalization. During the initialization stage, the two meshes are instantiated in MOAB's parallel representation, and spatial searching trees on the source mesh are constructed. Point location requires finding the source element and parametric location in that element of each target point (and the processor on which that element resides). Interpolation maps the source field to the target points, and conservation/normalization imposes global integral constraints on the mapped field. This capability is the basis for solution transfer being developed as part of the SHARP application discussed in Section 2.1.3 and also has been used to transfer radiation heat deposition results from the MCNP code (Brown, 2012) to a tetrahedral mesh for heat transfer/fluid flow computations (Sawan et al., 2009). Additional MOAB applications include Nek5000 (Fischer et al., 2012), the Denovo neutron transport code (Evans et al., 2010), and an ice-sheet modeling code (Brown, 2010; Biezuner et al., 2012).

MOOSE. The Multiphysics Object-Oriented Simulation Environment (MOOSE) (Gaston et al., 2009a,b) focuses on tightly coupled multiphysics applications using JFNK methods and is centered on use of the residual abstraction, given by equation (3). MOOSE uses a layered approach to provide core services in support of multiphysics applications. The heart of MOOSE is the Kernel abstraction, which represents a physics equation or term in an equation and provides core functionality for residual and Jacobian evaluation. One may consider a Kernel to be a mathematical operator, such as a Laplacian or a convection term in a PDE; Kernels may be swapped or coupled together to achieve different application goals. MOOSE employs the libMesh finite-element framework (Kirk et al., 2006), which provides utilities for parallel finite-element computation, including mesh I/O, a finite-element library, and interfaces to nonlinear solvers in PETSc and Trilinos. Multiphysics applications using MOOSE include BISON (Hansen et al., 2009a; Newman et al., 2009a,b) (which models nuclear reactor fuel performance computation across diverse scales that involve both the continuum and subcontinuum (Tonks et al., 2009, 2010)); PRONGHORN (an analysis code that models coupled coolant flow and neutronics for very high temperature gas cooled reactors (Park et al., 2009a, 2010; Knoll et al., 2009, 2011)); and geoscience analysis applications (Guo et al., 2010; Gaston et al., 2012).

OpenPALM. The main idea of the OpenPALM philosophy is to create applications by assembling independent components in sequences and letting them exchange data (Buis et al., 2006; Piacentini et al., 2011). The PALM framework performs the scheduling and synchronization of the components and the data exchanges. A PALM application can be described as a set of computational units arranged in a coupling algorithm. The different units are controlled by conditional and iterative constructs and belong to algorithmic sequences called computational branches. Inside a branch, the coupled independent programs are invoked as if they were subroutines of the branch program. The main features of PALM are the dynamic launching of the coupled components, the full independence of the components from the application algorithm, the parallel data exchanges with redistribution, and the separation of the physics from the algebraic manipulations performed by the PALM algebra toolbox. PALM multiphysics applications include oceanography, climate modeling, CFD, combustor cooling system optimization, and hydrology (OpenPALM, 2012).

PETSc. The Portable Extensible Toolkit for Scientific computing (PETSc) (Balay et al., 1997, 2012) incorporates

generic parallel programming primitives, matrix and vector interfaces, and solvers for linear, nonlinear, and transient problems. PETSc emphasizes ease of experimentation so that every component has a plug-in architecture by which the user can provide first-class implementations. The strong encapsulation in this design facilitates runtime composition of hierarchical methods for coupled problems. Several relatively recent components have substantially improved composability for multiphysics and multilevel methods (Brown et al., 2012; Brune et al., 2012; Smith et al., 2012). DMComposite, a distribution manager for coupled problems, handles the algebraic aspects of ''gluing'' together function spaces, decomposing them for residual evaluation, and setting up matrices to act on those coupled function spaces. DMComposite contains support for the Nest matrix format, which stores the submatrix associated with each physics component independently. A common matrix assembly interface is available, such that individual physics modules can assemble parts of global matrices without needing global knowledge or specializing on matrix format (Nest or monolithic). The FieldSplit preconditioner solves linear block systems using either block relaxation or approximate block factorization (as in ''physics-based'' preconditioners for stiff waves (Knoll et al., 2003) or block preconditioners for incompressible flow (Elman et al., 2008)); FieldSplit can be nested inside other preconditioners, including geometric or algebraic multigrid, with construction of the hierarchy and other algorithmic choices exposed as runtime options. Recent enhancements to the TS component support IMEX schemes as introduced in Section 3.2.2. Multiphysics applications using these capabilities include lithosphere dynamics (Aagaard et al., 2012), subduction and mantle convection (Katz et al., 2006, 2007; May and Moresi, 2008), ice sheet dynamics (Katz and Worster, 2010; Tautges et al., 2011a), subsurface reactive flow (Lichtner et al., 2010), tokamak fusion (Cary et al., 2009, 2011; Hakim et al., 2012; McCourt et al., 2012), mesoscale materials modeling (Wang et al., 2011a), and power networks (Abhyankar, 2011).

SUNDIALS. SUNDIALS (Woodward et al., 2012) is a suite of advanced computational codes for solving large-scale problems that can be modeled as a system of nonlinear algebraic equations, or as initial-value problems in ordinary differential or differential–algebraic equations. The current packages are KINSOL, CVODES, and IDAS, respectively. CVODES and IDAS are based on linear multistep methods for time-dependent problems and include forward and adjoint methods for carrying out first-order sensitivity analysis with respect to model parameters or initial conditions. ARKODE (Reynolds, 2012) is a package in development for SUNDIALS that focuses on the multiple-time-scale nature of multiphysics simulations. To this end, ARKODE provides high-order adaptive time-integration methods for fully implicit, fully explicit and mixed IMEX problems, as introduced in Section 3.2.2. Since multiple-time-scale problems may often be split into separate components corresponding to fast and slow scales (e.g., variables, equations, or even individual terms within equations), mixed IMEX solvers may apply the appropriate integrator to each component. Based on the family of additive Runge-Kutta methods (Ascher et al., 1997; Kennedy and Carpenter, 2003), the integrators in ARKODE allow a highly accurate and stable coupling between these split components, along with robust error estimation and time-adaptivity control (Gustafsson, 1991, 1994; Söderlind, 1998, 2003). For problems that include an implicit component, SUNDIALS, including the new ARKODE package, employs a full or modified inexact Newton method along with interfaces for dense, banded and Krylov iterative linear solvers. These iterative solvers may in turn employ a user-supplied preconditioner (often provided by a separate solver) to accelerate convergence and scalability of the method. All operations performed in SUNDIALS, except the dense and banded linear solvers, utilize a data-structure neutral model for algorithm construction, allowing their use from within a wide array of software frameworks.

Trilinos. Trilinos (Heroux et al., 2005, 2012) incorporates algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale multiphysics problems. A unique design feature of Trilinos is its focus on stand-alone, light-weight packages for scalable linear algebra, linear and eigen solvers, nonlinear analysis tools (nonlinear solvers, optimization, stability, bifurcation, UQ), I/O, discretizations, meshing, geometry, and load balancing. A key multiphysics capability is Thyra, an abstract operator/vector layer that can be used to build composite or blocked systems with contributions from different application codes. These block operators can be passed to Trilinos solvers, such as the NOX nonlinear solver for applications requiring Newton-based algorithms. Leveraging the blocked operator approach, the Teko package can be used to construct block-based and physics-based preconditioners. This capability has been demonstrated on CFD/MHD systems (Shadid et al., 2010a). Additional multiphysics-related support includes Moertel (Gee and Hansen, 2012), which supplies capabilities for non-conforming mesh tying and contact formulations in two and three dimensions using mortar methods, and Phalanx (Pawlowski, 2012), which provides a local field evaluation kernel that decomposes a complex problem into a number of simpler problems with managed dependencies (Notz et al., 2012). Multiphysics applications using Trilinos include climate modeling, reactor fuel performance modeling (Hansen, 2011), and Charon, which has been used to model MHD (Shadid et al., 2010b), multiphase chemically reacting flows (Musson et al., 2009), and semiconductor drift diffusion modeling (Lin et al., 2009).

Uintah. A good example of a production-strength multiphysics framework is Uintah (Parker, 2006; Parker et al., 2006), a set of software components and libraries that facilitate the solution of PDEs on structured AMR grids using hundreds to thousands of processors (Luitjens et al.,

2008). Uintah employs an abstract task graph representation to describe computation and communication. It can integrate multiple simulation components, analyze the dependencies and communication patterns between them, and efficiently execute the resulting multiphysics simulation. The underlying methods inside Uintah are a combination of standard fluid-flow methods and material point (particle) methods. Uintah was developed within the Center for Simulation of Accidental Fires and Explosions (Pershing et al., 2010) to solve reacting fluid–structure problems involving large deformations and fragmentation.

waLBerla. The Widely Applicable Lattice-Boltzmann from Erlangen (waLBerla) (Feichtinger et al., 2011) is a massively parallel framework for simulating multiphase flows based on the lattice-Boltzmann method. The flow simulator can be coupled to a rigid body dynamics solver (PE, physics engine) (Iglberger and Rüde, 2011) using an explicit FSI technique, with the goal of, for example, simulating particulate flows or suspensions as they appear in many technical, environmental, or biophysical applications. A highlight of the package is its careful performance-optimized design and good parallel scalability that has been demonstrated on more than 250,000 processor cores (Götz et al., 2010). Additionally, the mesoscopic CFD method can be coupled to finite-element-based PDE solvers in the waLBerla package.

4.2 Challenges in multiphysics software design

While current multiphysics applications (see Section 2.1) and software (see Section 4.1) have already demonstrated substantial advances, numerous software engineering challenges remain, providing opportunities for long-term research focusing on stable, accurate, robust, efficient, and scalable multiphysics algorithms, with easy extensibility for application-specific customization. In Sections 4.2.1 through 4.2.6 we discuss challenges in multiphysics software design and suggest some best practices. Related issues are discussed by Post and Kendall (2004) and Miller et al. (2004).

4.2.1 Enabling introduction of new models, algorithms, and data structures. It is particularly important for multiphysics software to attempt to balance the competing goals of interface stability and software reuse with the ability to innovate algorithmically and develop new physical models. Interface stability is important for several reasons:

- Decoupling the software development process, which a single person or single team cannot manage alone;
- Enabling reuse of model components for different kinds of analysis, for example, transient, steady-state, sensitivity analysis, optimization, and UQ;
- Enabling use of each "physics module" for stand-alone analysis and in various coupled configurations.

A related point is the need over time to re-examine interfaces in light of recent advances in mathematics and programming models. For example, providing (possibly optional) access to additional information (e.g., derivatives, not just fields, or other internal states) can inform library design for both couplers and solvers.

We discuss two important aspects of software design that promote interface stability and facilitate experimentation with existing approaches as well as the introduction of new capabilities over the lifetimes of multiphysics codes:

1. Library interfaces that are independent of physical processes and separate from choices of algorithms and data structures, and
2. Abstractions for mathematical objects such as vectors and matrices, which enable dealing with composite operators and changes in architecture.

Library interfaces. An important provision for extensibility is the separation of logically (or mathematically) distinct concerns into modules with library interfaces. Because a library cannot make assumptions about program startup or the use of state, it cannot seize control of "main" or assume MPI_COMM_WORLD; see Section 4.2.6 for additional software engineering issues. When preparing an application for use in a multiphysics context, a clear division should be made between the software components responsible for discretization of a physical model and the analysis being performed with the discrete model. The single-physics "analysis" component will typically be bypassed in a multiphysics context, so the interfaces between these components should be thought out carefully, especially with respect to ownership and hidden state.

A key concept is that any state data must be explicitly exchanged through an interface to maintain consistency; examples include common algebraic interfaces, such as vector and matrix classes available via PETSc and Trilinos. This approach may be used to specify inhomogeneous boundary conditions that would otherwise often be computed in a preprocessing stage. It is usually not critical which classes are used by a coupling interface, especially for vectors, because the memory backing a vector from one package can be rewrapped as a vector from a different package without a copy (though dynamic adaptivity and interactions with fast processors such as graphics processing units (GPUs) can cause complications).

Interfaces that are independent of physical process. Whenever possible, interfaces should be independent of the physical process and discretization in use; otherwise, they will prevent reuse of analysis/solver components and be brittle when different physical processes or discretizations are employed. It is still important, however, to provide enough "hooks" to implement the desired algorithms for a specific problem and to avoid excessive overhead (in terms of memory or time).

A representative example is the choice of whether to expose a semi-discrete or fully discrete interface for time integration. In the former case, one assumes that a method-of-lines approach is used and writes $\partial_t u = f(u)$, as given in coupled form by equations (2a)–(2b), where the vector u contains all state variables. If implicit methods are to be used, the semi-discrete form may also provide derivative information, such as approximations to $\partial f/\partial u$ and preconditioners.

Responsibility for timestepping is relinquished by the physics module, enabling reuse of software for time-integration schemes, including steady-state solvers and sensitivity or stability analysis methods, without modification or special support by the physics module. This approach is preferable from a software modularity perspective, but it has limitations, namely, little or no support for dynamically adaptive meshes and smaller, stable timesteps (for some transport discretizations). For multiphysics simulation where software reuse and algorithmic flexibility in coupling and analysis algorithms are often necessary, the semi-discrete form is usually desirable.

Interfaces that are independent of choices of algorithms and data structures. Different data structures are better suited to different solution strategies, especially when implicit solvers are used as part of a multiphysics simulation. The best solution strategy is often not known a priori or depends on the problem and computer architecture. For example, coupled problems can be solved by using monolithic methods involving a single sparse matrix representing all components or by split methods in which each component manages its own linear algebra. A common hybrid approach is to adopt a global Newton–Krylov method (see Section 3.1), with the preconditioners involving splitting and perhaps approximate Schur complements. This case requires the action of operators on the global system, but most of the work is done preconditioning single-physics and auxiliary (e.g., approximate Schur complement) systems. When multigrid methods are used, the splitting can be performed at the top level with independent single-physics multigrid, or multigrid can be applied directly to the global problem (often with splitting used to define the smoother) (Adams, 2004; Hron and Turek, 2006; Elman et al., 2008; Barker and Cai, 2010; Gee et al., 2011).

In the mathematical (algebraic) description of the solver, the distinction between these approaches involves a different order of operations for coarsening and extracting submatrices. Monolithic domain decomposition methods use different submatrices (e.g., corresponding to overlapping Dirichlet problems, non-overlapping Neumann problems, or combinations (Smith et al., 1996; Mandel et al., 2005; Toselli and Widlund, 2005; Dohrmann and Widlund, 2010)). When a suitable preconditioning algorithm is used, the data structures should exploit single-physics structure such as a constant block size or symmetry for memory and floating-point performance. While hierarchical or nested matrix formats generally require more messages to perform an operation with top-level semantics, they can also be more latency tolerant and involve operations on smaller communicators, which may be beneficial for performance.

High-level abstractions. As discussed above and in Section 4.1.3, mathematical abstractions from the fields of linear and nonlinear solution methods are effective software design elements for multiphysics. Also proving useful is template-based generic programming (Pawlowski et al., 2012a,b) and related work on even higher-level abstractions, including variational forms available via FEniCS (Logg et al., 2012), Sundance (Long et al., 2010), and COMSOL (2012). Such abstractions enable higher-level library components (such as linear, nonlinear, and timestepping solvers) to be written as operations on mathematical objects (such as vectors and matrices), thereby minimizing the cost of developing software for each new architecture (and its variations) and enabling composite operators.

Dealing with changes in architecture. The end of frequency scaling for microprocessors is forcing significant changes in computer hardware to meet ever-increasing needs for performance. Some applications are exploring the use of fast processors (for example GPUs); in the future, all HPC applications will need to adapt to a range of new architectures. Mathematical objects such as vectors and matrices enable specific implementations to be employed for underlying use of GPUs, OpenMP, and so forth. Also useful in this context are guided code generators (see e.g., Baumgartner et al., 2005), where a mathematical representation is combined with a tool designed for particular operations.

Composite operators. Particular to multiphysics coupling is the ability to build block composite operators that aggregate physics applications for fully coupled solution techniques. For example, one could build a composite abstract vector from two physics applications that each supply their own residual vector. Generic tools for building composite block operations take as input abstract operators and produce new abstract operators that can be handed off to solvers.

Also useful is the approach of combining kernels of code to form terms and equations that contribute to the overall multiphysics system residual. These kernels also can supply an approximate Jacobian contribution to be assembled by the coordinating framework or library to be used for preconditioning operations. For example, in MOOSE, these kernels are usually composed of volume and surface integral terms from a variational form of a governing equation, written in residual form.

Another layer of abstraction is the interface to an application that a multiphysics driver requires to support various coupling techniques. Such interfaces can be difficult to define because of the need for balancing design requirements, including flexibility to support solution algorithm requirements, efficiency, and ease of integration. To this end, LIME (Pawlowski et al., 2011) (see Section 4.1.3)

provides a high-level domain model for multiphysics coupling, which supports the CASL project (Kothe et al., 2012). This domain model describes the analysis in terms of residuals,

$$F_i(\dot{u}_i, u_i, \{p_{i,l}\}, \{z_{i,k}\}, t) = 0$$

transfer operators,

$$z_{i,k} = r_{i,k}(\{u_m\}, \{p_{m,n}\})$$

and responses.

$$g_j(\{\dot{u}_i\}, \{u_i\}, \{z_{i,k}\}, \{p_{m,n}\}, t) = 0$$

Here, F_i is the residual vector for physics i; u_i is the vector of unknowns corresponding to physics i; $\dot{u}_i = \partial u/\partial t$ is the vector of derivatives of unknowns corresponding to physics i with respect to time t; $\{p_{i,l}\}$ is the set of all independent parameters for physics i; $\{z_{i,k}\}$ is the set of coupling parameters for physics i; and $r_{i,k}$ is the transfer function k for physics i, where physics i has $k = 1, \ldots, n_{ik}$ transfer functions. A complete implementation of the domain model above should support all multiphysics coupling algorithms, ranging from loose to tight coupling. Based on the model, if less information is available from an application, the above interface is contracted, thereby limiting the available application coupling algorithms. Five forms of reduced domain models are identified (Pawlowski et al., 2011): the coupling elimination model $p \rightarrow g(p)$, the state elimination model $p \rightarrow u(p)$, the fully implicit timestep model $f(u,p) = 0$, the transient explicitly defined ODE model $\dot{u} = f(u,p,t)$, and the transient fully implicit DAE model $f(\dot{u},u,p,t) = 0$.

4.2.2 Sharing methods and codes among application fields. One of the primary challenges with software development for multiphysics is sharing and integrating algorithms across coupled applications. As codes are integrated into a multiphysics framework, each application may bring in a number of required library dependencies. The reliance on such a large number of libraries can cause difficulties, since software engineering standards and portability can vary dramatically from code to code. Unifying or standardizing common software around libraries with exceptional software practices (domain-driven design, object-oriented design, test-driven design, coverage testing, etc.) and portability will be critical from a maintenance standpoint. Areas where sharing of tools and methods is important include meshing, solvers, data mapping, visualization, I/O, interpolation/mortar methods, and coupling libraries.

The current practice is to develop common components that are released as a framework or library. Two paths are typically followed. The first path is the monolithic Framework (Framework with a capital "F"), which directs and controls all aspects of the problem, including memory management. This approach tends to allow rapid development, since all code is written to concrete implementations and much of the needed utility code is in place. However, the approach tends to lack flexibility because it forces applications to adopt the framework's data structures and algorithms. The current trend is to move away from monolithic frameworks, since multiphysics coupling requires that applications "play well together": they cannot insist on being in control of the overall coupled simulation. The second path is to develop frameworks that collaborate (framework with a lower-case "f"). These typically code to interfaces to allow for maximum flexibility. This approach allows users to provide their own concrete implementations of data structures at the cost of being somewhat more complex. See the glossary for additional discussion.

Data layout. Developing data layout standards is a challenging problem. Optimal data structures can vary for different applications and for different hardware architectures (especially in light of GPU and exascale efforts). In addition, requirements for data structures for distributed computation can be vastly different from requirements for structures for local computation. While supplying a common data structure or interface is convenient from the standpoints of development and coupling, this approach can result in a compromise on performance: a trade-off among flexibility, portability, and efficiency. Moreover, the integration with legacy codes where little knowledge or testing is in place may prohibit the adoption of new data structures.

Common infrastructure. From a software engineering standpoint, leveraging a common infrastructure for source-code management, build, and test environments is challenging for multiphysics integrated systems. Unifying all application codes under a single build/test system can be difficult and time-consuming. Further complexity is added if the application must support stand-alone builds outside the coupling framework. This could lead to inefficient resource allocation in maintaining multiple build systems. An example of a unifying build/test environment is the CASL project (Kothe et al., 2012). The approach taken with CASL was that if an application was under active development, then it was transitioned to natively use the build system and dependency tracking of the coupling framework, thereby easing integration issues.

Commonality in physics operators. Another issue is how to exploit commonalities among different physics operators. For example, convection-dominated systems can appear in a number of fields. Sharing both the mathematical methods and the software infrastructure for these types of problems can be difficult. An open research issue is how to decompose the operators. Additionally, as the nature of one physics operator changes, how does this affect the performance of the coupling and solution strategies? An example of leveraging similar operators is the Charon code (Shadid et al., 2010b) applied to both semiconductor device physics and MHD. In this case, both systems were used for convection-dominated flows. Not only were the fully

implicit solution strategies and mesh database shared, but so were the stabilization algorithms.

4.2.3 Multiphysics spatial discretizations. Two general strategies are available for spatial integration within multiphysics applications, where the multiple physics models each apply across the entire domain (Hansen and Owen, 2008; Dubcova et al., 2011). The first strategy is based on the use of a single mesh that spans the entire domain. Each physics model is hosted in terms of the basic operators supported on the mesh. The second strategy uses multiple meshes, in the extreme using a unique mesh for each physical model present in the code. This strategy is common when multiple codes are coupled; each code tends to use a mesh tailored to its requirements.

The use of a single mesh is straightforward in concept. However, this is almost always a compromise, since each physical model rarely has similar refinement requirements at a given location in the domain. If a single mesh is used, the mesh must be fine enough everywhere to support the most demanding physical model at that location. Moreover, the developers of the composite application must agree on a meshing strategy that meets the quality requirements and discretization assumptions of all concerned. On the other hand, if these issues can be overcome, this approach has several advantages. First, a single mesh incurs no error in transferring (or interpolating) data between models, since they all are hosted on a single mesh. Second, a single mesh is easier to implement, decompose for parallel execution, and load balance during a transient simulation.

The use of multiple meshes to host a multiphysics problem has nearly opposite considerations. Creating multiple meshes typically magnifies the cost, in person time, of running the application. In some cases, the problem domain may be sufficiently complex that generation of a single mesh is so expensive that the use of multiple meshes is not cost-effective. In addition, as discussed in Section 3.2.1, care must be taken in communicating (transferring) the data between meshes and models in order to avoid sacrificing the accuracy of the final result (Jiao and Heath, 2004a; Johnson et al., 2011).

One option aimed at maintaining the advantages of both approaches is a multimesh assembly strategy (Dubcova et al., 2011) to achieve the spatial diversity advantages of a multiple mesh approach without sacrificing the accuracy of the single mesh approach. For example, Hermes code (Solin et al., 2012) generates multiple hierarchically related meshes using a general version of the finite element method (FEM) known as *hp*-FEM, where *hp*-FEM provides for general adaptation and high-order convergence across the composite multiphysics problem. Each of these meshes hosts a physical model that is appropriately fine (in the *hp* sense) for the desired accuracy of each model. The composite multiphysics model is also refined to meet a global accuracy requirement. The *hp*-FEM assembly process then integrates across the different models and meshes to provide a coupled solution without using data transfer. Thus, neither data transfer error nor multiphysics coupling error is present in this approach.

In those cases where the individual physics analysis components being linked have independent meshes and internal data representations, a general remote procedure call approach must be taken. Keys to the success of this approach are the abstractions used to define the mesh, solution parameters, and field information, defined over the entities in the mesh. Since the mesh is a discrete representation of the domain of the simulation, it is also desirable that the mesh entities maintain their relationship to the overall definition of the domain. Such relationships are particularly important when adaptive mesh control methods are used and the overall definition of the domain is a solid model as defined in a system for computer-aided design (CAD). The field information, which defines the parameters in the mathematical equations being solved, is best defined in terms of tensor quantities defined over the appropriate geometric entities. In cases where a high-level domain representation is used, the input parameters (loads, material properties, and boundary conditions) can be defined as distributions over the geometric model entities (O'Bara et al., 2002). As part of the discretization process, these fields are mapped onto the mesh. The parameters that are solved for will be defined over the mesh and have known distributions as defined by the equation discretization processes. Maintaining knowledge of the distribution information is key to being able to properly map solution parameters between meshes in a multiphysics simulation.

The fundamental differences in representations used for structured and unstructured meshes indicate that the abstractions best suited for interacting with them are different. In the case of structured meshes, the two key components that must be addressed are the representation of the individual mesh blocks and the interactions between those blocks. In all cases, a compact representation of the mesh blocks in terms of a limited number of spacing and geometric mapping parameters is needed. The representation of the interactions between blocks varies strongly among the various methods (matched mapped blocks, overlapping Cartesian blocks, cut cells, etc.); thus, a separate abstraction tends to be used for each representation.

With unstructured meshes, topological entities from zero to three dimensions, and the adjacencies of those entities (Beall and Shephard, 1997; Ramacle and Shephard, 2003), provide an ideal abstraction for defining the mesh as well as maintaining the relationship of the mesh entities to the geometric domain that is typically defined as a boundary representation. The various ITAPS iMesh implementations (Chand et al., 2008; Devine et al., 2009; Ollivier-Gooch et al., 2010), and others, are based on such a topological abstraction of the mesh. Although it is possible to base the representations used on only the specific mesh entities and adjacencies used in the linkage of specific components, many implementations employ a complete representation in the sense that any of the desired 12 mesh adjacencies can

be determined in O(1) time (i.e. are not a function of the number of mesh entities) (Beall and Shephard, 1997; Remacle and Shephard, 2003; Seol and Shephard, 2006).

The parallel execution of multiphysics solution transfer requires extension of the mesh representations that maintain appropriate linkages between the entities in the mesh as they are distributed over the memories of the parallel processors. With structured meshes, the mechanisms for doing this are a strong function of the methods used to understand the interactions between the mesh blocks. For example, the parallel implementation of overlapping mesh blocks will often have one or more individual mesh blocks on a process, along with information on the mesh blocks on other processes with which those blocks interact, whereas with a tree-based decomposition a parallel tree representation might be used (e.g., a parallel octree). In the case of unstructured meshes, the common method of parallel distribution employs mesh patches of neighboring elements defined such that the numbers of mesh entities that share common boundaries with mesh entities on other processes areas are as small as possible. In this case the key component of parallelization is the addition of copies of the common bounding mesh entities on all neighboring processes with links between them (Seol and Shephard, 2006). A mechanism that can be employed to potentially support efficient parallel solution transfer is to "ghost" mesh entities, and their field data, from one process onto neighboring processes that happen to need that information to support solution transfer operations. Since only one process is allowed to change the field data, specific care must be applied to ghosted information to be sure it is up-to-date on any process that will be using it.

4.2.4 Timestep control. Selection of an appropriate timestep for accuracy in a multiphysics calculation can be challenging. Software features with utility for multiphysics calculations span the needs of uniphysics codes, such as robust capabilities for checkpoint and restart and options for timestep control (e.g., manual, programmed, semi-automatic, and automatic timestep selection). When one adds multiple physics models, perhaps in multiple loosely linked modules or separate codes, timestep control across the aggregate application can become very involved. For example, in fission reactor fuel performance simulations, introduced in Section 2.1.2, the timescales of a macroscale structural mechanics model and a coupled mesoscale material model differ by over six orders of magnitude (Tonks et al., 2010). The mesoscale timescale cannot be stepped over, since the evolution of the material properties definitely affects the macroscale simulation. Taking picosecond-size timesteps is not practical for a fuel performance application. Thus one may need to use high-order time integration to support larger mesoscale timesteps, along with subcycling the mesoscale physics. In general, robust and accurate approaches to handling coupled problems with this level of time disparity are still being sought.

In a composite application, one might consider that a timestep is composed of three phases: initializing each package/module in preparation for taking the step, taking the timestep, and then finalizing the step. With multiple modules, the second phase might entail appreciable subiteration between modules, data and results transfer, internal model state updates, and so forth. The third phase includes checking result consistency across the modules, convergence behavior, error metrics, and module intercommunication. The "failure" of a timestep might indicate the inability of subiteration between closely linked modules to converge, or some other error condition. In this case, logic would exist to reduce the timestep or otherwise recover from the error and retake the step. Related algorithmic issues are discussed in Section 3.2.2.

Unfortunately, automatic timestep selection is not robust for general multiphysics simulations and requires tuning for each problem class of interest. Analyst intuition and experience lead to a good understanding of the dynamical timescale of the problem of interest and are valuable in formulating effective approaches and "tuning" them for the application. This observation leads to one of the more effective timestep selection approaches. It is often effective to provide a general interface to a multiphysics code for the manual specification of a timestep history curve that the code should follow, in conjunction with providing residual information as postexecution feedback to the analyst. The analysis can then tune the timestep strategy to provide the desired accuracy and overall simulation behavior, given a few trial simulations.

4.2.5 Achieving performance. While performance is often a major issue in computational science codes, few applications have a quantitative estimate of the achievable performance; hence, they do not know whether the code is using the computing system efficiently. Gaining this understanding is important in judging the impact of different implementation choices and, in particular, the performance trade-offs with different approaches to connecting multiphysics components together.

Achieving this level of understanding requires constructing a performance model good enough to provide an upper bound on performance. Such models are often a combination of instruction, memory reference, and floating-point performance and can identify the critical performance-limiting resource (e.g., Anderson et al., 1999). The "roofline" model, which simply looks at the performance limits due to a number of constraints (each contributing a line to the "roofline"), is a recent representation of this idea.

4.2.6 Software engineering issues for multiphysics integration. Previous sections have discussed some design issues for introducing new models and algorithms into multiphysics applications. In preparing to incorporate an application into a multiphysics environment, a number of issues must be addressed; the following simple suggestions can ease the integration process:

- *Program startup:* A general multiphysics library must not make assumptions about startup. It cannot declare the main routine, as this will most likely be declared by the driver or coupling framework. Applications expected to be used in multiphysics coupling should be written as a library; a stand-alone version of the application can simply wrap a driver (containing main) around the library. In addition, applications should not expect to be able to directly parse input from the command line; multiple codes might interpret the same command-line argument for different purposes. Each application code should define an interface for accepting specific input parameters. The coordinating software (framework or library) can parse the command line and hand off specific data to each application.
- *MPI communicators:* Each application should accept an MPI communicator during startup through an interface. An application code should never directly code to MPI_COMM_WORLD. At startup a common multiphysics parallel decomposition pattern is to hand off blocks of processes (i.e. a subcommunicator) to each application. Creating a communicator on the fly by using MPI_COMM_WORLD inside an application will not work and will hang the coupled program.
- *Global scoping:* The naming of macros, variables, and functions in a code must be protected either by namespacing or by prefixing. In our experience, almost every code that did not take care to apply such protections and polluted the global namespace required extensive refactoring because of collisions with other applications.
- *Third-party library dependencies:* Source code for third-party libraries should not be included in an application. If multiple coupled applications required the same third-party library and if the library were included with each, multiple instantiations would be present. While some compilers can link in this circumstance, others fail with an error if duplicates are present. Additional complexity arises if coupled application codes require different versions of the same library.
- *Output support:* All output for an application code should be allowed to be redirected to its own ostreams/files. If two coupled applications are running concurrently and both dump output to the screen, the output from the applications will be mixed together, making the output unreadable.
- *Separation of algorithms from application physics:* When preparing an application for use in a multiphysics context, a clear division should be made between the software components responsible for discretization of a physical model and the analysis being performed with the discrete model; Section 4.2.1 discusses further details.
- *Level of granularity:* The extent to which data is exposed to the coupling framework will affect the choice in solution algorithms that can be applied to the coupled system; see Section 3.1.2. A code that cannot provide a residual or even expose a solution vector will have to be run as a black box, limiting couplings to nonlinear block Gauss–Seidel (see Algorithm 1) or nonlinear block Jacobi. If a residual can be supplied, then JFNK methods can be employed (see Algorithm 3). If sensitivity information can be supplied, various preconditioning strategies become available. Viable coupling strategies can be severely restricted by the design of application components.
- *Memory management:* Low-level memory management and the handling of pointers are critical to robust multiphysics coupled codes. As data is transferred between applications, strong memory-management tools such as reference-counted smart pointers (Alexandrescu, 2001) will reduce memory issues and debugging efforts.
- *Error handling:* Safe error-handling policies are needed for a robust coupling capability. If components are aborting on certain threads/processes, other threads/processes of the coupled application may hang. Strong error-handling strategies and safe ways to shut down individual application components are needed. Exception-safe code is critical.

4.3 Difficulties in collaborative multiphysics software. Building on the previous discussion of challenges in multiphysics software design, we now address practical difficulties in collaborative multiphysics software.

4.3.1 Maintenance of a stable and efficient development environment. Managing a multiphysics capability composed of individual application codes from multiple institutions places a number of additional hurdles on coupling infrastructure and application developers. A central issue is how to maintain a stable and efficient development environment in the face of distributed development teams, who may be working under intellectual property, export control, and cybersecurity constraints.

Since individual application codes are under active development at various institutions, procedures must be in place for synchronizing changes to the multiphysics infrastructure. A common procedure is to wait for official releases of each application and then incorporate the changes into the multiphysics infrastructure. Depending on how frequently releases are made, updates based on application release cycles can allow for large windows over which bugs can be introduced and compound. This circumstance makes updating applications difficult and time-consuming, since developers must track down when and how the codes have changed.

A more recent software engineering approach is to use continuous integration (CI), where an automated system continuously checks for "commits" to individual application codes. When a new commit is found, the code is automatically downloaded and tested against the coupling infrastructure. This approach allows an automatic documentation as to which commit breaks a code. Automated emails can be sent to the developer owning the commit. Many bugs can be caught and corrected the day they are

introduced, leading to a much more efficient and stable development environment. Such a system requires a rigorous testing system as well as a disciplined development team willing to maintain 100% passing tests so that new failures are not masked by earlier failures.

An advantage of using an automated server for CI is in handling complexities associated with intellectual property and export control. For example, application developers may not be allowed to access the complete coupled code base, yet changes by any application developer may impact the coupled code base. An automated testing system is the first line of defense for notifying developers that their change broke a downstream dependency in code that they cannot access. Cybersecurity concerns can be addressed by locating the CI server outside firewalls.

A recent success story for the CI model is the CASL Nuclear Energy Hub (Kothe et al., 2012). The consortium is composed of 10 core institutions plus an additional 11 contributing partners, each developing their own application codes that are unified under a virtual reactor environment. Because of intellectual property constraints, application developers across the institutions are not allowed access to all application modules. Instead, a three-tiered approach is taken to protect the code base. The first line of defense is a precommit testing script that runs a fast set of unit tests on all codes to which a developer has access. Once this local testing is complete, the commits are pushed to the appropriate institutional repositories. A CI server for the multiphysics infrastructure polls the institutional application repositories every few minutes for changes and will launch a comprehensive test suite that includes all CASL partner codes whenever changes are detected. Results are submitted to a dashboard, and developers are notified by email if their commit introduced a failure. The execution time for tests run for precommit and CI testing is strictly limited so that users can obtain quick feedback that does not impede development efforts. The final tier of defense is a more comprehensive nightly test suite consisting of all precommit/CI tests plus any tests whose execution time is longer than appropriate for CI testing. This is run overnight and takes on the order of hours to complete. This testing can include features such as memory-leak testing (see e.g., valgrind). The procedures developed for CASL are an extension of the Trilinos build and test system.

4.3.2 Investment in software tools and techniques. A significant issue that is exacerbated by multiphysics coupling is the lack of investment in software engineering tools and strong software quality practices across application development teams. The use of software engineering tools (integrated development environments, version control, compilers, build/test and reporting tools) varies widely across institutions and research groups. Multiphysics additionally burdens the performance of the tools because of scaling from a single application to multiple, linked applications. For example, dependency tracking can cause unacceptable configuration and build times if n^2 complexities exist in the configuration system. Such issues might not be noticed in a single application where a smaller number of files and/or dependencies are present.

The lack of adoption of good software engineering tools impacts the development cycle of applications. One issue is that the tool might not be portable to all platforms of interest. Another issue is that a tool may not be funded or supported in the future, leaving users in a difficult position. Investments in open-source software engineering tools should be a priority in order to support HPC applications. An example of a successful project along these lines is the CMake build system (Kitware, 2011). Tools like this are critical, not only for single applications, but especially for multiphysics coupled applications.

Poor software engineering practices can make integration of application components difficult. Some applications may have been developed to run only on a specific system or with a specific compiler with non-standard conforming extensions. Porting to new platforms can be difficult. Memory management, error handling, and global scoping (see Section 4.2.6) can all be controlled with proper software coding guidelines and toolsets (see e.g., Bartlett, 2010).

Multiphysics coupling invariably requires changes to each application code. The stronger the testing in place, the easier refactoring is. Fragile code bases with minimal testing often result in ''code stagnation'' as developers are wary of making the large changes necessary for integration. The adoption of test-driven development and agile development strategies can address these issues.

4.3.3 Adoption of third-party software. Despite the many advantages of code reuse and success stories of the sort presented in Section 4.1.3, some application developers resist the adoption of third-party software. As introduced in Section 2.2.2, there are technical, programmatic, and personal reasons for this resistance. One is the risk, or perceived risk, of not being in control of the software that is being leveraged, or its long-term availability. Concerns include the long-term maintenance of the third-party software, the viability of the organization producing it, and the ability to influence the development of the software to enhance or maintain its relevance to the application being developed. Concerns may also exist about the ''return on investment'' of time spent in learning the software; it is not always clear that the time invested in learning the third-party tools will be recouped by increased development efficiency and by reuse in the longer term. Along these lines are concerns about the ''opportunity cost'', or perceived cost, of using such tools. Deciding on a particular third-party capability may be hampered by concerns about making a poor selection, particularly if multiple third-party tools appear to meet the user's requirements. The difficulty in comparing and assessing complex, high-consequence products can result in decision paralysis.

Another concern is the portability of third-party software. The software must span the hardware of interest to

the user and must be both supported and tested on these platforms as part of the long-term maintenance strategy.

Another practical consideration is the quality of the third-party software. What processes will be used by the user and supplier to verify and validate the code base being leveraged? If the user makes changes to the third-party software, how do these changes get propagated back to the main repository? How are intellectual-property and export-control issues addressed?

Emotional considerations may also be a factor, such as the resistance to use software that is "not invented here." Members of the user's development team might feel security in developing their internal code and insecurity in using third-party capability. Indeed, a developer's self-worth may become intertwined with the application over time.

4.3.4 Licensing. Additional impediments to using common multiphysics software are (1) the cost of commercial software, especially on machines with large numbers of nodes, and (2) unclear or unacceptable licensing terms for some open-source software. Publicly available open-source software is particularly attractive, and it is consistent with the long tradition within the HPC community. Not only does open-source software remove a cost and inconvenience barrier, but it also allows users to determine exactly what the software does, promotes evolution according to user needs, encourages porting to many platforms, and ensures the long-term availability of widely used software. It is important to use well-known licenses such as the Berkeley Software Distribution (BSD), which confer broad rights to users. Licenses such as the GNU General Public License (GPL), which restrict what users can do with code they have written, are unacceptable to many users, and unfamiliar licenses, regardless of terms, often require lengthy legal consultations at commercial companies and government laboratories.

5 Opportunities in multiphysics simulation research

The preceding three sections have highlighted representative multiphysics applications, analyzed the structure of typical multiphysics systems algebraically, and explored implications for their implementation in software. Here we present a number of opportunities arising from these discussions that are ripe for research or development. Some of these opportunities call primarily for leveraging known concepts in new contexts. Some await breakthroughs. The prospect of predicting the behavior of complex systems combining multiple physical phenomena is one of the main motivations for extreme computing (Brown et al., 2010; Rosner et al., 2010), so adapting the uniphysics components to architectural and scaling stresses is paramount. Since the number of such components may often be just two and rarely exceeds three to five in practice, issues of scaling flops, bytes, or transfers due to the multiphysics combination introduce only a modest factor beyond the orders of magnitude often called for in scaling up the components. However, difficulties still lurk in the combination. We recall the admonition of physicist A. S. Eddington (2012):

We often think that when we have completed our study of one we know all about two, because "two" is "one and one." We forget that we still have to make a study of "and."

After reviewing issues in forthcoming extreme computing environments in Section 5.1, we discuss in Section 5.2 concepts that are ripe for leveraging in multiphysics contexts, followed in Section 5.3 by broader multiphysics opportunities that require new breakthroughs in research and development.

5.1 Exascale issues

For the most challenging multiphysics simulations, systems capable of sustaining an exaflop (10^{18} floating-point operations per second) or more will be necessary. But exascale systems are unlikely to look like current systems, even petascale systems. Very roughly, the power consumption of current peak petascale systems (meaning a system with a peak performance of roughly a petaflop) is already around 7–10 MW. At an average cost of US$1 million per MW-year, just providing power for current leadership-scale systems is a major part of the operating cost. For a sustained exascale system, computations must be about three orders of magnitude more efficient to keep the power bill alone under US$100 million per year. A more detailed analysis shows that data motion, whether it is between processors and local memory or between different processors, is a major contributor to the power cost. Consequently, successful programs on exascale machines may need to limit memory motion; current multiphysics approaches that move data representations between different physics components may not be feasible on exascale systems (Dongarra et al., 2011; Keyes, 2011).

Power consumption is also limiting the clock speeds of individual processor cores; in fact, since 2006, processor clock speeds have been relatively stable and are not expected to increase much in the coming years. Consequently, higher performance will require much greater degrees of parallelism. An exascale system may have 10^9 cores, each of which runs at 1–2 GHz. Further, current dynamic random access memory (DRAM), or main memory, is a major source of both cost and power consumption. Without a change in technology, an exascale system is likely to have far less memory per core or per flop than is common today, where the rule of thumb is about 1B per flop. A likely ratio for an exascale system is more likely to be around $0.01 - 0.1$ B per flop. Adapting to this will require methods that are simultaneously more memory-efficient (e.g., higher-order, compact schemes) and retain massive amounts of concurrency (easiest in lower-order, non-compact schemes).

Furthermore, there is the issue of fault resilience. Some predictions about the likelihood of faults, based on the

increasing number of processors cores, significantly overpredicted the failure rate. The reason is that faults are related more to the number of connections or parts than to the number of transistors. Thus, it has been possible to keep the failure rate of systems designed as massively parallel computers low enough that simple checkpointing strategies have been sufficient for applications. This may no longer be possible in an exascale system. While it may be possible to keep the number of parts roughly the same, other issues, such as approaches to reduce power consumption and increase memory size, may raise the probability of faults. Thus, applications may need to take a more active role in being resilient to faults (a general solution is possible but would consume extra power and may not be feasible).

5.2 Concepts ripe for leveraging

We describe some concepts that seem ripe for leveraging from existing applications and frameworks in the campaign to scale up uniphysics codes and to combine them.

5.2.1 Architecturally optimized, distributed, hierarchical data structures from solver packages. In coding for extreme architectures, there is much to learn from existing solver packages. Open-source solvers such as hypre (Falgout et al., 2011), PETSc (Balay et al., 2012), ScaLAPACK (Blackford et al., 1997), and Trilinos (Heroux et al., 2005) appear in inner loops in a vast variety of applications, have received attention from the global community for years, and are among the earliest pieces of software to explore and be ported to new architectures. They have benefited from autotuning in their use of the memory hierarchy and multicore structure, and from sophisticated manipulation of trade-offs in data transfers and in the use of local and global representations to satisfy the demands of working at different points on the performance/programmability spectrum. Examples of such issues include cache and register blocking, loop unrolling and jamming, coloring, message aggregation, and communication–computation overlap. Some use the data structure implementations in such libraries directly for customized ends; others call the solvers built upon them at a higher level. In high-end multiphysics applications if each component controls a separate set of data structures, there may be little chance for strong performance at extreme scales, because of the overwhelming electrical power and execution-time costs of transfers in the inner loops between different data structures. On the other hand, as discussed in Section 4.2, there is a benefit from separation of concerns: if the solver is intermingled with the physics, it is difficult for innovations in either domain to be introduced. A fairly commonly used interface between the physics and the solver is a call-back subroutine such as PETSc's FormFunction. Through this interface, PETSc provides state data, u, on a local data structure (submesh or sublist) that includes data owned by the local process and data gathered from interacting processes needed for the current step. The user locally evaluates the local piece of the residual or tendency function vector, $F(u)$, and returns it to the solver. The user code within FormFunction may be legacy or may be rewritten or generated automatically from an abstract problem description language for optimal cache and register use.

5.2.2 High-accuracy interpolation. Multiphysics codes often require interpolation between continua represented on non-conforming meshes at interfaces or throughout bulk regions, or between different formulations, such as particles and fields. High-accuracy multivariate interpolation is a well-studied problem, though high-performance implementations that make use of existing data structures without creating extensive intermediate working sets may be too much to ask for in library form. Nevertheless, multiphysics applications should not be limited by interpolation errors. In cases where the discrete data structures are solution-adaptive or track a dynamically evolving interface, it may be necessary to preserve the underlying geometry of interface and query it directly to adapt the discretization.

5.2.3 Multiphysics as a form of adaptation. Multiple physical models often arise as a form of solution adaptivity, for instance, in atomistic–continuum models of crack propagation as introduced in Section 2.1.4, in which a continuum description that would possess a mathematical singularity at the crack tip is replaced with a more first-principles atomistic description. Such models may fit the classic design paradigm of local nested h-type mesh refinement, as an abstraction. Instead of refining the mesh in a subregion, one refines the model. Decades of experience with AMR frameworks such as BoxLib (Bell et al., 2012) may be useful in this context. They possess built-in features such as timestep control, subcycling, and re-fluxing that may be leverageable in far more general contexts; see Section 3.2.2.

5.2.4 Applications of classical numerical analysis. Multiphysics coupling of components that are individually high-order in time are formally limited to first-order in time for standard splittings into two sequential half-steps, though formal second-order temporal accuracy may be achieved in a two-component problem by Strang splitting (Strang, 1968), which is equivalent to reading out the states at the half-step of one of the component steps, or at a quarter-step overall. First- or second-order splitting can reduce the order of accuracy of the multiphysics integration below that of the components; and unless the componentwise operators satisfy various commuting relationships, which would be rare except in certain space-dimensional splittings in linear constant coefficient problems, this would be the end of the road. One solution is to use Richardson extrapolation in time. Another approach is to use SDC (see Section 3.2.2). A third option is to go fully implicit with all of the components.

Often, slow convergence in a multiphysics iteration is due to the degree of coupling. In a loosely coupled solution

procedure, numerical effort to reduce the residual in one physics component may lead to an increase in the residual of other components. Finding ways to improve convergence behavior, short of going fully implicit, can reduce this inefficiency.

If going fully implicit, the coupled system may possess linear stiffness and/or "nonlinear stiffness" that is much greater than any of the individual components. Ordinary, nonlinear globalization strategies, such as backtracking, are designed to prevent stepwise increase in the residual; however, this strategy fails when increases in the norm of the nonlinear residual function need to rise in order to pass from one local minimum to another. Different globalization criteria, such as pseudo-transient continuation (Kelley and Keyes, 1998), may be needed for cases where simple globalizations fail.

5.2.5 Exploitation of additional concurrency. Hardware developed in the campaign for extreme computing has made flops relatively cheap, and memory and memory bandwidth more scarce than ever, while vastly expanding concurrency. This may have enormous implications for multiphysics simulations. Time-parallel methods, for example parareal (Lions et al., 2001), are probably contraindicated as a source of additional concurrency, since they store many additional copies of the state vector a different time levels (essentially one new copy of global state per processor/thread engaged in the temporal direction). Additional concurrently available flops may encourage physics and solver modules that store many fewer intermediate common subexpressions and compute more on the fly.

Arithmetic intensity (roughly the ratio of flops to loads/stores required to execute an algorithm) is boosted for formulations that increase the interaction between elements of a given state at each timestep. This is typically the case for implicit methods, which may also be employed to reduce the number of timesteps relative to explicit methods in many systems, where stability, and not accuracy, sets the timestep limits. Arithmetic intensity may also be boosted for formulations that increase state (which has detrimental storage implications) but increase useful flops faster: superlinearly in the size of the state. This may be true of many of the advanced formulations of exascale simulations, in which the additional "state" is actually auxiliary to the physical state, such as sensitivities in uncertain quantification problems or adjoints in optimization problems. Stochastic error analysis is perhaps newly affordable.

5.2.6 Exploitation of "tricks" for implicit algorithms. Computations that are implicit often use a JFNK algorithm (see Section 3.1.2), which avoids formulation and storage of a Jacobian matrix (which can otherwise contribute to the dominant-sized and least reusable object in the solver) and relies on evaluation of many residual functions, $F(u)$, for different states u presented by the Krylov method, as given by equation (9). Normally, a second evaluation at a perturbed argument $F(u + i\sigma v)$ is necessary to estimate by finite differences $F'(u)v$, in order to generate the next Krylov vector. It is, however, possible for many functions F to evaluate $F(u)$ and $F'(u)v$ simultaneously from real and imaginary parts of an overloaded complex-valued residual evaluation. Implicit capability can also be employed to capture stationary periodic behavior, for which we expect $u(t+T) = u(t)$, in any transient model with a black-box propagator, $u(t+\tau) = P(u(t), \tau)$. One can apply Newton's method directly to the fixed point $F(u) \equiv u - P(u, T) = 0$.

5.2.7 Provision of special hardware. The extra transistors on leading-edge chips may be dedicated to functions that are not used often but significantly reduce the latency of other operations they replace in hardware. One example of such a "luxury" may be random-number hardware, which would be useful for multiphysics models that include stochastic physics components or stochastic forms of UQ.

5.2.8 Software maintenance best practices. Multiphysics simulation codes can be tested and maintained with reference to the tests and maintenance practices of the uniphysics component codes; however, as discussed in Section 4, there may be subtleties. Manufactured solutions used to verify the components, which assume that $\partial F_1/\partial u_2 = 0$ and $\partial F_2/\partial u_1 = 0$, should be generalized to take into account the variation of the cross-components. A regression test suite for the multiphysics problem should hierarchically combine the uniphysics regression tests, again to take into account the new cross-component dependencies in the flow from inputs to outputs.

5.3 Required breakthroughs

In contrast to the opportunities above, which should be straightforward developments of existing techniques in new contexts, we describe here some concepts that seem to require new breakthroughs in mathematical understanding, algorithmic development, or software design and implementation.

5.3.1 Formulation of objective functions useful for optimizing simulations. Excluding the human cost of programming and operating supercomputers, general figures of merit in assessing the impact and efficiency of simulations may often take the form of "scientific impact per unit energy" or "scientific impact per unit time." Scientific impact is obviously hard to measure in absolute terms but may be represented by a proxy such as accuracy. If the goal of the extreme-scale simulation is beyond the domain of original insight and is related to the quality of prediction, then "accuracy per joule" is a reasonable desideratum. Like the Drake equation of astronomy, this estimate can be factored into a number of ratios coming from more basic assumptions, in this case from numerical analysis and computer architecture. For instance: accuracy per stencil

evaluation, stencil evaluations per flop, and flops per joule, which can be subfactored in turn, given problem- and machine-specific details. Each of these factors depends on myriad assumptions about smoothness, convergence rate, resources of the memory system, and so forth. As difficult and subjective as it may be to define a good objective function to assess scientific impact and efficiency, we cannot hope to optimize it without quantifying it.

5.3.2 Characterization of uncertainty. Each component in a multiphysics application needs to have an uncertainty characterization of its own in quantities of ultimate interest. These may not be the primitive variables of the simulation but functionals of components of the state vector, such as the integral over a surface of a normal derivative of a field representing its flux. Uncertainties that depend on quantities coming from other components, in the form of boundary conditions, interior forcing terms, constituent laws, and so forth, must also be characterized. This characterization may be particularly interesting for quantities of different types such as particles and fields. The best ways of quantifying uncertainty for simulations of multiphysics processes whose models have nondeterministic aspects (e.g., bubble formation in foams, surface tension effects, and cracks) remain to be seen. In general (see e.g., Section 3.5), one cannot ignore cross-component effects in quantifying uncertainty, since uncertainties propagate across couplings, and there may be cyclical effects. Consequently, multiphysics applications will need to have the ability to sample from the output of individual physics components, and UQ will need to become more intrusive.

5.3.3 Equidistribution and control of error. Mature uniphysics component codes come with error-control mechanisms, such as variable-order, variable-size timestepping; *h*-, *p*-, and *r*-type spatial refinement; particle generation and removal; and model substitutions, which respond to error estimators, typically of a posteriori type (see e.g., Section 3.4). However, multiphysics applications must evaluate the cost and benefit of refinement strategies in a much broader context. It is possible to be deceived into spending nearly all of the execution resources on a component whose accuracy is already much better than the limiting accuracy of the system, which could be due to a different component, or at some interface between components. Context-appropriate premiums must be placed on gross checks like integral conservation, and consistency of the representation of fluxes at boundaries. Of course, error requirements may sometimes be exceeded when parameters are chosen with respect to some other criterion, like stability, which may be more stringent for a coupled system than for any of its components. These thresholds of limiting parameters should be identified because they could potentially trigger dynamic adaptations of the algorithms.

5.3.4 Dynamic characterization of cross-physics interaction strength. Though cross-coupling enters into performance efficiency considerations in subtle ways like arithmetic intensity, which may be better for a coupled system than for its components, making the coupling effectively "free," efficient multiphysics algorithms will preferably allocate the extra work of enforcing tight coupling only where the interaction is strong and will default to loose coupling where the interaction is weak, or sequential where it is unidirectional. Algorithmic selection, which may be dynamic over the duration of the computation, therefore requires dimensionless or relative measures of cross-variable sensitivities. Just as error estimators are required to inform adaptive discretizations, some measure of interaction strengths between the components is required to inform adaptive coupling tightness.

5.3.5 Reduction of data transfers. Multiphysics simulation, as an extreme-scale simulation, will generally execute on hardware on which the energy costs of data transfers, between processor-memory units or within the memory hierarchy of a single unit, is at a premium. Many new programming paradigms are under consideration for such hardware, which represents the end of a successful era of correctness-portable, relatively performance-portable, MPI-based single instruction multiple data (SIMD) code design. There is discussion of paradigms such as "moving data to code, rather than moving code to data," even though this cannot be perfectly realized for applications with all-to-all or even one-to-all data dependencies (like nearly all physical world models). Whatever successful paradigms are developed for uniphysics problems may be vulnerable to a high energy cost for transfers if the uniphysics codes are combined in a black-box manner, since this normally requires the repackaging and redistribution of data. We expect that significant efforts in codesign will be required in order to keep data transfer costs manageable in the multiphysics applications for which extreme computing environments are built.

5.3.6 Relaxation of synchrony. Tomorrow's extreme-scale computing systems will have less performance reliability than do their contemporary counterparts for a number of reasons well documented in the International Exascale Software Roadmap (Dongarra et al., 2011). Energy-efficient, low-voltage differentials between logical "0" and "1", for instance, may mean that some arithmetic operations may be corrupted by noise and need to be redone. Because of the decreasing performance reliability, it will not be possible to load balance with sufficiently high precision for today's SIMD bulk-synchronous algorithms to run efficiently. Less-synchronous algorithms with work stealing will need to be developed within components and across components. Today we synchronize artifactually with too heavy a reliance on nested loop structures, and far more frequently than is required for physically causally correct computations. In many places, it should not be difficult to beat back to causal limits. However, asynchronous

algorithms typically compromise robustness relative to strongly synchronous counterparts. Both programming paradigms and mathematics will need to be revisited. For instance, global MPI_COMM_WORLD will be replaced with local communicators with the smallest required scope to allow greater independence of threads, allowing different physics components to complement each other in cycle and resource scavenging without interference. Algorithmically, data that is causally required but unavailable may often be predicted and the computations dependent on such data rolled back if, when the data becomes available, it is determined that the output would be too sensitive a function of the discrepancy between the actual and the predicted data. In practice, low-frequency components of the predicted data may be accurate, and high-frequency discrepancies may be demonstrably irrelevant because they are rapidly decaying. There are numerous such opportunities for the relaxation of artifactual synchronization or too strict an observance of bona fide synchronization, both within and between components.

5.3.7 User-specified checkpointing. At a level of unreliability beyond individual operations, executing long-running multiphysics codes on extreme-scale computing systems with many points of potential hardware and software failure will need user-specified checkpointing. Operating-system checkpointing must be conservative and save complete program state, without appreciation for the fact that much of the bulk of the computational state can be reproduced from a much smaller physical state. For example, a Jacobian matrix does not need to be saved for restart since it can be computed from a (typically much smaller) current state vector. Data related to multiphysics coupling may be reproducible from the union of the uniphysics states. There is opportunity to specify minimum-size checkpoints for reproducible restarts as a union of checkpoints from individual physics packages so that the overall simulation is robust. This may also be good software discipline for debugging complex multiphysics codes on contemporary machines.

5.3.8 Combining continuum-based and discrete phenomena. Many multiphysics problems span scales from atomistic (or other discrete) phenomena to the continuum. The coupling of discrete and continuous formulations transcends all scales of computation and remains a theoretical challenge; see Section 3.3. To perform judiciously truncated computations in either the discrete or continuum-based world requires quantification of their associated errors and the errors of coupling so that errors can be "equidistributed" and no one phase, including the coupling, is driven too hard. There are several opportunities to be investigated here, including how best to transfer state, fluxes, and forces between the formulations, how to estimate and control errors incurred in the transfers, and how to develop a unified formulation, perhaps involving penalties or discrete conservation principles.

5.3.9 Multiphysics software design. The time is ripe for the multiphysics community to extend the successful software approaches introduced in Section 4, as well as to leverage the features of emerging computer architectures, so that new algorithms and approaches can be explored. Innovative next-generation multiphysics software has the power to transform computational science, with a goal of enabling fully predictive simulations.

5.3.10 Multiphysics performance models. The performance of many computational science applications has been dominated for many years by the cost of moving data rather than raw floating-point rates. For a large multiphysics application, data typically needs to be moved between components; it may also need to be reorganized if the data structure details are different. These factors raise numerous questions:

- What is the cost of moving data between components? If the cost is high, are there approaches that can mitigate the overhead, such as multistage communications or other communication-minimizing or avoiding techniques?
- What is the overhead of translating between representations? Is it more efficient to translate or to select a common representation (which may not be optimal for any individual component but is optimal for the overall simulation)?
- In a parallel program, what is the impact on process placement? For example, should the different components be laid out in compact sections on the parallel system or intermingled to reduce communication costs? Is dynamic load balancing practical?
- What is the overhead of using library routines (particularly a sequence of routines) over using custom code?

Relatively simple performance models can be developed to address these questions. For example, models that determine the number of compulsory cache misses can be used to determine an upper bound on performance and are still sensitive to the parameters of the system's hardware. Similarly, these performance models can be used to help design and optimize the use of hybrid programming models, such as MPI with OpenMP.

6 Insertion paths for algorithms and software into multiphysics applications

As many of the challenges presented in the previous section are tackled, new algorithms and software designs will be available for use in existing multiphysics applications. The next set of challenges will lie in how to incorporate these new technologies into simulations in ways that will minimize the implementation time and maximize their effectiveness. The issues that arise involve both technical and non-technical areas, and we examine considerations from

both of these viewpoints. We also present ideas for successfully incorporating new technologies into multiphysics application areas.

6.1 Mechanisms

Before looking at factors that affect insertion of new technologies, we outline three common mechanisms for such insertions.

6.1.1 Leveraging of interfaces. One successful insertion path is that of engineering current software to leverage interfaces. Software infrastructure that encapsulates functionality in terms of well-defined interfaces is able to easily adopt new technologies by swapping capabilities behind those interfaces; see Section 4.2. This design allows a new technology to ''look'' like the old from the point of view of the application. In this framework, direct comparisons between the old and new technologies can easily be made. In addition, well-defined interfaces often allow the adoption of external libraries, in turn providing the ability to investigate the performance of many methods before choosing a final one for use. Encapsulation of functionality behind good interfaces provides an effective way to transition between technologies that adhere to those interfaces. As a result, more options are available to a given application.

We note that well-defined interfaces can sometimes prohibit extreme performance tuning. For mature technologies, where one technology shows a clear benefit, the code framework may be more closely tied to the technology implementation in order to benefit more from efficiency gains. However, the trade-off often means that interfacing with any further new technologies is much more difficult.

6.1.2 Small side-effort. In many cases, a transition to a new technology starts with a small side-effort. Often this activity is a splinter from a main, funded effort and is conducted by a student or junior staff member who can devote significant time to the project. Within this effort new technologies can be tried in a specific branch or simplified code. If this proof-of-principle effort proves successful, the technology can then be moved into the main code. The principal risk with this mechanism is that since junior staff (typically, temporary or shorter-term) are often the ones conducting most of the work, the people who are most knowledgeable about the new technology may not be present after the initial testing and use.

6.1.3 Full rewrites. An effective way to port new technologies into multiphysics applications is to fully redesign the software considering the demands of the existing multiphysics setting; unfortunately, this is also the most expensive in terms of time required to begin conducting simulations. Additionally, there is no guarantee that for the present project a full rewrite will produce significantly better results than will reusing the existing code. Also, with a full rewrite, it will likely take longer to begin initial simulations.

Even so, fully rewriting software is the best approach if (1) the existing software is out of date by today's software engineering standards, (2) the mechanism for coupling this software to the multiphysics application is fragile, and (3) someone is willing to pay for a full rewrite. By totally redesigning the technology with a multiphysics mindset, coupling and scalability issues related to data traffic can be addressed effectively. Also, the software can be designed extensibly to take advantage of the most modern algorithms and libraries, as well as to prepare for future evolutions; this approach will help delay any future rewrites and encourage incorporation of new technologies across the community by demonstrating a successful implementation. The value of this approach is in the long-term viability of the multiphysics application achieved by designing it to adapt to existing and coming software practices while still leveraging the knowledge present in the original code. Having the original designer of the technology available is helpful in understanding the underlying mentality of the original design and ensuring that the most effective aspects still appear in the rewrite. We further note that one cannot assume that the new code is as correct as the original; therefore, a full verification is often required when a rewrite takes place.

6.2 Factors affecting successful insertion paths

Successful integration of new technologies into existing multiphysics (and often multidisciplinary) applications depends on adequate funding, a solid collaboration among participants that is well supported by the home institutions, new technologies meeting well-motivated needs, and software engineering that allows for flexibility. These conditions are rarely all present simultaneously. Below we list some key considerations that have given rise to successful integrations of new technology into simulation areas. Our intention is that when all the above conditions are not present, participants can look at these areas as ways to make good progress.

6.2.1 Clear motivation. The most successful integrations of new technologies into existing simulation codes occur when the new technology or enhancement is clearly needed. However, understanding the goals and hence needs of applications research codes can sometimes be difficult. In addition, assessing the current and potential performance of the code and any new technologies is not straightforward.

A typical research code is often used to analyze or simulate multiple and very different physical phenomena over its lifetime. As a result, the goals of such codes are fluid, and identifying specific requirements can be challenging.

When specific goals and needs are identified, however, a new factor comes into play: the determination of whether a new technology can help the simulation code meet a goal. In order to address this question, performance analyses are required. Multiphysics applications are often large and sometimes difficult to manage, making performance

analysis on multiphysics software complex. This complexity, in turn, can make it challenging to quantify the computational goals of multiphysics applications (e.g., memory demands, data traffic, computational speed, order of accuracy). One approach to conducting this analysis is to produce "skeleton codes" that allow for simpler identification and classification of contributing costs. These skeleton codes have the same semantic structure as the complete multiphysics codes but without the full complexity, both computationally and physically.

In order to use this approach, simplified models of the relevant component physics are developed and then included in the skeleton simulation. These skeleton simulations are then profiled, with the goal of identifying the issues that will be present in the full multiphysics simulation and produce clear targets for further progress. These activities invariably help focus the needs of potential new technologies and expose their motivations for use.

6.2.2 Collaboration. One of the most significant impediments to multiphysics applications also produces the most potential for great progress: collaboration between interdisciplinary groups. While the initial implementation of the technology may have been designed only by an application developer, moving to the multiphysics setting, and hence more complex simulations, will likely require the input of software developers to optimize efficiency and the input of mathematicians to study stability and accuracy.

The value of communication goes both ways, though. Math and computer science experts must understand the problems that are involved in the simulation and the assumptions that went into the underlying models. Without this input there is likely no way that full analysis or optimization can be done, and time will be lost while necessary physical assumptions or conditions may be violated during the development. All members of the interdisciplinary development team must understand both their role and the role of the other participants. The most successful collaborations will produce multiphysics simulations that allow for new physical insight obtained with computational efficiency and mathematical accuracy.

Multidisciplinary collaborations also tend to be most successful when each member takes the time to understand the reward structure of other members' institutions and fields. Authorship order, choice of journals, choice of conferences to present the work, and software releases can all be important factors, but of different weight in different disciplines. It behooves team members to understand these and, to the extent possible, help the members meet their institutional requirements.

6.2.3 Discretization and meshes. The development of new multiphysics capabilities for which solution parameters must be related between physics components requires understanding and interacting with the meshes, as well as distribution of the field parameters over the mesh, as indicated by the equation discretization process. The complexity of developing the needed coupling strongly depends on the degree to which the types of meshes, physics discretizations, and data structures used are, or are not, matched. The insertion options associated with various approaches are outlined in Section 4.2.3.

6.2.4 Software packages. The approaches for addressing multiphysics software challenges given in Section 4 apply to both application-specific projects and to software packages. The burden on package developers is greater, however, because the success of a package depends on its integration into other applications.

Good software engineering alone does not address the reasons given in Section 4.3.3 for resistance to adopting external packages. The developers of software packages need to minimize the real and perceived costs of using their work. Documentation has always been a necessity; and as projects become more complex, that documentation needs to be not just thorough but also well structured. Equally important are examples of the intended use of the package. Examples, particularly multiphysics examples, can demonstrate relevance, give application developers an idea of how to match their codes to the new technology, and drive home the advantages that new technology has over what it is intended to replace.

The chances of an external package being successfully integrated into an application increase if its developers are able to provide support during its use. The developers of multiphysics software often have experience with more varied HPC resources than do their users; hence, their help with migrating to new platforms, troubleshooting, and profiling can be extremely valuable. Support should also include a willingness to adapt software packages to the needs of applications. Just as applications should adopt software packages with new technologies to stay competitive, software packages should incorporate the feature requests of applications to stay relevant.

6.2.5 Funding. Funding is a key element of the successful integration of new technologies. Often, funding in scientific applications areas is provided for use of the simulation to discover new science or conduct an analysis of a given phenomenon. Funding agencies tend to be reluctant to fund new code development without new science. On the algorithms side, funding is usually available only for development of novel methods or software libraries containing algorithm implementations done in a general way so as to benefit multiple application areas.

Clear motivation for new technologies often leads to greater chances of success for their integration. It also leads to greater success in funding that integration, because justifications are more readily made.

One reason science applications and their funding agencies give less priority to adopting new mathematical and computer science technologies is that new technologies can be less robust and hence less reliable. While a new technology may present an opportunity for greater speed

or accuracy, the adopting code will need to go through a new verification study. In addition, input files and models may need to be adjusted in order to produce accurate simulations with the new technologies. Even when funding is available for the initial insertion, it is often not available farther down the road for new input files and verifications. Regularly run, detailed verification suites (not solely unit tests) with documented results significantly help in this endeavor. With these suites, new technologies can be verified more quickly, since the full machinery for doing verification with the code is already developed and functioning.

6.2.6 Personnel. Another important consideration for insertion of new technologies is personnel. Often, integration of new technologies requires some software engineering expertise. However, a small- to moderate-sized application effort may not have a dedicated software engineer. Hiring such expertise can be difficult for a short-term position. As a result, scientists often perform large software engineering tasks themselves. However, reward systems at many institutions do not take such work into account. In addition, the work is often not in the scientists' main line of interest. As a result, software engineering performed in such an environment may not take advantage of some best practices, causing possible delays downstream. Furthermore, a scientist may be less able to tune an algorithm or, when an algorithm fails, determine whether there is a bug in the implementation or an algorithm failure for a problem.

On the mathematics and computer science side, a similar trend is often seen. Again, because institutional reward systems value publishing in disciplinary fields, staff are most interested in implementing and developing new methods, not in placing known methods into a scientific code. This interest is in direct conflict with the scientists' need for proven and reliable methods to move the science simulation forward. In addition, while some funding exists to package software for use across applications, often package writers are not funded for longer-term support.

A further hindrance to effective cross-disciplinary work from the software developer side is the issue of recognition. Often, software development is seen as a necessary requirement to move a simulation's capability forward. However, since the main new science is done in modeling and interpreting code results, the software work is not given credit in publications nor in most criteria used for promotions at most institutions. As a result, recruiting for these tasks is difficult, and only rarely have packages and software been supported long-term.

For many of these reasons, implementations often become the task of junior researchers, since they often are the only ones with large enough blocks of time to do complex implementations. These junior staff tend to be less familiar with effective software practices. In addition, junior staff have a high turnover rate and thus cannot be relied on to provide long-term knowledge of and support for the software implementations.

6.3 Observations

We present the following observations on ways that are most likely to ensure success in integrating new technologies into existing multiphysics simulation areas.

1. Ongoing verification and software support activities require adequate scoping, budgeting, and funding from funding agencies and principal investigators.
2. Software development activities aimed at increased simulation speed and accuracy (and not just new capabilities) are important aspects of software development and merit consideration in proposal and publication reviews.
3. New technologies often bring in new models and new numerics. The most successful projects engage experts in validation and verification as well as in UQ.
4. Successful long-term projects engage a software architect who ensures use of best practices and enforces standards.
5. Good meshing technologies enable complex simulations and allow the necessary discretization schemes to facilitate flexibility in algorithms.
6. Compartmentalization of code functionalities leads to extensibility for future technologies. Mesh, visualization, interpolation, solvers, and UQ are areas where high-quality external code can be incorporated to facilitate software coupling.

7 Conclusions

The preceding sections demonstrate a vast variety of approaches to high-performance multiphysics coupling of PDE-based models contributed from the applications, mathematics, and computer science communities in developing the science, the methods, and the computational infrastructure necessary for bringing together complex simulations of multiple physics processes in parallel computing environments. As introduced in Section 1 and further detailed for various applications in Section 2, numerical coupling strategies range from loosely coupled Gauss–Seidel and operator-splitting approaches to tightly coupled Newton-based techniques. Researchers have made substantial progress in understanding coupling issues for multiphysics components in space and time, including aspects of problem formulation, discretization, meshing, multidomain interfaces, interpolation, partitioned time-stepping, operator-specific preconditioning, as well as difficulties that can arise in multiphysics operator decomposition (see Section 3). Building on the experience of teams that have developed application-specific multiphysics software and more general infrastructure in support of parallel PDE-based multiphysics problems, Section 4 identifies software features that facilitate robustness, extensibility, and ease of use, essential to supporting research on multiphysics algorithmic issues, architecture-specific enhancements, and changes over time.

While the computational science community has made significant progress in advancing scientific knowledge and engineering practice through these methods, we have outlined in Sections 5 and 6 numerous challenges that await the focused attention and critical thought necessary for further advancements. Many of these issues have received attention in venues of note. In particular, the report by Brown et al. (2008), integrates input from a broad spectrum of scientific and engineering applications to highlight key challenges from an applied mathematics perspective in the predictive M&S of complex systems. Included are the following two research objectives:

1. To advance the fidelity, predictability, and sophistication of M&S methodologies by developing the mathematical tools needed for the analysis and simulation of complex systems characterized by combinations of multiple length and timescales, multiple processes or components.
2. To develop analytical and computational approaches needed to understand and model the behavior of complex, multiphysics and multiscale phenomena.

These two points underscore the significant and still remaining key need for conducting the mathematical analysis necessary to ensure that coupling schemes are accurate, stable, robust, consistent, and are implemented correctly, as prerequisites for developing predictive multiphysics simulations. Moreover, emerging extreme-scale computational environments require fundamentally rethinking approaches to multiphysics modeling, algorithms, and solvers (Brown et al., 2010; Rosner et al., 2010; Dongarra et al., 2011; Ang et al., 2012), with attention to issues of data motion, data structure conversions, and overall software design.

Clearly, much work remains before we can fully realize the potential of multiphysics simulations on emerging high-performance architectures for high-impact policy and decision support. We expect that over the next decade, such investigations will unify practitioners of multiphysics simulations and advance many areas from the common current practice of "two scientists meet at a conference and decide to combine their codes" to fully integrated, numerically accurate, efficient simulations. It is our hope that by documenting the current state from the complementary perspectives of multiple applications areas, applied mathematics, and computer science, we will enable significant further progress through cross-fertilization of ideas that will lead to a next generation of integrated and accurate multiphysics simulations that enable scientific discovery.

Glossary

A continuing issue on the frontier dividing science and applied mathematics is the cultural barrier of language. The growth of multiphysics exacerbates this problem because it engages scientists and mathematicians from increasingly diverse areas. For multiphysics collaborations to thrive, a common language is needed. This glossary defines terms as used in this report.

- **coupling**
  - **strong (versus weak) coupling of physical models**: We use strong (versus weak) coupling to refer to strong (versus weak) interactions between different physics models that are intrinsic between the different physics in a natural process. A strong coupling could be due to large overlap of geometric domains or due to a strong influence of one model on another. For example, in climate modeling, the coupling between the ocean and sea-ice models is strong, while the coupling between ocean and land models is weak. Mathematically, the off-diagonal block of the Jacobian matrix of a strongly coupled multiphysics model may be nearly full, or it may be sparse but contain relatively larger entries. In contrast, a weakly coupled multiphysics model may contain relatively few or relatively small off-diagonal entries. See Section 2 for further discussion.
  - **tight (versus loose) coupling of numerical models**: We use tight (versus loose) coupling to refer to the algorithmic aspect of multiphysics coupling schemes (or numerical models) in terms of whether the state variables across different models are well synchronized. A tightly coupled scheme (sometimes referred to as a strongly coupled scheme) would keep all the state variables as synchronized as possible across different models at all times, whereas a loosely coupled scheme might allow the state variables to be shifted by one timestep or be staggered by a fraction of timesteps (see Figure 1). For example, a monolithic fluid–structure approach (e.g., Blom, 1998) and a full iterative scheme (e.g., Matthies and Steindorf, 2002) are tightly coupled, whereas a sequential staggered scheme (e.g., Farhat et al., 2006) is loosely coupled. Note that there is not a direct one-to-one correspondence between strong (versus weak) physical coupling and tight (versus loose) numerical coupling, since a tight or a loose numerical coupling scheme may be used for strongly or weakly coupled physical models. See Section 2 for further discussion.
- **error and residual**: Error and residual are related concepts that are sometimes not sufficiently carefully distinguished. Operationally and mathematically their distinctions should be preserved. Given a putative solution, the error is a measure of the distance of that putative solution from the true solution. The residual is a measure of how well the putative solution solves the intended equation. The residual is always computable, whereas the error usually is not. In systems that possess multiple roots or are ill-conditioned, a small or even zero residual does not necessarily imply a small

error. Most iterative methods for the solution of linear or nonlinear systems employ, in an inner loop, a map from a residual to an estimated error. In a system of equations $F(u) = 0$ satisfied by u^*, the residual is $F(u)$ while the error is $e \equiv u - u^*$. When the system is linear, that is, $F(u) = Au - b$, the relationship between residual and error is $r \equiv F(u) = Au - b = Au - Au^* = Ae$. Solving exactly for the error in this case is as difficult as the original problem. Solving approximately may be easy, in which case a convergent iteration is often possible. When the system is nonlinear, a Newton iteration accompanied by an approximate linear solution with the Jacobian matrix maps the nonlinear residual to an estimated error.

- **explicit methods**: In an explicit method, future states are given by a formula involving known computed quantities that can be evaluated directly. The simplest example is forward Euler: $y_{n+1} = y_n + \Delta t F(y_n)$. Explicit methods are inexpensive per iteration because they require a fixed number of function evaluations to complete the step; however, they typically have severe stability restrictions and therefore are suited only for non-stiff problems. See Section 3.2.2.
- **Frameworks and frameworks**: In the context of software, a framework provides a generic functionality (such as solving a system of linear equations) that can be customized by the user to provide an application-specific solution (such as a specific matrix and preconditioner). Frameworks typically handle the flow of control, which distinguishes them from the use of library routines. Some frameworks take complete control of the application; we called these "Frameworks" with a capital "F". Other frameworks assert control only over the task they perform and can be combined in an application with other code or even other frameworks; we called these "frameworks" with a lower-case "f". See Section 4 and also http://en.wikipedia.org/wiki/Software_framework.
- **IMEX methods**: An IMEX method is a partitioned method that uses an implicit integrator on one partition and an explicit integrator on the other. See Section 3.2.2.
- **implicit methods**: A fully implicit method advances the solution in time by using current information and an inversion process with no explicit steps. The simplest example is backward Euler: $y_{n+1} = y_n + \Delta t F(y_{n+1})$. Fully implicit methods are relatively expensive because of the need to solve (non)linear systems at each step; however, they have favorable stability properties (Shampine and Gear, 1979; Higham and Trefethen, 1993). See Section 3.2.2.
- **multiphysics**: Semantically, a multiphysics system consists of more than one component governed by its own principle(s) for evolution or equilibrium, typically conservation or constitutive laws. A major classification in such systems is whether the coupling occurs in the bulk (e.g., through source terms or constitutive relations that are active in the overlapping domains of the individual components) or whether it occurs over an idealized interface that is lower-dimensional or a narrow buffer zone (e.g., through boundary conditions that transmit fluxes, pressures, or displacements). Section 1 discusses further details, including similarities between multiphysics and multiscale, multirate, multilevel, and multimodel problems.
- **multiscale model**: A multiscale model of a physical system finds use when important features and processes occur at multiple and widely varying physical scales within a problem. (These scales are frequently expressed as spatial and/or temporal scales.) In multiscale models, a solution to the behavior of the system as a whole is aided by computing the solution to a series of subproblems within a hierarchy of scales. At each level in the hierarchy, a subproblem concerns itself with a range of the physical domain appropriate to the scale at which it operates. An example of a multiscale model is described in Section 2.1.4, where the development of cracks in solid materials is computed through a successive hierarchy of models: fast-acting, close-range atomic-force calculations on the smallest microscopic scale through to macroscopic continuum models on the largest macroscopic scale.
  - **macroscale**: The largest physical scale within a physical system or field of interest. Where the microscale and mesoscale frequently concern themselves with constituent quanta, such as individual atoms, molecules, or groups of such, the macroscale frequently deals with the continuum and bulk properties of a system. For example, in atmospheric science, the macroscale is at the scale of the largest features within weather or climate modeling, which are on the order of 1000 km; in engineering fields, the macroscale is often given by scale of the entire device of interest, which, depending on the device, can range from nanometers to kilometers.
  - **mesoscale**: A physical scale intermediate to the largest and smallest scales existing within a physical system of interest. Like all physical systems with multiple scales, models at the mesoscale may represent one or more levels of a multiscale model or may represent the scale of a standalone model in its own right. The term "mesoscale modeling" first found use in meteorology for the modeling of severe storm systems on a local scale. Reliable storm forecasts cannot be achieved with standard continental-scale weather models, where grid sizes are typically of a scale larger than the size of the storm being modeled.
  - **microscale**: The smallest physical scale within a physical system or field of interest. Calculations at the microscale frequently treat fundamental system constituents and interactions. Models at the microscale may exist as standalone models or as a level within a multiscale model. Examples of the

microscale in chemistry and materials science usually refer to quantum-mechanical, atomic-scale interactions (on the order of 10 nm). In meteorology, the microscale range is on the order of 1–10^5 cm and treats phenomena such as tornados, thunderstorms, and cloud-scale processes.

- **operator splitting**: Multiphysics operator splitting is a loose coupling scheme that evolves one timestep for each physics component in sequence, as shown by Algorithm 2.
- **Schur complement**: The Schur complement of a square diagonal sub-block of a square matrix is itself a square matrix that represents the original linear system after unknowns corresponding to the complementary non-singular diagonal sub-block have been eliminated. Static condensation of a finite-element stiffness matrix is an instance; here, typically, interior degrees of freedom of an element are eliminated. The concept generalizes algebraically. If a matrix A is partitioned as in equation (5), and A_{II} is invertible, the Schur complement for the Γ block is $S \equiv A_{\Gamma,\Gamma} - A_{\Gamma,I}A_{I,I}^{-1}A_{I,\Gamma}$. Though smaller and, in important classes of problems, better-conditioned, the Schur complement is generally dense, so its solution may be approached iteratively, where each iteration requires in principle a high-precision solution of the block $A_{I,I}$.
- **semi-implicit methods**: Inclusion of both implicit and explicit elements gives rise to semi-implicit methods. See Section 3.2.2.
- **subcycling**: Given two time-dependent variables w_1 and w_2, assume their time evolution is coupled between time levels t^{n-1} and t^n. Let $u_i \approx w_i$ for $i = 1, 2$ be decoupled approximations satisfying

$$\partial_t u_1 + f_1(u_1, u_2(t_{n-1})) = 0$$

and

$$\partial_t u_2 + f_2(u_1(t_{n-1}), u_2) = 0$$

(Note that Algorithm 2 may be written in this form by redefining f_2.) It is said that u_2 is subcycled M times relative to u_1 if $u_2(t_n)$ is approximated by any timestepping method requiring M timesteps to evolve from t_{n-1} to t_n.

Acknowledgments

This report is the outcome of a workshop on *Multiphysics Simulations: Challenges and Opportunities*, sponsored by the Institute of Computing in Science (ICiS) (Stevens, 2011). We gratefully acknowledge the ICiS for workshop support, and we thank all ICiS staff, especially Jutta Strate-Meyer and Cheryl Zidel. Additional information about the workshop is available via https://sites.google.com/site/icismultiphysics2011/, including relevant reading, presentations on multiphysics issues in applications, algorithms, and software, and the start of an illustrative multiphysics problem suite. We are especially grateful to Dana Knoll, Juan Meza, and Linda Petzold for detailed and insightful comments as reviewers. Their advice greatly improved the precision and perspective of this manuscript. We thank Satish Balay for establishing a Mercurial repository to facilitate collaborative writing of this document, as well as Gail Pieper for her masterful editing. We also thank Phil Colella and Hans Johansen for providing information about Chombo, and Tim Tautges for providing information about MOAB.

Funding

The workshop from which this document originated was sponsored by the Institute for Computing in Science, funded by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

References

Aagaard B, Kientz S, Knepley MG, et al. (2012) *PyLith user manual, version 1.7.1*. Available at: http://geodynamics.org/cig/software/pylith.

Abhyankar S (2011) *Development of an implicitly coupled electromechanical and electromagnetic transients simulator for power systems*. PhD Thesis, Illinois Institute of Technology, USA.

Abraham F, Broughton J, Bernstein N, et al (1998) Spanning the length scales in dynamic simulation. *Computers in Physics* 12: 538–546.

Adams MF (2004) Algebraic multigrid methods for constrained linear systems with applications to contact problems in solid mechanics. *Numerical Linear Algebra with Applications* 11(2–3): 141–153. DOI 10.1002/nla.374.

Adams M, Higdon D, et al. (2012) *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification*. Washington, DC: The National Academies Press. Available at: http://www.nap.edu/catalog.php?record_id=13395.

Ainsworth M and Oden JT (1997) A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering* 142(1–2): 1–88. DOI 10.1016/S0045-7825(96)01107-3.

Akcelik V, Biros G, Ghattas O, et al. (2005) Adjoint methods for electromagnetic shape optimization of the low-loss cavity for the International Linear Collider. *Journal of Physics: Conference Series* 16: 435–445. DOI 10.1088/1742-6596/16/1/059.

Akcelik V, Candel A, Kabel A, et al. (2008a) *Parallel Computation of Integrated Electromagnetic, Thermal and Structural Effects for Accelerator Cavities*. Technical Report SLAC-PUB-13280, Stanford Linear Accelerator Center (SLAC).

Akcelik V, Ko K, Lee L-Q, et al. (2008b) Shape determination for deformed electromagnetic cavities. *Journal of Computational Physics* 227: 1722–1738. DOI 10.1016/j.jcp.2007.09.029.

Akcelik V, Lee L-Q, Li Z, et al. (2009) Thermal analysis of SRF cavity couplers using parallel multiphysics tool TEM3P. In: *Particle accelerator conference*, Vancouver, Canada.

Alexandrescu A (2001) *Modern C++ Design: Generic Programming and Design Patterns Applied (C++ In-Depth)*. Reading, MA: Addison-Wesley.

Almgren A, et al. (2012) *NYX user guide*. Lawrence Berkeley National Laboratory.

Almgren A, Beckner V, Bell J, et al. (2010) CASTRO: A new compressible astrophysical solver. I. Hydrodynamics and self-gravity. *The Astrophysical Journal* 715: 1221–1238. DOI 10.1088/0004-637X/715/2/1221.

Almgren A, Bell J, Colella P, et al. (1998) A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations. *Journal of Computational Physics* 142: 1–46. DOI 10.1006/jcph.1998.5890.

Almgren A, Bell J, Nonaka A, et al. (2008) Low Mach number modeling of Type Ia supernovae. III. Reactions. *The Astrophysical Journal* 684: 449–470. DOI 10.1086/590321.

Almgren A, Bell J, Rendleman C, et al. (2006a) Low Mach number modeling of Type Ia supernovae. I. Hydrodynamics. *The Astrophysical Journal* 637: 922–936. DOI 10.1086/498426.

Almgren A, Bell J, Rendleman C, et al. (2006b) Low Mach number modeling of Type Ia supernovae. II. Energy evolution. *The Astrophysical Journal* 649: 927–938. DOI 10.1086/507089.

Anderson WK, Gropp WD, Kaushik DK, et al. (1999). Achieving high sustained performance in an unstructured mesh CFD application. In: *Proceedings of the 1999 ACM/IEEE conference on supercomputing*. DOI 10.1145/331532.331600.

Ang J, Evans K, Geist A, et al. (2012) *Report on the workshop on extreme-scale solvers: Transitions to future architectures*. Washington, DC: Office of Advanced Scientific Computing Research, U.S. Department of Energy. Available at: http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/reportExtremeScaleSolvers2012.pdf.

Ascher UM and Petzold LR (1998) *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Ascher UM, Ruuth SJ and Spiteri RJ (1997) Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics* 25: 151–167. DOI 10.1016/S0168-9274(97)00056-1.

Ascher UM, Ruuth SJ and Wetton BTR (1995) Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis* 32(3): 797–823. DOI 10.1137/0732037.

Ashby SF and Falgout RD (1996) A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations. *Nuclear Science and Engineering* 124(1): 145–159.

Ateljevich E, Colella P, Graves DT, et al. (2010). CFD modeling in the San Francisco Bay and Delta. In: *Proceedings of the fourth SIAM conference on mathematics for industry (MI09)*, 2009, pp. 99–107.

Axelsson O (1972) A generalized SSOR method. *BIT* 12(4): 443–467. DOI 10.1007/BF01932955.

Baaijens FPT (2001) A fictitious domain/mortar element method for fluid–structure interaction. *International Journal for Numerical Methods in Fluids* 35(7): 743–761. DOI 10.1002/1097-0363(20010415)35:7<743::AID-FLD109>3.0.CO;2-A.

Baker AH, Falgout RD, Kolev TV, et al. (2012) Scaling hypre's multigrid solvers to 100,000 cores. In: Berry M, et al. (eds) *High Performance Scientific Computing: Algorithms and Applications – A Tribute to Prof. Ahmed Sameh*. New York: Springer, pp. 261–279. DOI 10.1007/978-1-4471-2437-5_13.

Balay S, Brown J, Buschelman K, et al. (2012) *PETSc Users Manual, ANL-95/11 – revision 3.3.* Argonne National Laboratory. Available at: http://www.mcs.anl.gov/petsc.

Balay S, Gropp WD, McInnes LC, et al. (1997) Efficient management of parallelism in object oriented numerical software libraries. In: Arge E, Bruaset AM and Langtangen HP (eds) *Modern Software Tools in Scientific Computing*. Basel: Birkhäuser, pp. 163–202.

Balhoff MT, Thomas SG and Wheeler MF (2008) Mortar coupling and upscaling of pore-scale models. *Computational Geosciences* 12: 15–27. DOI 10.1007/s10596-007-9058-6.

Balhoff MT, Thompson KE and Hjortsø M (2007) Coupling pore-scale networks to continuum-scale models of porous media. *Computers & Geosciences* 33: 393–410. DOI 10.1016/j.cageo.2006.05.012.

Banach S (1922) Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae* 3: 133–181.

Barker AT and Cai X-C (2010) Two-level Newton and hybrid Schwarz preconditioners for fluid-structure interaction. *SIAM Journal of Scientific Computing* 32(4): 2395–2417. DOI 10.1137/090779425.

Barry DA, Miller CT and Culligan-Hensley PJ (1996) Temporal discretisation errors in non-iterative split-operator approaches to solving chemical reaction groundwater transport models. *Journal of Contaminant Hydrology* 22(1–2): 1–17. DOI 10.1016/0169-7722(95)00062-3.

Bartel A and Günther M (2002) A multirate W-method for electrical networks in state-space formulation. *Journal of Computational and Applied Mathematics* 147(2): 411–425. DOI 10.1016/S0377-0427(02)00476-4.

Bartlett R (2010) *Thyra Coding and Documentation Guidelines*. Sandia Technical Report SAND2010-2051, Sandia National Laboratories.

Battiato I, Tartakovsky DM, Tartakovsky AM, et al. (2011) Hybrid models of reactive transport in porous and fractured media. *Advances in Water Resources* 34(9): 1140–1150. DOI 10.1016/j.advwatres.2011.01.012.

Bauman PT, Oden JT and Prudhomme S (2008) Adaptive multiscale modeling of polymeric materials: Arlequin coupling and Goals algorithms. *Computer Methods in Applied Mechanics and Engineering* 198(5–8): 799–818. DOI 10.1016/j.cma.2008.10.014.

Baumgartner G, Auer A, Bernholdt DE, et al. (2005) Synthesis of high-performance parallel programs for a class of ab initio quantum chemistry models. *Proceedings of the IEEE* 93: 276–292. DOI 10.1109/JPROC.2004.840311.

Beall MW and Shephard MS (1997) A general topology-based mesh data structure. *International Journal of Numerical Methods in Engineering* 40(9): 1573–1596. DOI 10.1002/(SICI)1097-0207(19970515)40:9<1573::AID-NME128>3.0.CO;2-9.

Becker R and Rannacher R (2001) An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica* : 1–102. DOI 10.1017/S0962492901000010.

Bell J, et al. (2012) *BoxLib User's Guide*. Technical Report, CCSE, Lawrence Berkeley National Laboratory. Available at: https://ccse.lbl.gov/BoxLib/BoxLibUsersGuide.pdf.

Belytschko T and Xiao SP (2003) Coupling methods for continuum model with molecular model. *International Journal for Multiscale Computational Engineering* 1: 115–126. DOI 10.1615/IntJMultCompEng.v1.i1.100.

Benssousan A, Lions J and Papanicoulau G (1978) *Asymptotic Analysis for Periodic Structures*. Amsterdam: North-Holland Publishing Company.

Benzi R, Succi S and Vergassola M (1992) The lattice Boltzmann equation: Theory and applications. *Physics Reports* 222: 145–197. DOI 10.1016/0370-1573(92)90090-M.

Bergen B, Gradl T, Hülsemann F, et al. (2006) A massively parallel multigrid method for finite elements. *Computing in Science and Engineering* 8(6): 56–62. DOI 10.1109/MCSE.2006.102.

Berger MJ and Colella P (1989) Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 82: 64–84. DOI 10.1016/0021-9991(89)90035-1.

Bernardi C, Maday Y and Patera A (1993) Domain decomposition by the mortar element method. In: Kaper H, et al. (eds) *Asymptotic and Numerical Methods for Partial Differential Equations with Critical Parameters*. Dordrecht: Reidel, pp. 269–286.

Bernardi C, Maday Y and Patera AT (1994) A new nonconforming approach to domain decomposition: The mortar element method. In: Brezzi H and Lions JL (eds) *Nonlinear PDEs and Their Applications*. New York: Longman.

Biezuner R, Brown J, Ercole G, et al. (2012) Computing the first eigenpair of the p-Laplacian via inverse iteration of sublinear supersolutions. *Journal of Scientific Computing* 52: 180–201. DOI 10.1007/s10915-011-9540-0.

Blackford LS, Choi J, Cleary A, et al. (1997) *ScaLAPACK users' guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Blankenbach B, Busse F, Christensen U, et al. (1989) A benchmark comparison for mantle convection codes. *Geophysical Journal International* 98(1): 23–38. DOI 10.1111/j.1365-246X.1989.tb05511.x.

Blom FJ (1998) A monolithical fluid–structure interaction algorithm applied to the piston problem. *Computer Methods in Applied Mechanics and Engineering* 167: 369–391. DOI 10.1016/S0045-7825(98)00151-0.

Bochev P and Shashkov M (2005) Constrained interpolation (remap) of divergence-free fields. *Computer Methods in Applied Mechanics and Engineering* 194(2–5): 511–530. DOI 10.1016/j.cma.2004.05.018.

Bowers RL and Wilson JR (1982) A numerical model for stellar core collapse calculations. *The Astrophysical Journal Supplement Series* 50: 115–159.

Box GEP and Draper NR (2007) *Response Surfaces, Mixtures, and Ridge Analysis (Wiley Series in Probability and Statistics*, vol. 15). Hoboken, NJ: Wiley.

Bramble J, Pasciak J and Xu J (1990) Parallel multilevel preconditioners. *Mathematics of Computation* 55: 1–22. DOI 10.1090/S0025-5718-1990-1023042-6.

Brandt A (1973) Multi-level adaptive technique (MLAT) for fast numerical solutions to boundary value problems. In: *Proceedings of the 3rd international conference on numerical methods in fluid mechanics (Lecture Notes in Physics*, vol. 18). Berlin: Springer-Verlag, pp. 82–89.

Brandt A (2002) Multiscale scientific computation: Review 2001. *Lecture Notes in Computational Science and Engineering* 20: 3–96.

Brown PN (1987) A local convergence theory for combined inexact-Newton/finite-difference projection methods. *SIAM Journal of Numerical Analysis* 24: 407–434. DOI 10.1137/0724031.

Brown J (2010) Efficient nonlinear solvers for nodal high-order finite elements in 3D. *Journal of Scientific Computing* 45: 48–63. DOI 10.1007/s10915-010-9396-8

Brown F (2012) MCNP webpage. Available at: https://laws.lanl.gov/vhosts/mcnp.lanl.gov/.

Brown D, et al. (2008) *Applied mathematics at the U.S. Department of Energy: Past, present, and a view to the future*. Office of Science, U.S. Department of Energy. Available at: http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Brown_report_may_08.pdf.

Brown J, Knepley MG, May DA, et al. (2012) Composable linear solvers for multiphysics. In: *Proceedings of the 11th international symposium on parallel and distributed computing (ISPDC 2012)*. DOI 10.1109/ISPDC.2012.16.

Brown D, Messina P, et al. (2010) *Scientific grand challenges: Crosscutting technologies for computing at the exascale*. Office of Science, U.S. Department of Energy. Available at: http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/crosscutting_grand_challenges.pdf.

Bruenn SW (1985) Stellar core collapse: Numerical model and infall epoch. *The Astrophysical Journal Supplement Series* 58: 771–841.

Brune P, Knepley M, Smith B, et al. (2012) Composing scalable nonlinear solvers. Preprint ANL/MCS-P2010-0112, Argonne National Laboratory.

Buis S, Piacentini A and Déclat D (2006) PALM: A computational framework for assembling high-performance computing applications. *Concurrency and Computation: Practice and Experience* 18(2): 231–245. DOI 10.1002/cpe.914.

Bulatewicz T and Cuny J (2006) A domain-specific language for model coupling. In: *Proceedings of the 2006 winter simulation conference*.

Bulatov V, Abraham FF, Kubin L, et al. (1998) Connecting atomistic and mesoscopic simulations of crystal plasticity. *Nature* 391: 669–672. DOI 10.1038/35577.

Bungartz H-J, Gatzhammer B, Mehl M, et al. (2010) Partitioned simulation of fluid–structure interaction on Cartesian grids. In: Bungartz H-J, Mehl M and Schäfer M (eds) *Fluid–Structure Interaction – Modelling, Simulation, Optimisation, Part II, LNCSE*. Berlin: Springer, pp. 255–284.

Bunge H, Richards MA, Lithgow-Bertelloni C, et al. (1998). Time scales and heterogeneous structure in geodynamic earth models. *Science* 280(5360): 91–95. DOI 10.1126/science.280.5360.91.

Butcher JC (2008) *Numerical Methods for Ordinary Differential Equations*. 2nd edn. New York: Wiley.

Cai X-C (1989) *Some Domain Decomposition Algorithms for Nonselfadjoint Elliptic and Parabolic Partial Differential Equations*. Technical Report 461, Courant Institute.

Cai X-C and Keyes DE (2002) Nonlinearly preconditioned inexact Newton algorithms. *SIAM Journal of Scientific Computing* 24(1): 183–200. DOI 10.1137/S106482750037620X.

Cai X-C and Sarkis M (1999) A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal of Scientific Computing* 21: 792–797. DOI 10.1137/S106482759732678X.

Cai Z and Zhang S (2009) Recovery-based error estimator for interface problems: Conforming linear elements. *SIAM Journal of Numerical Analysis* 47(3): 2132–2156. DOI 10.1137/080717407.

Cai Z and Zhang S (2010a) Flux recovery and a posteriori error estimators: Conforming elements for scalar elliptic equations. *SIAM Journal of Numerical Analysis* 48(2): 578–602. DOI 10.1137/080742993.

Cai Z and Zhang S (2010b) Recovery-based error estimators for interface problems: Mixed and nonconforming finite elements. *SIAM Journal of Numerical Analysis* 48(1): 30–52. DOI 10.1137/080722631.

Camporese M, Paniconi C, Putti M, et al. (2009) A comparison of data assimilation techniques for a coupled model of surface and subsurface flow. *Vadose Zone Journal* 8(4): 837–845. DOI 10.2136/vzj2009.0018.

Cao Y and Petzold L (2006) A posteriori error estimation and global error control for ordinary differential equations by the adjoint method. *SIAM Journal of Scientific Computing* 26(2): 359–374. DOI 10.1137/S1064827503420969.

Car R and Parrinello M (1985) Unified approach for molecular-dynamics and density-functional theory. *Physical Review Letters* 55: 2471–2474. DOI 10.1103/PhysRevLett.55.2471.

Carey V, Estep D and Tavener S (2009) A posteriori analysis and adaptive error control for multiscale operator decomposition solution of elliptic systems I: Triangular systems. *SIAM Journal of Numerical Analysis* 47(1): 740–761. DOI 10.1137/070689917.

Carrayrou J, Mosé R and Behra P (2004) Operator-splitting procedures for reactive transport and comparison of mass balance errors. *Journal of Contaminant Hydrology* 68(3–4): 239–268. DOI 10.1016/S0169-7722(03)00141-4.

Cary JR, et al. (2011) Framework Application for Core-Edge Transport Simulations (FACETS). Proto-FSP project, available at: https://ice.txcorp.com/trac/facets.

Cary JR, Hakim A, Miah M, et al. (2009) FACETS – a framework for parallel coupling of fusion components. In: *The 18th Euromicro international conference on parallel, distributed and network-based computing*, Pisa, Italy.

Castor JI (2004) *Radiation Hydrodynamics*. Cambridge: Cambridge University Press.

Cederberg GA, Street RL and Leckie JO (1985) A groundwater mass transport and equilibrium chemistry model for multicomponent systems. *Water Resources Research* 21(8): 1095–1104. DOI 10.1029/WR021i008p01095.

Chacón L and Knoll D (2003) A 2D high-β Hall MHD implicit nonlinear solver. *Journal of Computational Physics* 188(2): 573–592 DOI 10.1016/S0021-9991(03)00193-1.

Chacón L, Knoll D and Finn J (2002) An implicit, nonlinear reduced resistive MHD solver. *Journal of Computational Physics* 178: 15–36. DOI 10.1006/jcph.2002.7015.

Chan T and Zou J (1994) Additive Schwarz domain decomposition methods for elliptic problems on unstructured meshes. *Numerical Algorithms* 8: 329–346. DOI 10.1007/BF02142697.

Chand KK, Diachin LF, Li X, et al. (2008) Toward interoperable mesh, geometry and field components for PDE simulation development. *Engineering with Computers* 24(2): 165–182. DOI 10.1007/s00366-007-0080-z.

Chen J, McInnes LC and Zhang H (2012) *Analysis and Practical Use of Flexible BiCGStab*. Technical Report ANL/MCS-P3039-0912, Argonne National Laboratory.

Cheng G, Wang H and Smithe D (2011) RF-thermal combined simulations of a superconducting HOM coaxial coupler. In: *Proceedings of 2011 particle accelerator conference*, New York, NY.

Chidyagwai P and Rivière B (2010) Numerical modelling of coupled surface and subsurface flow systems. *Advances in Water Resources* 33(1): 92–105. DOI 10.1016/j.advwatres.2009.10.012.

Chidyagwai P and Rivière B (2011) A two-grid method for coupled free flow with porous media flow. *Advances in Water Resources* 34(9): 1113–1123. DOI 10.1016/j.advwatres.2011.04.010.

Choi DI, Brown JD, Imbiriba B, et al. (2004) Interface conditions for wave propagation through mesh refinement boundaries. *Journal of Computational Physics* 193(2): 398–425. DOI 10.1016/j.jcp.2003.07.036.

Choi S, Potsdam M, Lee KH, et al. (2008) Helicopter rotor design using a time-spectral and adjoint-based method. In: *12th AIAA/ISSMO multidisciplinary and optimization conference*, British Columbia, Canada.

Christensen UR and Yuen DA (1985) Layered convection induced by phase transitions. *Journal of Geophysical Research* 90(B12): 10,291–10,300. DOI 10.1029/JB090iB12p10291.

Colella P, Graves DT, Modiano D, et al. (2000) *Chombo Software Package for AMR Applications*. Technical Report, Lawrence Berkeley National Laboratory. Available at: http://seesar.lbl.gov/anag/chombo/.

COMSOL (2012) Homepage at: http://www.comsol.com.

Concus P, Golub GH and O'Leary DP (1976) A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. In: Bunch JR and Rose DJ (eds) *Sparse Matrix Computations*. New York: Academic Press, pp. 309–332.

Connors JM, Banks JW, Hittinger JA, et al. (2012) *Quantification of Errors for Operator-Split Advection–Diffusion Calculations*. Technical Report LLNL-JRNL-566435, Lawrence Livermore National Laboratory.

Connors JM, Howell J and Layton W (2009) Partitioned timestepping for a parabolic two domain problem. *SIAM Journal of Numerical Analysis* 47(5): 3526–3549. DOI 10.1137/080740891.

Connors JM and Miloua A (2011) Partitioned time discretization for parallel solution of coupled ODE systems. *BIT* 51(2): 253–273. DOI 10.1007/s10543-010-0295-z.

Constantinescu EM and Sandu A (2007) Multirate timestepping methods for hyperbolic conservation laws. *Journal of Scientific Computing* 33(3): 239–278. DOI 10.1007/s10915-007-9151-y.

Constantinescu EM and Sandu A (2010a) Extrapolated implicit-explicit time stepping. *SIAM Journal of Scientific Computing* 31(6): 4452–4477. DOI 10.1137/080732833.

Constantinescu EM and Sandu A (2010b) Optimal explicit strong-stability-preserving general linear methods. *SIAM Journal of Scientific Computing* 32: 3130–3150. DOI 10.1137/090766206.

Constantinescu EM, Sandu A and Carmichael G (2008) Modeling atmospheric chemistry and transport with dynamic adaptive resolution. *Computational Geosciences* 12(2): 133–151. DOI 10.1007/s10596-007-9065-7.

Cornford SL and Martin DF, Graves DT, et al. (2012) Adaptive mesh, finite-volume modeling of marine ice sheets. *Journal of Computational Physics*. Epub ahead of print. DOI: 10.1016/j.jcp.2012.08.037.

Crouzeix M (1980) Une méthode multipas implicite-explicite pour l'approximation des équations d'évolution parabolique. *Numerische Mathematik* 35: 257–276.

Dai Y, Zeng X, Dickinson RE, et al. (2003) The common land model. *Bulletin of the American Meteorological Society* 84(8): 1013–1023. DOI 10.1175/BAMS-84-8-1013.

Davies DR, Wilson CR and Kramer SC (2011) Fluidity: A fully unstructured anisotropic adaptive mesh computational modeling framework for geodynamics. *Geochemistry Geophysics Geosystems* 12. DOI 10.1029/2011GC003551.

Day M and Bell J (2000) Numerical simulation of laminar reacting flows with complex chemistry. *Combustion Theory and Modelling* 4(4): 535–556. DOI 10.1088/1364-7830/4/4/309.

De Boer A, van Zuijlen AH and Bijl H (2007) Review of coupling methods for non-matching meshes. *Computer Methods in Applied Mechanics and Engineering* 196(8): 1515–1525. DOI 10.1016/j.cma.2006.03.017.

Degroote J, Bathe KJ and Vierendeels J (2009) Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction. *Computers and Structures* 87: 793–801. DOI 10.1016/j.compstruc.2008.11.013.

Dekker C (2007) Solid state nanopores. *Nature Nanotechnology* 2: 209–215. DOI 10.1038/nnano.2007.27.

Dembo RS, Eisenstat SC and Steihaug T (1982) Inexact Newton methods. *SIAM Journal of Numerical Analysis* 19: 400–408. DOI 10.1137/0719025.

Dennis JM, Vertenstein M, Worley PH, et al. (2012) Computational performance of ultra-high-resolution capability in the Community Earth System Model. *International Journal of High Performance Computing Applications* 26(1): 5–16. DOI 10.1177/1094342012436965.

Devincre B and Kubin L (1997) Mesoscopic simulations of dislocations and plasticity. *Materials Science and Engineering: A* 234–236: 8–14. DOI 10.1016/S0921-5093(97)00146-9.

Devine K, Diachin L, Kraftcheck J, et al. (2009) Interoperable mesh components for large-scale, distributed-memory simulations. *Journal of Physics: Conference Series* 180: 012011. DOI 10.1088/1742-6596/180/1/012011.

DeVito Z, Joubert N, Palacios F, et al. (2011) Lizst: A domain specific language for building portable mesh-based PDE solvers. In: *SC '11: Proceedings of the 2011 ACM/IEEE international conference for high performance computing, networking, storage and analysis*.

Dohrmann CR (2003) A preconditioner for substructuring based on constrained energy minimization. *SIAM Journal of Scientific Computing* 25(1): 246–258. DOI 10.1137/S1064827502412887.

Dohrmann CR and Widlund OB (2010) Hybrid domain decomposition algorithms for compressible and almost incompressible elasticity. *International Journal for Numerical Methods in Engineering* 82(2): 157–183. DOI 10.1002/nme.2761.

Dongarra J, Beckman P, et al. (2011) The International Exascale Software Project roadmap. *International Journal of High Performance Computing Applications* 25: 3–60. DOI 10.1177/1094342010391989.

Dorr MR, Cohen RH, Colella P, et al. (2010) Numerical simulation of phase space advection in gyrokinetic models of fusion plasmas. In: *Proceedings of the 2010 Scientific Discovery through Advanced Computing (SciDAC) conference*, Chattanooga, TN, 11–15 July 2010. Oak Ridge, TN: Oak Ridge National Laboratory, pp. 42–52. Available at: http://computing.ornl.gov/workshops/scidac2010/2010_SciDAC_Proceedings.pdf.

Drake RP, Doss FW, McClarren RG, et al. (2011) Radiative effects in radiative shocks in shock tubes. *High Energy Density Physics* 7: 130–140. DOI 10.1016/j.hedp.2011.03.005.

Dryja M and Widlund O (1987) *An Additive Variant of the Schwarz Alternating Method for the Case of Many Subregions*. Technical Report 339, Courant Institute.

Dubcova L, Solin P, Hansen G, et al. (2011) Comparison of multimesh hp-FEM to interpolation and projection methods for spatial coupling of thermal and neutron diffusion calculations. *Journal of Computational Physics* 230(4): 1182–1197. DOI 10.1016/j.jcp.2010.10.034.

Dubey A, Antypas K, Ganapathy MK, et al. (2009) Extensible component-based architecture for FLASH, a massively parallel, multiphysics simulation code. *Parallel Computing* 35(10–11): 512–522. DOI 10.1016/j.parco.2009.08.001.

Durran DR and Blossey PN (2012) Implicit–explicit multistep methods for fast-wave–slow-wave problems. *Monthly Weather Review* 140: 1307–1325. DOI 10.1175/MWR-D-11-00088.1.

Dutt A, Greengard L and Rokhlin V (2000) Spectral deferred correction methods for ordinary differential equations. *BIT* 40: 241–266. DOI 10.1023/A:1022338906936.

E Weinan, Engquist B and Huang Z (2003) Heterogeneous multiscale method: A general methodology for multiscale modeling. *Physical Review* B 67(9): 092101-1–092101-4. DOI 10.1103/PhysRevB.67.092101.

E Weinan, Engquist B, Li X, RenWand Vanden-Eijenden E (2007) Heterogeneous multiscale methods: A review. *Communications in Computational Physics* 2(3): 367–450. Available at: http://www.global-sci.com/openaccess/v2_367.pdf.

Ebel BA, Mirus BB, Heppner CS, et al. (2009) First-order exchange coefficient coupling for simulating surface water-groundwater interactions: Parameter sensitivity and consistency with a physics-based approach. *Hydrological Processes* 23(13): 1949–1959. DOI 10.1002/Hyp.7279.

Eddington AS (2012) Quote available at: http://www.brainyquote.com/quotes/authors/a/arthur_eddington.html.

Eisenstat SC and Walker HF (1994) Globally convergent inexact Newton methods. *SIAM Journal on Optimization* 4: 393–422. DOI 10.1137/0804022.

Eisenstat SC and Walker HF (1996) Choosing the forcing terms in an inexact Newton method. *SIAM Journal of Scientific Computing* 17: 16–32. DOI 10.1137/0917003.

Elman H, Howle V, Shadid J, et al. (2003) A parallel block multi-level preconditioner for the 3D incompressible Navier–Stokes equations. *Journal of Computational Physics* 187: 504–523. DOI 10.1016/S0021-9991(03)00121-9.

Elman H, Howle V, Shadid J, et al. (2008) A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 227(3): 1790–1808. DOI 10.1016/j.jcp.2007.09.026.

Elwasif W, Bernholdt DE, Shet AG, et al. (2010) The design and implementation of the SWIM Integrated Plasma Simulator. In: *18th Euromicro international conference on parallel, distributed and network-based processing (PDP)*, pp. 419–427. DOI 10.1109/PDP.2010.63.

Escript/Finley (2012) Details available at: https://launchpad.net/escript-finley/.

Estep D, Ginting V, Ropp D, et al. (2008a) An a posteriori–a priori analysis of multiscale operator splitting. *SIAM Journal on Numerical Analysis* 46(3): 1116–1146. DOI 10.1137/07068237X.

Estep D, Larson M and Williams R (2000) Estimating the error of numerical solutions of systems of reaction–diffusion equations. *Memoirs of the American Mathematical Society* 146:

Estep D, Tavener S and Wildey T (2008b) A posteriori analysis and improved accuracy for an operator decomposition solution of a conjugate heat transfer problem. *SIAM Journal on Numerical Analysis* 46: 2068–2089. DOI 10.1137/060678737.

Estep D, Tavener S and Wildey T (2009) A posteriori error analysis for a transient conjugate heat transfer problem. *Finite Elements in Analysis and Design* 45(4): 263–271. DOI 10.1016/j.finel.2008.10.011.

Evans KJ, Rouson DWI, Salinger AG, et al. (2009) A scalable and adaptable solution framework within components of the community climate system model. In: *Proceedings of the 9th international conference on computational science*, pp. 332–341. DOI 10.1007/978-3-642-01973-9_37.

Evans TM, Stafford AS, Slaybaugh RN, et al. (2010) Denovo: A new three-dimensional parallel discrete ordinates code in SCALE. *Nuclear Technology* 171(2): 171–200.

Ewing RE, Lazarov RD and Vassilevski PS (1991) Local refinement techniques for elliptic problems on cell–centered grids. I: Error analysis. *Mathematics of Computation* 56: 437–461. DOI 10.1090/S0025-5718-1991-1066831-5.

Faber V and Manteuffel T (1984) Necessary and sufficient conditions for the existence of a conjugate-gradient method. *SIAM Journal on Numerical Analysis* 21(2): 352–362. DOI 10.1137/0721026.

Falgout R, et al. (2011) *Hypre Users Manual, Revision 2.8*. Technical Report, Lawrence Livermore National Laboratory. Available at: http://www.llnl.gov/CASC/hypre.

Farhat C, Degand C, Koobus B, et al. (1998a) Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering* 163(1–4): 231–245. DOI 10.1016/S0045-7825(98)00016-4.

Farhat C, Geuzaine P and Brown G (2003) Application of a three-field nonlinear fluid–structure formulation to the prediction of the aeroelastic parameters of an F-16 fighter. *Computers and Fluids* 32: 3–29. DOI 10.1016/S0045-7930(01)00104-9.

Farhat C, Lesoinne M and Le Tallec P (1998b) Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity. *Computer Methods in Applied Mechanics and Engineering* 157(1–2): 95–114. DOI 10.1016/S0045-7825(97)00216-8.

Farhat C, Pierson K and Degand C (2001) Multidisciplinary simulation of the maneuvering of an aircraft. *Engineering with Computers* 17: 16–27. DOI 10.1007/PL00007193.

Farhat C, Rallu A, Wang K, et al. (2010) Robust and provably second-order explicit–explicit and implicit–explicit staggered time-integrators for highly nonlinear fluid–structure interaction problems. *International Journal for Numerical Methods in Engineering* 84: 73–107. DOI 10.1002/nme.2883.

Farhat C and Roux FX (1991) A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering* 32: 1205–1227. DOI 10.1002/nme.1620320604.

Farhat C, van der Zee G and Geuzaine P (2006) Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity. *Computer Methods in Applied Mechanics and Engineering* 195: 1973–2001. DOI 10.1016/j.cma.2004.11.031.

Feichtinger C, Donath S, Koestler H, et al. (2011) WaLBerla: HPC software design for computational engineering simulations. *Journal of Computational Science* 2(2): 105–112. DOI 10.1016/j.jocs.2011.01.004.

Filbet F and Jin S (2010) A class of asymptotic preserving schemes for kinetic equations and related problems with stiff sources. *Journal of Computational Physics* 229(20): 7625–7648. DOI 10.1016/j.jcp.2010.06.017.

Fischer P, et al. (2012) Nek5000. Available at: https://nek5000.mcs.anl.gov.

Fischer P, Kruse GW and Loth F (2002) Spectral element methods for transitional flows in complex geometries. *Journal of Scientific Computing* 17: 81–98. DOI 10.1023/A:1015188211796.

Fischer P, Lottes J, Pointer D, et al. (2008) Petascale algorithms for reactor hydrodynamics. *Journal of Physics: Conference Series* 125(1): 012076. DOI 10.1088/1742-6596/125/1/012076.

Fischer P, Lottes J, Siegel A, et al. (2007) Large eddy simulation of wire wrapped fuel pins. In: *Joint international topical meeting on mathematics & computation and supercomputing in nuclear applications*.

Fish J (2010) *Multiscale Methods: Bridging the Scales in Science and Engineering*. Oxford: Oxford University Press.

Fish J, Nuggehally MA, Shephard MS, et al. (2007) Concurrent AtC coupling based on a blend of the continuum stress and the atomistic force. *Computer Methods in Applied Mechanics and Engineering* 196(45–48): 4548–4560. DOI 10.1016/j.cma.2007.05.020.

Flemisch B, Kaltenbacher M, Triebenbacher S, et al. (2010) The equivalence of standard and mixed finite element methods in applications to elasto-acoustic interaction. *SIAM Journal of Scientific Computing* 32(4): 1980–2006. DOI 10.1137/090758507.

Ford RW, Riley GD, Bane MK, et al. (2006) GCF: A general coupling framework. *Concurrency and Computation: Practice and Experience* 18: DOI 10.1002/cpe.910.

Frank J, Hundsdorfer W and Verwer JG (1997) On the stability of implicit-explicit linear multistep methods. *Applied Numerical Mathematics* 25(2–3): 193–205. DOI 10.1016/S0168-9274(97)00059-7.

Frank R and Ueberhuber W (1977) Iterated defect correction for the efficient solution of stiff systems of ordinary differential equations. *BIT* 17(2): 146–159. DOI 10.1007/BF01932286.

Freeze RA and Harlan RL (1969) Blueprint for a physically-based digitally-simulated, hydrologic response model. *Journal of Hydrology* 9: 237–258. DOI 10.1016/0022-1694(69)90020-1.

Freund RW, Golub GH and Nachtigal NM (1992) Iterative solution of linear systems. *Acta Numerica* : 87–100. DOI 10.1017/S0962492900002245.

Fyta M, Melchionna S, Kaxiras E, et al. (2006) Multiscale coupling of molecular dynamics and hydrodynamics: Application to DNA translocation through a nanopore. *Multiscale Modeling and Simulation* 5: 1156–1173. DOI 10.1137/060660576.

Fyta M, Melchionna S and Succi S (2011) Translocation of biomolecules through solid-state nanopores: Theory meets experiments. *Journal of Polymer Science Part B: Polymer Physics* 49: 985–1011. DOI 10.1002/polb.22284.

Gaston D, Guo L, Hansen G, et al. (2012) Parallel algorithms and software for nuclear, energy, and environmental applications, Part II: Multiphysics software. *Communications in Computational Physics* 12(3): 834–865. DOI 10.4208/cicp.091010.150711s.

Gaston D, Hansen G, Kadioglu S, et al. (2009a) Parallel multiphysics algorithms and software for computational nuclear engineering. *Journal of Physics: Conference Series* 180(1): 012012. DOI 10.1088/1742-6596/180/1/012012.

Gaston D, Newman C and Hansen G (2009b) MOOSE: a parallel computational framework for coupled systems of nonlinear equations. In: *American Nuclear Society 2009 international conference on advances in mathematics, computational methods, and reactor physics*, Saratoga Springs, NY.

Gaston D, Newman C, Hansen G, et al. (2009c) MOOSE: A parallel computational framework for coupled systems of nonlinear equations. *Nuclear Engineering and Design* 239: 1768–1778. DOI 10.1016/j.nucengdes.2009.05.021.

Gatzhammer B, Mehl M and Neckel T (2010) A coupling environment for partitioned multiphysics simulations applied to fluid–structure interaction scenarios. *Procedia Computer Science* 1: 681–689. DOI 10.1016/j.procs.2010.04.073.

Gautschi W (1997) *Numerical Analysis: An Introduction*. Cambridge, MA: Birkhauser.

Gear CW and Wells DR (1984) Multirate linear multistep methods. *BIT* 24: 484–502. DOI 10.1007/BF01934907.

Gee M and Hansen G (2012) Moertel mortar methods package. Homepage at: http://trilinos.sandia.gov/packages/moertel.

Gee M, Küttler U and Wall W (2011) Truly monolithic algebraic multigrid for fluid–structure interaction. *International Journal for Numerical Methods in Engineering* 85(8): 987–1016. DOI 10.1002/nme.3001.

Gee M, Siefert C, Hu J, et al. (2006) *ML 5.0 Smoothed Aggregation User's Guide*. Technical Report SAND2006-2649, Sandia National Laboratories.

Gent PR, Danabasoglu G, Donner LJ, et al. (2011) The community climate system model version 4. *Journal of Climate* 24(19): 4973–4991. DOI 10.1175/2011JCLI4083.1.

Geuzaine P, Brown G, Harris C, et al. (2003) Aeroelastic dynamic analysis of a full F-16 configuration for various flight conditions. *AIAA Journal* 41: 363–371.

Ghanem R and Spanos P (1991) *Stochastic Finite Elements: A Spectral Approach*. Berlin: Springer-Verlag.

Gingold RA and Monaghan JJ (1977) Smoothed particle hydrodynamics: Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society* 181: 375–389.

Giraldo FX (2005) Semi-implicit time-integrators for a scalable spectral element atmospheric model. *Quarterly Journal of the Royal Meteorological Society* 131(610): 2431–2454.

Giraldo FX, Perot JB and Fischer PF (2003) A spectral element semi-Lagrangian (SESL) method for the spherical shallow water equations. *Journal of Computational Physics* 190(2): 623–650. DOI 10.1016/S0021-9991(03)00300-0.

Giraldo FX and Restelli M (2009) High-order semi-implicit time-integrators for a triangular discontinuous Galerkin oceanic shallow water model. *International Journal for Numerical Methods in Fluids* 63(9): 1077–1102. DOI 10.1002/fld.2118.

Giraldo FX, Restelli M and Läuter M (2010) Semi-implicit formulations of the Navier-Stokes equations: Application to nonhydrostatic atmospheric modeling. *SIAM Journal of Scientific Computing* 32(6): 3394–3425. DOI 10.1137/090775889.

Girault V and Riviére B (2009) DG approximation of coupled Navier-Stokes and Darcy equations by Beaver-Joseph-Saffman interface condition. *SIAM Journal on Numerical Analysis* 47(3): 2052–2089. DOI 10.1137/070686081.

Glass M, et al. (2011) STK: Sierra Toolkit Mesh. Homepage at: http://trilinos.sandia.gov/packages/stk.

Glatzmaier GA and Roberts PH (1995) A three-dimensional convective dynamo solution with rotating and finitely conducting inner core and mantle. *Physics of the Earth and Planetary Interiors* 91(1–3): 63–75. DOI 10.1016/0031-9201(95)03049-3.

Gmeiner B, Gradl T, Köstler H, et al. (2012) Highly parallel geometric multigrid algorithm for hierarchical hybrid grids. In: Binder K, Münster G and Kremer M (eds) *NIC Symposium 2012*, vol. 45, pp. 323–330.

Golub GH and O'Leary D (1989) Some history of the conjugate-gradient and Lanczos algorithms: 1948-1976. *SIAM Review* 31(1): 50–102. DOI 10.1137/1031003.

Golub GH and Ye Q (1999) Inexact preconditioned conjugate gradient method with inner-outer iteration. *SIAM Journal of Scientific Computing* 21(4): 1305–1320. DOI 10.1137/S1064827597323415.

Goodale T, Allen G, Lanfermann G, et al. (2003) The Cactus framework and toolkit: Design and applications. In: *Vector and Parallel Processing – VECPAR 2002, 5th International Conference (Lecture Notes in Computer Science)*. Berlin: Springer.

Götz J, Iglberger K, Stürmer M, et al. (2010) Direct numerical simulation of particulate flows on 294,912 processor cores. In: *SC'10: Proceedings of the 2010 ACM/IEEE international conference for high performance computing, networking, storage and analysis*. Washington, DC: IEEE Computer Society, pp. 1–11. DOI 10.1109/SC.2010.20.

Gragg WB and Stetter HJ (1964) Generalized multistep predictor-corrector methods. *Journal of the ACM* 11(2): 188–209. DOI 10.1145/321217.321223.

Granas A and Dugundji J (2003) *Fixed Point Theory (Springer Monographs in Mathematics)*. New York: Springer.

Grandy J (1999) Conservative remapping and region overlays by intersecting arbitrary polyhedra. *Journal of Computational Physics* 148(2): 433–466. DOI 10.1006/jcph.1998.6125.

Grandy J (2004) *Simulations on Multiple Meshes: Algorithms*. Technical Report U-COPJ-2004-0545, Lawrence Livermore National Laboratory.

Graziani F (2005) Radiation diffusion: An overview of physical and numerical concepts. In: Mezzacappa A and Fuller GM (eds) *Open Issues in Core Collapse Supernova Theory*. Singapore: World Scientific Publishing Co., pp. 29–66.

Gropp WD, Keyes DE, McInnes LC, et al. (2000) Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD. *International Journal of High Performance Computing Applications* 14: 102–136. DOI 10.1177/109434200001400202.

Gross L, Bourgouin L, Hale AJ, et al. (2007) Interface modeling in incompressible media using level sets in Escript. *Physics of the Earth and Planetary Interiors* 163: 23–34. DOI 10.1016/j.pepi.2007.04.004.

Guo L, Huang H, Gaston D, et al. (2010) Modeling of calcite precipitation driven by bacteria-facilitated urea hydrolysis in a flow column using a fully coupled, fully implicit parallel reactive transport simulator. In: *Eos Transactions of the American Geophysical Union, 90(52), Fall Meeting Supplement*, San Francisco, CA.

Gupta A, Karypis G and Kumar V (1997) A highly scalable parallel algorithm for sparse matrix factorization. *IEEE Transactions on Parallel and Distributed Systems* 8(5): 502–520. DOI 10.1109/71.598277.

Gustafsson K (1991) Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods. *ACM Transactions on Mathematical Software* 17(4): 533–554. DOI 10.1145/210232.210242.

Gustafsson K (1994) Control theoretic techniques for stepsize selection in implicit Runge-Kutta methods. *ACM Transactions on Mathematical Software* 20(4): 496–517. DOI 10.1145/198429.198437.

Hakim AH, Rognlien TD, Groebner RJ, et al. (2012) Coupled core-edge simulations of H-mode buildup using the fusion application for core-edge transport simulations (FACETS) code. *Physics of Plasmas* 19: 032505. DOI 10.1063/1.3693148.

Hammond GE and Lichtner PC (2010) Field-scale modeling for the natural attenuation of uranium at the Hanford 300 area using high-performance computing. *Water Resources Research* 46: W09527. DOI 10.1029/2009WR008819.

Hammond GE, Valocchi AJ and Lichtner PC (2005) Application of Jacobian-free Newton–Krylov with physics-based preconditioning to biogeochemical transport. *Advances in Water Resources* 28(4): 359–376. DOI 10.1016/j.advwatres.2004.12.001.

Hansen G (2011) A Jacobian-free Newton Krylov method for mortar-discretized thermomechanical contact problems. *Journal of Computational Physics* 230: 6546–6562. DOI 10.1016/j.jcp.2011.04.038.

Hansen G, Martineau R, Newman C, et al. (2009a) Framework for simulation of pellet cladding thermal interaction (PCTI) for fuel performance calculations. In: *American Nuclear Society 2009 international conference on advances in mathematics, computational methods, and reactor physics*, Saratoga Springs, NY.

Hansen G, Newman C, Gaston D, et al. (2009b) An implicit solution framework for reactor fuel performance simulation. In: *20th international conference on structural mechanics in reactor technology (SMiRT 20)*, Espoo (Helsinki), Finland.

Hansen G and Owen S (2008) Mesh generation technology for nuclear reactor simulation; barriers and opportunities. *Nuclear Engineering and Design* 238(10): 2590–2605. DOI 10.1016/j.nucengdes.2008.05.016.

Hayken S (1998) *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall.

Heath M, et al. (2010) Center for Simulation of Advanced Rockets. Homepage at: http://www.csar.illinois.edu/.

Hegewald J, Krafczyk M, Tölke J, et al. (2008) An agent-based coupling platform for complex automata. In: *Proceedings of the 8th international conference on computational science, Part II*, pp. 227–233. DOI 10.1007/978-3-540-69387-1_25.

Henshaw WD (2002) Overture: An object-oriented framework for overlapping grid applications. In: *2002 AIAA conference on applied aerodynamics*, St. Louis, MO.

Henson VE and Yang UM (2000) BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics* 41: 155–177. DOI 10.1016/S0168-9274(01)00115-5.

Heroux MA, et al. (2012) Trilinos webpage. Homepage at: http://trilinos.sandia.gov.

Heroux MA, Bartlett R, Howle V, et al. (2005) An overview of the Trilinos project. *ACM Transactions on Mathematical Software* 31(3): 397–423. DOI 10.1145/1089014.1089021.

Hestenes MR and Stiefel E (1952) Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* 49(6): 409–436.

Higham DJ and Trefethen LN (1993) Stiffness of ODEs. *BIT* 33(2): 285–303. DOI 10.1007/BF01989751.

Hindmarsh AC, Brown PN, Grant KE, et al. (2005) SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software* 31(3): 363–396. DOI 10.1145/1089014.1089020.

Hirth JP and Lothe J (1992) *Theory of Dislocations*. Malabar, FL: Krieger Publishing.

Horowitz LW, Walters S, Mauzerall DL, et al. (2003) A global simulation of tropospheric ozone and related tracers: Description and evaluation of MOZART, version 2. *Journal of Geophysical Research* 108(D24): 4784. DOI 10.1029/2002JD002853.

Hron J and Turek S (2006) A monolithic FEM/multigrid solver for an ALE formulation of fluid–structure interaction with applications in biomechanics. In: Bungartz H-J and Schäfer M (eds) *Fluid–Structure Interaction (Lecture Notes in Computational Science and Engineering* vol. 53). Berlin Heidelberg: Springer, pp. 146–170. DOI 10.1007/3-540-34596-5_7.

Hughes TJR and Liu WK (1978a) Implicit-explicit finite elements in transient analysis: Stability theory. *Journal of Applied Mechanics* 45: 371–374. DOI 10.1115/1.3424305.

Hughes TJR and Liu WK (1978b) Implicit-explicit finite elements in transient analysis: Implementation and numerical examples. *Journal of Applied Mechanics* 45: 375–378. DOI 10.1115/1.3424304.

Hundsdorfer W and Ruuth SJ (2007) IMEX extensions of linear multistep methods with general monotonicity and boundedness properties. *Journal of Computational Physics* 225(2): 2016–2042. DOI 10.1016/j.jcp.2007.03.003.

Iglberger K and Rüde U (2011) Large scale rigid body dynamics. *Multibody System Dynamics* 25(1): 81–95. DOI 10.1007/s11044-010-9212-0.

Illinois Rocstar (2012) Homepage at: http://www.illinoisrocstar.com.

Irons BM and Tuck RC (1969) A version of the Aitken accelerator for computer iteration. *International Journal for Numerical Methods in Engineering* 1: 275–277. DOI 10.1002/nme.1620010306.

ITER (2012) International Thermonuclear Experimental Reactor. Homepage at: http://www.iter.org.

Jaiman R, Geubelle P, Loth E, et al. (2011a) Combined interface boundary condition method for unsteady fluid–structure interaction. *Computer Methods in Applied Mechanics and Engineering* 200: 27–39. DOI 10.1016/j.cma.2010.06.039.

Jaiman R, Geubelle P, Loth E, et al. (2011b) Transient fluid–structure interaction with non-matching spatial and temporal discretization. *Computers & Fluids* 50: 120–135. DOI 10.1016/j.compfluid.2011.07.001.

Jaiman R, Jiao X, Geubelle P, et al. (2006) Conservative load transfer along curved fluid–solid interface with non-matching meshes. *Journal of Computational Physics* 218(1): 372–397. DOI 10.1016/j.jcp.2006.02.016.

Jiao X and Heath MT (2004a) Common-refinement-based data transfer between nonmatching meshes in multiphysics simulations. *International Journal for Numerical Methods in Engineering* 61(14): 2402–2427. DOI 10.1002/nme.1147.

Jiao X and Heath MT (2004b) Overlaying surface meshes, part I: Algorithms. *International Journal of Computational Geometry & Applications* 14(6): 379–402. DOI 10.1142/S0218195904001512.

Jiao X and Heath MT (2004c) Overlaying surface meshes, part II: Topology preservation and feature matching. *International Journal of Computational Geometry & Applications* 14(6): 403–419. DOI 10.1142/S0218195904001524.

Jin J (2002) *The Finite Element Method in Electromagnetics*. New York: John Wiley & Sons.

Jobs S (1997) Apple world-wide developers' conference, closing keynote Q & A. Available at: http://www.youtube.com/watch?v= 3LEXae1j6EY#t=41m26s.

Johnson RW, Hansen G and Newman C (2011) The role of data transfer on the selection of a single vs. multiple mesh architecture for tightly coupled multiphysics applications. *Applied Mathematics and Computation* 217: 8943–8962. DOI 10.1016/j.amc.2011.03.101.

Jones J, Sudicky E, Brookfield A, et al. (2006) An assessment of the tracer-based approach to quantifying groundwater contributions to streamflow. *Water Resources Research* 42: DOI 10.1029/2005wr004130.

Jones J and Woodward C (2001) Newton–Krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems. *Advances in Water Resources* 24(7): 763–774. DOI 10.1016/S0309-1708(00)00075-0.

Kadioglu S and Knoll D (2010a) A fully second order implicit/explicit time integration technique for hydrodynamics plus nonlinear heat conduction problems. *Journal of Computational Physics* 229: 3237–3249. DOI 10.1016/j.jcp.2009.12.039.

Kadioglu S, Knoll D, Lowrie R, et al. (2010b) A second order self-consistent IMEX method for radiation hydrodynamics. *Journal of Computational Physics* 229: 8313–8332. DOI 10.1016/j.jcp.2010.07.019.

Kanevsky A, Carpenter MH, Gottlieb D, et al. (2007) Application of implicit–explicit high order Runge-Kutta methods to discontinuous-Galerkin schemes. *Journal of Computational Physics* 225: 1753–1781. DOI 10.1016/j.jcp.2007.02.021.

Kanney JF, Miller CT and Barry DA (2003a) Comparison of fully coupled approaches for approximating nonlinear transport and reaction problems. *Advances in Water Resources* 26(4): 353–372. DOI 10.1016/S0309-1708(02)00188-4.

Kanney JF, Miller CT and Kelley CT (2003b) Convergence of iterative split-operator approaches for approximating nonlinear reactive transport problems. *Advances in Water Resources* 26(3): 247–261. DOI 10.1016/S0309-1708(02)00162-8.

Kasianowicz JJ, Brandin E, Branton D, et al. (1996) Characterization of individual polynucleotide molecules using a membrane channel. *Proceedings of the National Academy of Sciences* 93: 13,770–13,773.

Katz RF, Knepley MG, Smith B, et al. (2007) Numerical simulation of geodynamic processes with the Portable Extensible Toolkit for Scientific Computation. *Physics of the Earth and*

Planetary Interiors 163: 52–68. DOI 10.1016/j.pepi.2007.04.016.

Katz RF, Spiegelman M and Holtzman B (2006) The dynamics of melt and shear localization in partially molten aggregates. *Nature* 442: 676–679. DOI 10.1038/nature05039.

Katz RF, Spiegelman M and Langmuir CH (2003) A new parameterization of hydrous mantle melting. *Geochemistry Geophysics Geosystems* 4: 19. DOI 10.1029/2002GC000433.

Katz RF and Worster MG (2010) The stability of ice-sheet grounding lines. *Proceedings of the Royal Society A* 466: 1597–1620. DOI 10.1098/rspa.2009.0434.

Kaus B, Mühlhaus H and May DA (2010) A stabilization algorithm for geodynamic numerical simulations with a free surface. *Physics of the Earth and Planetary Interiors* 181(1–2): 12–20. DOI 10.1016/j.pepi.2010.04.007.

Kaushik D, Smith M, Wollaber A, et al. (2009) Enabling high-fidelity neutron transport simulations on petascale architectures. In: *Supercomputing 2009 (SC '09): Proceedings of the conference on high performance computing networking, storage and analysis*, Portland, OR. New York: ACM Press, pp. 67:1–67:12. DOI 10.1145/1654059.1654128.

Kelley CT (1995) *Iterative Methods for Linear and Nonlinear Equations (Frontiers in Applied Mathematics)*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Kelley CT and Keyes DE (1998) Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis* 35: 508–523. DOI 10.1137/S0036142996304796.

Kennedy CA and Carpenter MH (2003) Additive Runge-Kutta schemes for convection–diffusion–reaction equations. *Applied Numerical Mathematics* 44(1–2): 139–181. DOI 10.1016/S0168-9274(02)00138-1.

Kettler R (1982) Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate-gradient methods. *Lecture Notes in Mathematics* 960: 502–534. DOI 10.1007/BFb0069941.

Kevrekidis IG, Gear CW, Hyman JM, et al. (2003) Equation-free coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level tasks. *Communications in Mathematical Sciences* 1(4): 715–762. Available at: http://projecteuclid.org/euclid.cms/1119655353.

Keyes DE (2011) Exaflop/s: The why and the how. *Comptes Rendus Mécanique* 339: 70–77. DOI 10.1016/j.crme.2010.11.002.

Kim S-H (2003) Simulation of quench dynamics in SRF cavities under pulsed operation. In: *Proceedings of the 2003 particle accelerator conference*.

King SD, Lee C, van Keken PE, et al. (2010) A community benchmark for 2-D Cartesian compressible convection in the Earth-â€™s mantle. *Geophysical Journal International* 180(1): 73–87. DOI 10.1111/j.1365-246X.2009.04413.x.

Kirk BS, Peterson JW, Stogner RH, et al. (2006) libMesh: A C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers* 22(3–4): 237–254. DOI 10.1007/s00366-006-0049-3.

Kitware (2011) CMake: The cross-platform, open-source build system. Homepage at: http://www.cmake.org/.

Klöppel T, Popp A, Küttler U, et al. (2011) Fluid–structure interaction for non-conforming interfaces based on a dual mortar formulation. *Computer Methods in Applied Mechanics and Engineering* 200(45–46): 3111–3126. DOI 10.1016/j.cma.2011.06.006.

Knap J and Ortiz M (2001) An analysis of the quasicontinuum method. *Journal of the Mechanics and Physics of Solids* 49: 1899–1923. DOI 10.1016/S0022-5096(01)00034-5.

Knoll D, Chacón L, Margolin L, et al. (2003) On balanced approximations for time integration of multiple time scale systems. *Journal of Computational Physics* 185: 583–611. DOI 10.1016/S0021-9991(03)00008-1.

Knoll D and Keyes DE (2004) Jacobian-free Newton–Krylov methods: A survey of approaches and applications. *Journal of Computational Physics* 193(2): 357–397. DOI 10.1016/j.jcp.2003.08.010.

Knoll D, Mousseau V, Chacón L, et al. (2005) Jacobian-free Newton–Krylov methods for the accurate time integration of stiff wave systems. *Journal of Scientific Computing* 25(1): 213–230. DOI 10.1007/s10915-004-4640-8.

Knoll D, Park H and Newman C (2011) Acceleration of -eigenvalue/criticality calculations using the Jacobian-free Newton–Krylov method. *Nuclear Science and Engineering* 167(2): 133–140.

Knoll D, Park R and Smith K (2009) Application of the Jacobian-free Newton–Krylov method in computational reactor physics. In: *American Nuclear Society 2009 international conference on advances in mathematics, computational methods, and reactor physics*, Saratoga Springs, NY.

Kollet SJ and Maxwell RM (2006) Integrated surface-groundwater flow modeling: A free-surface overland flow boundary condition in a parallel groundwater flow model. *Advances in Water Resources* 29(7): 945–958. DOI 10.1016/j.advwatres.2005.08.006.

Kollet SJ and Maxwell RM (2008) Capturing the influence of groundwater dynamics on land surface processes using an integrated, distributed watershed model. *Water Resources Research* 44(2): 18. DOI 10.1029/2007WR006004.

Kollet SJ, Maxwell RM, Woodward CS, et al. (2010) Proof of concept of regional scale hydrologic simulations at hydrologic resolution utilizing massively parallel computer resources. *Water Resources Research* 46: W04201. DOI 10.1029/2009wr008730.

Kothe D, et al. (2012) Consortium for Advanced Simulation of Light water reactors (CASL). Homepage at: http://www.casl.gov.

Kuang W and Bloxham J (1999) Numerical modeling of magnetohydrodynamic convection in a rapidly rotating spherical shell: Weak and strong field dynamo action. *Journal of Computational Physics* 153(1): 51–81. DOI 10.1006/jcph.1999.6274.

Kumar M, Duffy CJ and Salvage KM (2009) A second-order accurate, finite volume-based, integrated hydrologic modeling (FIHM) framework for simulation of surface and subsurface flow. *Vadose Zone Journal* 8(4): 873–890. DOI 10.2136/vzj2009.0014.

Lambert JD (1991) *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. New York: John Wiley & Sons, Inc.

Larson JW (2009a) Ten organising principles for coupling in multiphysics and multiscale models. *ANZIAM* 48: C1090–C1111.

Larson JW (2009b) Graphical notation for diagramming coupled systems. In: *Proceedings of the 9th international conference on computational science (ICCS2009) (Lecture Notes in Computer Science*, vol. 5544). New York: Springer, pp. 745–754. DOI 10.1007/978-3-642-01970-8_74.

Larson JW, Jacob R and Ong E (2005) The Model Coupling Toolkit: a new Fortran90 toolkit for building multiphysics parallel coupled models. *International Journal of High Performance Computing Applications* 19(3): 277–292. DOI 10.1177/1094342005056115.

Lee KH (2010) *Design optimization of periodic flows using a time-spectral discrete adjoint method*. PhD Thesis, Stanford University.

Lee LQ, Ge L, Kowalski M, et al. (2004) Solving large sparse linear systems in end-to-end accelerator structure simulations. In: *Proceedings of the 18th international parallel and distributed processing symposium*, Santa Fe, NM.

Lee L-Q, Li Z, Ng C, et al. (2009) *Omega3P: A Parallel Finite-Element Eigenmode Analysis Code for Accelerator Cavities*. Technical Report SLAC-PUB-13529, Stanford Linear Accelerator Center (SLAC).

Lesrel J, Bousson S, Junquera T, et al. (1997) Study of thermal effects in SRF cavities. In: *Proceedings of the 1997 workshop on RF superconductivity*, Abano Terme (Padova), Italy.

LeVeque R (2002) *Finite Volume Methods for Hyperbolic Problems*. Cambridge: Cambridge University Press.

LeVeque R and Oliger J (1983) Numerical methods based on additive splittings for hyperbolic partial differential equations. *Mathematics of Computation* 40(162): 469–497. DOI 10.1090/S0025-5718-1983-0689466-8.

Lewis EE and Miller WFJr (1993) *Computational Methods of Neutron Transport*. La Grange Park, IL: American Nuclear Society, Inc.

Li Z, et al. (2007) Towards simulations of electromagnetics and beam physics at the petascale. *Proceedings of PAC2007*, Albuquerque, NM.

Li J, Gershow M, Stein D, et al. (2003) DNA molecules and configurations in a solid-state nanopore microscope. *Nature Materials* 2: 611–615. DOI 10.1038/nmat965.

Li A, Li R and Fish J (2008) Generalized mathematical homogenization: From theory to practice. *Computer Methods in Applied Mechanics and Engineering* 197(41–42): 3225–3248. DOI 10.1016/j.cma.2007.12.002.

Liao B-S, Bai Z, Lee L-Q, et al. (2006) *Solving Large Scale Nonlinear Eigenvalue Problem in Next-Generation Accelerator Design*. Technical Report SLAC-PUB-12137, Stanford Linear Accelerator Center (SLAC).

Lichtner PC (1985) Continuum model for simultaneous chemical reactions and mass transport in hydrothermal systems. *Geochimica et Cosmochimica Acta* 49(3): 779–800DOI 10.1016/0016-7037(85)90172-3.

Lichtner PC, et al. (2010). *Modeling multiscale-multiphase-multicomponent subsurface reactive flows using advanced computing*. SciDAC-2 project. Available at: http://www.scidac.gov/groundwater/gwflow.html.

Liepe M, Moeller WD and Simrock SN (2001) Dynamic Lorentz force compensation with a fast piezoelectric tuner. In: *Proceedings of the 2001 particle accelerator conference*, Chicago, IL.

Lin P, Shadid J, Sala M, et al. (2009) Performance of a parallel algebraic multilevel preconditioner for stabilized finite element semiconductor device modeling. *Journal of Computational Physics* 228: 6250–6267. DOI 10.1016/j.jcp.2009.05.024.

Lions J-L, Maday Y and Turinici G (2001) A parareal in time discretization of PDEs. *Comptes Rendus de l'Académie des Sciences Paris Series I* 332: 661–668. DOI 10.1016/S0764-4442(00)01793-6.

Liu F and Sosonkina M (2011) A multilevel parallelism support for multi-physics coupling. *Procedia Computer Science* 4: 261–270. DOI 10.1016/j.procs.2011.04.028.

Logg A (2003a) Multi-adaptive Galerkin methods for ODEs I. *SIAM Journal of Scientific Computing* 24(6): 1879–1902. DOI 10.1137/S1064827501389722.

Logg A (2003b) Multi-adaptive Galerkin methods for ODEs II: Implementation and applications. *SIAM Journal of Scientific Computing* 25(4): 1119–1141. DOI 10.1137/S1064827501389734.

Logg A (2004) *Multi-Adaptive Galerkin Methods for ODEs III: Existence and Stability*. Technical Report 2004-04, Chalmers Finite Element Center Preprint Series.

Logg A (2006) Multi-adaptive Galerkin methods for ODEs III: A priori estimates. *SIAM Journal on Numerical Analysis* 43(6): 2624–2646. DOI 10.1137/040604133.

Logg A, Mardal K-A, Wells GN, et al. (2012) *Automated Solution of Differential Equations by the Finite Element Method*. New York: Springer.

Long KR, Kirby RC and van Bloemen Waanders BG (2010) Unified embedded parallel finite element computations via software-based Fréchet differentiation. *SIAM Journal of Scientific Computing* 32(6): 3323–3351. DOI 10.1137/09076920X.

Lu G and Kaxiras E (2005) Overview of multiscale simulations of materials. In: Rieth M and Schommers W (eds), *Handbook of Theoretical and Computational Nanotechnology*, vol. 10. Valencia, CA: American Scientific Publishers, pp. 1–33.

Luitjens J, Worthen B, Berzins M, et al. (2008) Scalable parallel AMR for the Uintah multiphysics code. In: Bader D (ed.), *Petascale Computing Algorithms and Applications*. New York: Chapman and Hall.

Luo X, Stylianopoulos T, Barocas VH, et al. (2009) Multiscale computation for soft tissues with complex geometries. *Engineering with Computers* 25(1): 87–96. DOI 10.1007/s00366-008-0111-4.

McCourt M, Rognlien TD, McInnes LC, et al. (2012) Improving parallel scalability for edge plasma transport simulations with neutral gas species. *Computational Science and Discovery* 5: . DOI 10.1088/1749-4699/5/1/014012. Focus on Twenty Second International Conference on Numerical Simulations of Plasmas (ICNSP 2011).

McInnes LC, Allan B, Armstrong R, et al. (2006). Parallel PDE-based simulations using the Common Component Architecture.

In: *Numerical Solution of Partial Differential Equations on Parallel Computers (Lecture Notes in Computational Science and Engineering*, vol. 51). Berlin: Springer-Verlag, pp. 327–384. DOI 10.1007/3-540-31619-1_10.

Mahadevan V, Ragusa JC and Mousseau V (2009) Verification of multiphysics software: Space and time convergence studies for nonlinearly coupled applications. In: *Proceedings of the international conference on mathematics, computational methods and reactor physics*, Saratoga Springs, NY, 3–7 May 2009. La Grange Park, IL: American Nuclear Society.

Maman N and Farhat C (1995) Matching fluid and structure meshes for aeroelastic computations: A parallel approach. *Computers & Structures* 54(4): 779–785. DOI 10.1016/0045-7949(94)00359-B.

Mandel J (1993) Balancing domain decomposition. *Communications in Numerical Methods in Engineering* 9: 233–241. DOI 10.1002/cnm.1640090307.

Mandel J, Dohrmann CR and Tezaur R (2005) An algebraic theory for primal and dual substructuring methods by constraints. *Applied Numerical Mathematics* 54(2): 167–193. DOI 10.1016/j.apnum.2004.09.022.

Matthies HG and Steindorf J (2002) Partitioned but strongly coupled iteration schemes for nonlinear fluid–structure interaction. *Computers and Structures* 80: 1991–1999. DOI 10.1016/S0045-7949(02)00259-6.

Maxwell RM, Chow FK and Kollet SJ (2007) The groundwater–land-surface–atmosphere connection: Soil moisture effects on the atmospheric boundary layer in fully-coupled simulations. *Advances in Water Resources* 30(12): 2447–2466. DOI 10.1016/j.advwatres.2007.05.018.

Maxwell RM, Lundquist JK, Mirocha JD, et al. (2011) Development of a coupled groundwater–atmosphere model. *Monthly Weather Review* 139(1): 96–116. DOI 10.1175/2010MWR3392.1.

Maxwell RM and Miller NL (2005) Development of a coupled land surface and groundwater model. *Journal of Hydrometeorology* 6(3): 233–247. DOI 10.1175/JHM422.1.

May DA and Moresi L (2008) Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics. *Physics of the Earth and Planetary Interiors* 171(1–4): 33–47. DOI 10.1016/j.pepi.2008.07.036.

Meijerink J and van der Vorst HA (1977) An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation* 31: 148–162. DOI 10.1090/S0025-5718-1977-0438681-4.

Meller A, Nivon L, Brandin E, et al. (2000) Rapid nanopore discrimination between single polynucleotide molecules. *Proceedings of the National Academy of Sciences* 97: 1079–1084. DOI 10.1073/pnas.97.3.1079.

Michopoulos J, Tsompanopoulou P, Houstis E, et al. (2003) DDEMA: A data-driven environment for multiphysics applications. In: *Proceedings of ICCS 2003*, Melbourne, Australia (*Lecture Notes in Computer Science* vol. 2660, part 4). New York: Springer, pp. 309–318.

Mihalas D and Mihalas BW (1984) *Foundations of Radiation Hydrodynamics*. New York, Oxford: Oxford University Press.

Miller MC, Reus JF, Matzke RP, et al. (2004) *Smart Libraries: Best SQE Practices for Libraries with Emphasis on Scientific Computing*. Technical Report UCRL-JRNL-28636, Lawrence Livermore National Laboratory.

Miller RE and Tadmor EB (2002) The quasicontinuum method: Overview, applications and current directions. *Journal of Computer-Aided Materials Design* 9(3): 203–239. DOI 10.1023/A:1026098010127.

Minion ML (2003) Semi-implicit spectral deferred correction methods for ordinary differential equations. *Communications in Mathematical Sciences* 1(3): 471–500.

Minion ML (2004) Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. *Applied Numerical Mathematics* 48(3–4): 369–387. DOI 10.1016/j.apnum.2003.11.005. Workshop on Innovative Time Integrators for PDEs, Amsterdam, Netherlands, Nov. 25-27, 2002.

Mitchell K (2005) *The community Noah land-surface model user's guide, release 2.7.1.* Available at: http://www.emc.ncep.noaa.gov/mmb/gcp/noahlsm/Noah_LSM_USERGUIDE_2.7.1.htm.

Morgan RB (1996) On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Mathematics of Computation* 65(215): 1213–1230. DOI 10.1090/S0025-5718-96-00745-4.

Mosthaf K, Baber K, Flemisch B, et al. (2011) A coupling concept for two-phase compositional porous-medium and single-phase compositional free flow. *Water Resources Research* 47: . DOI 10.1029/2011WR010685.

Mousseau V (2004) Implicitly balanced solution of the two-phase flow equations coupled to nonlinear heat conduction. *Journal of Computational Physics* 200: 104–132. DOI 10.1016/j.jcp.2004.03.009.

MpCCI (2012) Multi-physics Coupling Code Interface. Homepage at: http://www.mpcci.de.

Musson L, Pawlowski R, Salinger A, et al. (2009) Multiphase reacting flow modeling of singlet oxygen generators for chemical oxygen iodine lasers. *Proceedings of SPIE* 7131: 71310J: 1–71310J:8.

Myra ES, Bludman SA, Hoffman Y, et al. (1987) The effect of neutrino transport on the collapse of iron stellar cores. *The Astrophysical Journal* 318: 744–759. DOI 10.1086/165408.

Nabarro FRN (1947) Dislocations in a simple cubic lattice. *Proceedings of the Physical Society of London* 59: 256–272. DOI 10.1088/0959-5309/59/2/309.

Newman C, Gaston D and Hansen G (2009a) Computational foundations for reactor fuel performance modeling. In: *American Nuclear Society 2009 international conference on advances in mathematics, computational methods, and reactor physics*, Saratoga Springs, NY.

Newman C, Hansen G and Gaston D (2009b) Three dimensional coupled simulation of thermomechanics, heat, and oxygen diffusion in nuclear fuel rods. *Journal of Nuclear Materials* 392: 6–15. DOI 10.1016/j.jnucmat.2009.03.035.

Nonaka A, Almgren A, Bell J, et al. (2010) MAESTRO: An adaptive low Mach number hydrodynamics algorithm for stellar flows. *The Astrophysical Journal Supplement Series* 188: 358–383. DOI 10.1088/0067-0049/188/2/358.

Notz P, Pawlowski R and Sutherland J (2012) Graph-based software design for managing complexity and enabling

concurrency in multiphysics PDE software. *ACM Transactions on Mathematical Software* 39(1): Article 1. DOI 10.1145/2382585.2382586.

Nuggehally MA, Shephard MS, Picu CR, et al. (2007) Adaptive model selection procedure for concurrent multiscale problems. *International Journal for Multiscale Computational Engineering* 5(5): 369–386. DOI 10.1615/IntJMultCompEng.v5.i5.20.

Oakley J (2004) Estimating percentiles of uncertain computer code outputs. *Journals of the Royal Statistical Society: Series C (Applied Statistics)* 53: 83–93. DOI 10.1046/j.0035-9254.2003.05044.x.

O'Bara RM, Beall MW and Shephard MS (2002) Attribute management system for engineering analysis. *Engineering with Computers* 18(4): 339–351. DOI 10.1007/s003660200030.

Oden JT and Prudhomme S (2002) Estimation of modeling error in computational mechanics. *Journal of Computational Physics* 182: 496–515. DOI 10.1006/jcph.2002.7183.

Oden JT, Prudhomme S, Romkes A, et al. (2006) Multiscale modeling of physical phenomena: Adaptive control of models. *SIAM Journal of Scientific Computing* 28(6): 2359–2389. DOI 10.1137/050632488.

Ollivier-Gooch C, Diachin LF, Shephard MS, et al. (2010) An interoperable, data-structure-neutral component for mesh query and manipulation. *ACM Transactions on Mathematical Software* 37(3): 29:1–29:28. DOI 10.1145/1824801.1864430.

OpenPALM (2012) Homepage at: http://www.cerfacs.fr/globc/PALM_WEB/.

Panday S and Huyakorn PS (2004) A fully coupled physically-based spatially-distributed model for evaluating surface/subsurface flow. *Advances in Water Resources* 27(4): 361–382. DOI 10.1016/j.advwatres.2004.02.016.

Park H, Gaston D, Kadioglu S, et al. (2009a) Tightly coupled multiphysics simulations for pebble bed reactors. In: *American Nuclear Society 2009 international conference on advances in mathematics, computational methods, and reactor physics*, Saratoga Springs, NY, 3–7 May 2009.

Park H, Knoll D, Gaston D, et al. (2010) Tightly coupled multiphysics algorithms for pebble bed reactors. *Nuclear Science and Engineering* 166(2): 118–133.

Park Y, Sudicky E, Panday S, et al. (2009b). Implicit subtime stepping for solving the nonlinear equations of flow in an integrated surface–subsurface system. *Vadose Zone Journal* 8(4): 825–836. DOI 10.2136/vzj2009.0013.

Parker S (2006) A component-based architecture for parallel multi-physics PDE simulation. *Future Generation Computer Systems* 22(1): 204–216. DOI 10.1016/j.future.2005.04.001.

Parker S, Guilkey J and Harman T (2006) A component-based parallel infrastructure for the simulation of fluid–structure interaction. *Engineering with Computers* 22(3–4): 277–292. DOI 10.1007/s00366-006-0047-5.

Pau G, Almgren A, Bell J, et al. (2009) A parallel second-order adaptive mesh algorithm for incompressible flow in porous media. *Philosophical Transactions of the Royal Society A* 367: 4633–4654. DOI 10.1098/rsta.2009.0160.

Pawlowski R (2012) Phalanx multiphysics assembly package. Homepage at: http://trilinos.sandia.gov/packages/phalanx/.

Pawlowski R, Bartlett R, Belcourt N, et al. (2011) *A Theory Manual for Multi-Physics Code Coupling in LIME*. Sandia Technical Report SAND2011-2195, Sandia National Laboratories. DOI 10.2172/1011710.

Pawlowski R, Phipps E and Salinger A (2012a) Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part I: Template-based generic programming. *Scientific Programming* 20(2): 197–219.

Pawlowski R, Phipps E, Salinger A, et al. (2012b) Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part II: Application to partial differential equation. *Scientific Programming* 20(3): 327–345.

Peierls R (1940) The size of a dislocation. *Proceedings of the Physical Society of London* 52: 34–37. DOI 10.1088/0959-5309/52/1/305.

Pember R, Howell L, Bell J, et al. (1998) An adaptive projection method for unsteady, low-Mach number combustion. *Combustion Science and Technology* 140(1–6): 123–168. DOI 10.1080/00102209808915770.

Pernice M and Tocci MD (2001) A multigrid-preconditioned Newton–Krylov method for the incompressible Navier–Stokes equations. *SIAM Journal of Scientific Computing* 23: 398–418. DOI 10.1137/S1064827500372250.

Pershing D, et al. (2010) Center for the Simulation of Accidental Fires and Explosions (C-SAFE). Homepage at: http://www.csafe.utah.edu/.

Peters A, Melchionna S, Kaxiras E, et al. (2010) Multiscale simulation of cardiovascular flows on the IBM Blue Gene/P: Full heart-circulation system at near red-blood cell resolution. In: *2010 ACM/IEEE international conference for high performance computing, networking, storage and analysis*. New Orleans, LA. DOI 10.1109/SC.2010.33.

Piacentini A, Morel T, Thévenin A, et al. (2011) Open-PALM: An open source dynamic parallel coupler. In: *Coupled problems 2011*, Kos Island, Greece.

Pointer WD, et al. (2012a) *Multi-physics reactor performance and safety simulations*. Available at: http://www.ne.anl.gov/capabilities/sharp/.

Pointer WD, et al. (2012b) *SHARP: Reactor performance and safety simulation suite*. Available at: http://www.ne.anl.gov/capabilities/sharp/SHARPbrochure_v7.pdf.

Pomraning GC (1973) *The Equations of Radiation Hydrodynamics*. Oxford, New York: Pergamon Press.

Post DE, Batchelor DB, Bramley RB, et al. (2004) *Report of the fusion simulation project steering committee*. Available at: http://w3.pppl.gov/usjapanim/FSPReport.pdf.

Post DE and Kendall RP (2004) Software project management and quality engineering practices for complex, coupled multiphysics, massively parallel computational simulations: Lessons learned from ASCI. *International Journal of High Performance Computing Applications* 18(4): 399–416. DOI 10.1177/1094342004048534.

Qu Y and Duffy CJ (2007) A semi-discrete finite-volume formulation for multi-process watershed simulation. *Water Resources Research* 43(W08419): 18. DOI 10.1029/2006WR005752.

Reece CE, Daly EF, Davis GK, et al. (2007) Diagnosis, analysis, and resolution of thermal stability issues with HOM couplers on prototype CEBAF SRF cavities. In: *Proceedings of SRF 2007*, Beijing, China, p. 656.

Reid JK (1971) On the method of conjugate gradients for the solution of large sparse systems of linear equations. In: Reid JK (ed.), *Large Sparse Sets of Linear Equations*. New York: Academic Press, pp. 231–254.

Remacle J-F and Shephard MS (2003) An algorithm oriented mesh database. *International Journal for Numerical Methods in Engineering* 58(2): 349–374. DOI 10.1002/nme.774.

Reynolds DR (2012) ARKODE webpage. Homepage at: http://faculty.smu.edu/reynolds/arkode.

Reynolds DR, Samtaney R and Woodward CS (2006) A fully implicit numerical method for single-fluid resistive magnetohydrodynamics. *Journal of Computational Physics* 219: 144–162. DOI 10.1016/j.jcp.2006.03.022.

Reynolds DR, Swesty FD and Woodward CS (2008) A Newton–Krylov solver for implicit solution of hydrodynamics in core collapse supernovae. *Journal of Physics: Conference Series* 125(1): 012085. DOI 10.1088/1742-6596/125/1/012085.

Richardson LF (1911) The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London, Series A* 210: 307–357. DOI 10.1098/rsta.1911.0009.

Richardson LF and Gaunt JA (1927) The deferred approach to the limit. *Philosophical Transactions of the Royal Society of London, Series A* 226: 299–361. DOI 10.1098/rsta.1927.0008.

Ringler T, Jacobsen DW, Gunzburger M, et al. (2011). Exploring a multi-resolution modeling approach within the shallow-water equations. *Monthly Weather Review* 139: 3348–3368. DOI 10.1175/MWR-D-10-05049.1.

Robinson BA, Viswanathan HS and Valocchi AJ (2000) Efficient numerical techniques for modeling multicomponent ground-water transport based upon simultaneous solution of strongly coupled subsets of chemical components. *Advances in Water Resources* 23(4): 307–324. DOI 10.1016/S0309-1708(99)00034-2.

Ropp D and Shadid J (2009) Stability of operator splitting methods for systems with indefinite operators: Advection–diffusion–reaction systems. *Journal of Computational Physics* 228(9): 3508–3516. DOI 10.1016/j.jcp.2009.02.001.

Ropp D, Shadid J and Ober C (2004) Studies of the accuracy of time integration methods for reaction–diffusion equations. *Journal of Computational Physics* 194(2): 544–574. DOI 10.1016/j.jcp.2003.08.033.

Rosner R, et al. (2010) *The opportunities and challenges of exascale computing*. Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee on Exascale Computing, Office of Science, U.S. Department of Energy. Available at: http://science.energy.gov/~/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf.

Rosner R, et al. (2012). Center for Exascale Simulation of Advanced Reactors (CESAR). Homepage at: http://cesar.mcs.anl.gov/.

Ross M (2006) *Coupling and simulation of acoustic fluid–structure interaction systems using localized Lagrange multipliers*. PhD Thesis, University of Colorado.

Rubin J (1983) Transport of reacting solutes in porous media: Relation between mathematical nature of problem formulation and chemical nature of reactions. *Water Resources Research* 19(5): 1231–1252. DOI 10.1029/WR019i005p01231.

Ruuth SJ (1995) Implicit-explicit methods for reaction-diffusion. *Journal of Mathematical Biology* 34: 148–176. DOI 10.1007/BF00178771.

Saad Y (1993) A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal of Scientific Computing* 14(2): 461–469. DOI 10.1137/0914028.

Saad Y and Schultz MH (1986) GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 7: 856–869. DOI 10.1137/0907058.

Saad Y and van der Vorst HA (2000) Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics* 123(1–2): 1–33. DOI 10.1016/S0377-0427(00)00412-X.

Sacks J, Welch WJ, Mitchell TJ, et al. (1989) Design and analysis of computer experiments. *Statistical Science* 4: 409–423. DOI 10.1214/ss/1177012413.

Sahu R, Panthaki MJ and Gerstle WH (1999) An object-oriented framework for multidisciplinary, multi-physics computational mechanics. *Engineering with Computers* 15(1): 105–125. DOI 10.1007/s003660050008.

Sanchez-Palencia E (1980) *Non-Homogeneous Media and Vibration Theory (Lecture Notes in Physics*, vol. 127). Springer-Verlag.

Sandu A and Constantinescu EM (2009) Multirate explicit Adams methods for time integration of conservation laws. *Journal of Scientific Computing* 38(2): 229–249. DOI 10.1007/s10915-008-9235-3.

Savcenco V, Hundsdorfer W and Verwer J (2007) A multirate time stepping strategy for stiff ordinary differential equations. *BIT* 47(1): 137–155. DOI 10.1007/s10543-006-0095-7.

Sawan M, Wilson P, Tautges T, et al. (2009) Application of CAD-neutronics coupling to geometrically complex fusion systems. In: *SOFE 2009: The 23rd IEEE/NPSS symposium on fusion engineering*, pp. 1–6.

Schäfer M, Sternel D, Becker G, et al. (2010) Efficient numerical simualtion and optimization of fluid–structure interaction. In: Bungartz H-J, Mehl M and Schäfer M (eds) *Fluid Structure Interaction II (Lecture Notes in Computational Science and Engineering*, vol. 73). Berlin: Springer, pp. 131–158.

Schmidt R, Belcourt K, Clarno K, et al. (2010) *Foundational Development of an Advanced Nuclear Reactor Integrated Safety Code*. Sandia Technical Report SAND2010-0878, Sandia National Laboratories.

Scholz D, Kollmannsberger S, Düster A, et al. (2006) Thin solids for fluid–structure interaction. In: Bungartz H-J and Schäfer M (eds) *Fluid–Structure Interaction (Lecture Notes in Computational Science and Engineering*, vol. 53). Berlin: Springer, pp. 294–335.

Schubert G, Turcotte DL and Olson P (2001) *Mantle Convection in the Earth and Planets*. Cambridge: Cambridge University Press.

Seol ES and Shephard MS (2006) Efficient distributed mesh data structure for parallel automated adaptive analysis. *Engineering with Computers* 22(3–4): 197–213. DOI 10.1007/s00366-006-0048-4.

Shadid J, Cyr E, Pawlowski R, et al. (2010a) Initial performance of fully-coupled AMG and approximate block factorization preconditioners for solution of implicit FE resistive MHD. In: *Fifth European conference on computational fluid dynamics*, Lisbon, Portugal.

Shadid J, Pawlowski R, Banks J, et al. (2010b) Towards a scalable fully-implicit fully-coupled resistive MHD formulation with stabilized FE methods. *Journal of Computational Physics* 229(20): 7649–7671. DOI 10.1016/j.jcp.2010.06.018.

Shampine LF and Gear CW (1979) A user's view of solving stiff ordinary differential equations. *SIAM Review* 21(1): 1–17. DOI 10.1137/1021001.

Shampine LF, Sommeijer BP and Verwer JG (2006) IRKC: An IMEX solver for stiff diffusion-reaction PDEs. *Journal of Computational and Applied Mathematics* 196(2): 485–497. DOI 10.1016/j.cam.2005.09.014.

Siegel A, Tautges T, Caceres A, et al. (2007) Software design of SHARP. In: *Joint international topical meeting of mathematics & computation and supercomputing in nulear applications*, Monterey, CA.

Simon H, Zacharia T, Stevens R, et al. (2007) *Modeling and simulation at the exascale for energy and the environment*. Office of Science, U.S. Department of Energy. Available at: http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Townhall.pdf.

Smith B, Bjørstad P and Gropp WD (1996). *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge: Cambridge University Press.

Smith B, McInnes LC, Constantinescu E, et al. (2012) PETSc's software strategy for the design space of composable extreme-scale solvers. In: *DOE exascale research conference*, Portland, OR, 16–18 April 2012. Preprint no. ANL/MCS-P2059-0312. Available at: http://www.mcs.anl.gov/uploads/cels/papers/P2059-0312.pdf.

Smith M, Marin-Lafleche A, Yang W, et al. (2011) Method of characteristics development targeting the high performance Blue Gene/P computer at Argonne National Laboratory. In: *International conference on advances in mathematics, computational methods, and reactor physics (M&C)*.

Smith GS, Tadmor EB, Bernstein N, et al. (2001) Multiscale simulation of silicon nanoindentation. *Acta Materialia* 49: 4089–4101. DOI 10.1016/S1359-6454(01)00267-1.

Söderlind G (1998) The automatic control of numerical integration. *CWI Quarterly* 11(1): 55–74.

Söderlind G (2003) Digital filters in adaptive time-stepping. *ACM Transactions on Mathematical Software* 29(1): 1–26. DOI 10.1145/641876.641877.

Solin P, et al. (2012) *Hermes project*. Details available at: http://en.wikipedia.org/wiki/Hermes_Project.

Solin P, Cerveny J, Dubcova L, et al. (2010) Monolithic discretization of linear thermoelasticity problems via adaptive multimesh FEM. *Journal of Computational and Applied Mathematics* 234(7): 2350–2357. DOI 10.1016/j.cam.2009.08.092. Fourth International Conference on Advanced Computational Methods in Engineering (ACOMEN 2008).

Solomon SD, Qin D, Manning M, et al. (2007) *Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge: Cambridge University Press.

Spiegelman M (1993a) Flow in deformable porous media, Part 1: Simple analysis. *Journal of Fluid Mechanics* 247: 17–38. DOI 10.1017/S0022112093000369.

Spiegelman M (1993b) Flow in deformable porous media, Part 2: Numerical analysis – the relationship between shock waves and solitary waves. *Journal of Fluid Mechanics* 247: 39–63. DOI 10.1017/S0022112093000370.

Steefel CI and MacQuarrie KTB (1996) Approaches to modeling of reactive transport in porous media. *Reviews in Mineralogy* 34: 83–129.

Stevens R (2011) Institute for Computing in Science (ICiS). Homepage at: http://www.icis.anl.gov/.

Stewart JR and Edwards HC (2004) A framework approach for developing parallel adaptive multiphysics applications. *Finite Elements in Analysis and Design* 40(12): 1599–1617. DOI 10.1016/j.finel.2003.10.006.

Storm AJ, Storm C, Chen J, et al. (2005) Fast DNA translocation through a solid-state nanopore. *Nano Letters* 5: 1193–1197. DOI 10.1021/nl048030d.

Strang G (1968) On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis* 5: 506–517. DOI 10.1137/0705041.

Succi S (2001) *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford: Clarendon Press.

Sun Y, Folwell N, Li Z, et al. (2002) High precision accelerator cavity design using the parallel eigensolver Omega3P. In: *Proceedings of the 18th annual review of progress in applied computational electromagnetics*.

Sussman A (2006) Building complex coupled physical simulations on the grid with InterComm. *Engineering with Computers* 22(3–4): 311–323. DOI 10.1007/s00366-006-0037-7.

Swesty FD and Myra ES (2009) A numerical algorithm for modeling multigroup neutrino-radiation hydrodynamics in two spatial dimensions. *The Astrophysical Journal Supplement Series* 181: 1–52. DOI 10.1088/0067-0049/181/1/1.

Szyld DB and Vogel JA (2001) FQMR: A flexible quasi-minimal residual method with inexact preconditioning. *SIAM Journal of Scientific Computing* 23(2): 363–380. DOI 10.1137/S106482750037336X.

Tadmor EB, Ortiz M and Phillips R (1996) Quasicontinuum analysis of defects in solids. *Philosophical Magazine A* 73: 1529–1563. DOI 10.1080/01418619608243000.

Tadmor EB, Waghmare UV, Smith GS, et al. (2002) Polarization switching in PbTiO3: An ab initio finite element simulation. *Acta Materialia* 50: 2989–3002. DOI 10.1016/S1359-6454(02)00127-1.

Tang W, Keyes DE, Sauthoff N, et al. (2009) *Scientific grand challenges: Fusion energy and the role of computing at the extreme scale*. Office of Science, U.S. Department of Energy. Available at: http://extremecomputing.labworks.org/fusion/PNNL_Fusion_final19404.pdf.

Tarduno J, Bunge H, Sleep N, et al. (2009) The bent Hawaiian-Emperor hotspot track: Inheriting the mantle wind. *Science* 324(5923): 50–53. DOI 10.1126/science.1161256.

Tartakovsky AM and Scheibe TD (2011) Dimension reduction numerical closure method for advection–diffusion–reaction systems. *Advances in Water Resources* 34(12): 1616–1626. DOI 10.1016/j.advwatres.2011.07.011.

Tartakovsky AM, Tartakovsky DM, Scheibe TD, et al. (2008) Hybrid simulations of reaction-diffusion systems in porous media. *SIAM Journal of Scientific Computing* 30(6): 2799–2816. DOI 10.1137/070691097.

Tautges T, et al. (2011a) *SISIPHUS: Scalable ice-sheet solvers and infrastructure for petascale, high-resolution, unstructured simulations*. Available at: http://trac.mcs.anl.gov/projects/sisiphus/wiki.

Tautges T and Caceres A (2009) Scalable parallel solution coupling for multiphysics reactor simulation. *Journal of Physics: Conference Series* 180: . DOI 10.1088/1742-6596/180/1/012017.

Tautges T, Kraftcheck J, Bertram N, et al. (2011b) Mesh interface resolution and ghost exchange in a parallel mesh representation. In: *26th IEEE international parallel & distributed processing symposium*, Shanghai, China.

Tautges T, Kraftcheck J, Smith B, et al. (2011c) MOAB: Mesh-Oriented datABase, version 4.0. Available at: http://trac.mcs.anl.gov/projects/ITAPS/wiki/MOAB.

Tautges T, Meyers R, Merkley K, et al. (2004) *MOAB: A Mesh-Oriented Database*. Technical Report SAND2004-1592, Sandia National Laboratories.

Tonks M, Gaston D, Permann C, et al. (2010) A coupling methodology for mesoscale-informed nuclear fuel performance codes. *Nuclear Engineering and Design* 240(10): 2877–2883. DOI 10.1016/j.nucengdes.2010.06.005.

Tonks M, Hansen G, Gaston D, et al. (2009) Fully-coupled engineering and mesoscale simulations of thermal conductivity in UO_2 fuel using an implicit multiscale approach. *Journal of Physics: Conference Series* 180(1): 012078. DOI 10.1088/1742-6596/180/1/012078.

Toselli A and Widlund OB (2005) *Domain Decomposition Methods – Algorithms and Theory (Springer Series in Computational Mathematics)*. New York: Springer.

Trottenberg U, Oosterlee C and Schüller A (2001) *Multigrid*. New York: Academic Press.

Turner NJ and Stone JM (2001) A module for radiation hydrodynamic calculations with ZEUS-2D using flux-limited diffusion. *The Astrophysical Journal Supplement Series* 135: 95–107. DOI 10.1086/321779.

Valocchi AJ and Malmstead M (1992) Accuracy of operator splitting for advection-dispersion-reaction problems. *Water Resources Research* 28(5): 1471–1476. DOI 10.1029/92WR00423.

Van Brummelen EH (2009) Added mass effects of compressible and incompressible flows in fluid–structure interaction. *Journal of Applied Mechanics* 76(2): 021206. DOI 10.1115/1.3059565.

Van Brummelen EH (2010) Partitioned iterative solution methods for fluid–structure interaction. *International Journal for Numerical Methods in Fluids* 65: 3–27. DOI 10.1002/fld.2465.

Van der Holst B, Toth G, Sokolov IV, et al. (2011) CRASH: A block-adaptive-mesh code for radiative shock hydrodynamics–-implementation and verification. *The Astrophysical Journal Supplement Series* 194: 23. DOI 10.1088/0067-0049/194/2/23.

VanderKwaak JE and Loague K (2001) Hydrologic-response simulations for the R-5 catchment with a comprehensive physics-based model. *Water Resources Research* 37: 999–1013. DOI 10.1029/2000WR900272.

Van der Vorst HA (1992) BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 13: 631–644. DOI 10.1137/0913035.

Van Keken PE, King SD, Schmeling H, et al. (1997) A comparison of methods for the modeling of thermochemical convection. *Journal of Geophysical Research*. DOI 10.1029/97JB01353

Van Straalen B, Colella P, Graves D, et al. (2011) Petascale block-structured AMR applications without distributed meta-data. In: Jeannot E, Namyst R and Roman J (eds) *Euro-Par 2011 Parallel Processing (Lecture Notes in Computer Science*, vol. 6853). Berlin: Springer, pp. 377–386. DOI 10.1007/978-3-642-23397-5_37.

Verwer JG and Sommeijer BP (2004) An implicit-explicit Runge–Kutta–Chebyshev scheme for diffusion-reaction equations. *SIAM Journal of Scientific Computing* 25(5): 1824–1835. DOI 10.1137/S1064827503429168.

Vogel JA (2007) Flexible BiCG and flexible Bi-CGSTAB for nonsymmetric linear systems. *Applied Mathematics and Computation* 188(1): 226–233. DOI 10.1016/j.amc.2006.09.116.

Voter AF (1997a) A method for accelerating the molecular dynamics simulation of infrequent events. *Journal of Chemical Physics* 106: 4665–4677. DOI 10.1063/1.473503.

Voter AF (1997b) Hyperdynamics: Accelerated molecular dynamics of infrequent events. *Physical Review Letters* 78: 3908–3911. DOI 10.1103/PhysRevLett.78.3908.

Wagner G and Liu WK (2003) Coupling of atomic and continuum simulations using a bridging scale decomposition. *Journal of Computational Physics* 190: 249–274. DOI 10.1016/S0021-9991(03)00273-0.

Wall WA, Mok DP and Ramm E (2001) Accelerated iterative substructuring schemes for instationary fluid–structure interaction. *Computational Fluid and Solid Mechanics*: 1325–1328.

Wang L, Lee J, Anitescu M, et al. (2011a) A differential variational inequality approach for the simulation of heterogeneous materials. In: *Proceedings of the SciDAC 2011 conference*. DOI 10.1002/fld.2556.

Wang K, Rallu A, Gerbeau J-F, et al. (2011b) Algorithms for interface treatment and load computation in embedded boundary methods for fluid and fluid–structure interaction problems. *International Journal for Numerical Methods in Fluids* 67: 1175–1206.

Washington W, et al. (2008) *Scientific grand challenges: Challenges in climate change science and the role of computing at the extreme scale*. Office of Science, U.S. Department of Energy. Available at: http://extremecomputing.labworks.org/climate/reports/ClimateReport_6-24-09.pdf.

Watts AB (2001) *Isostasy and Flexure of the Lithosphere*. Cambridge: Cambridge University Press.

Whiteley JP, Gillow K, Tavener SJ, et al. (2011). Error bounds on block Gauss–Seidel solutions of coupled multiphysics problems. *International Journal for Numerical Methods in Engineering* 88(12): 1219–1237. DOI 10.1002/nme.3217.

Wigton LB, Yu NJ and Young DP (1985) GMRES acceleration of computational fluid dynamics codes. In: *AIAA 7th computational fluid dynamics conference*.

Williamson R and Knoll D (2009) Enhancing the ABAQUS thermomechanics code to simulate steady and transient fuel rod behavior. In: *Proceedings of TopFuel 2009*.

Wohlmuth B (2011) Variationally consistent discretization schemes and numerical algorithms for contact problems. *Acta Numerica* 20: 569–734. DOI 10.1017/S0962492911000079.

Woodward C, et al. (2012) SUNDIALS webpage. Homepage at: https://computation.llnl.gov/casc/sundials.

Xiao S and Belystschko T (2004) A bridging domain method for coupling continua with molecular dynamics. *Computer Methods in Applied Mechanics and Engineering* 193: 1645–1669. DOI 10.1016/j.cma.2003.12.053.

Xiao L, Ko K, Lee KH, et al. (2011) Effects of elliptically deformed cell shape in the Cornell ERL cavity. In: *15th international conference on RF superconductivity*, Chicago, IL.

Xiu D (2010) *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton, NJ: Princeton University Press.

Yeckel A, Lun L and Derby JJ (2009) An approximate block Newton method for coupled iterations of nonlinear solvers: Theory and conjugate heat transfer applications. *Journal of Computational Physics* 228(23): 8566–8588. DOI 10.1016/j.jcp.2009.08.003.

Yeh GT and Tripathi VS (1989) A critical evaluation of recent developments in hydrogeochemical transport models of reactive multichemical components. *Water Resources Research* 25(1): 93–108. DOI 10.1029/WR025i001p00093.

Zhang F, Docan C, Parashar M, et al. (2011a) Enabling multi-physics coupled simulations within the PGAS programming framework. In: *The 11th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid)*, pp. 84–93. DOI 10.1109/CCGrid.2011.73.

Zhang W, Howell L, Almgren A, et al. (2011b) CASTRO: A new compressible astrophysical solver. II. Gray radiation hydrodynamics. *The Astrophysical Journal Supplement Series* 196(2):. DOI 10.1088/0067-0049/196/2/20.

Zhong X (1996) Additive semi-implicit Runge-Kutta methods for computing high-speed nonequilibrium reactive flows. *Journal of Computational Physics* 128: 19–31. DOI 10.1006/jcph.1996.0193.

Zingale M, Almgren A, Bell J, et al. (2009) Low Mach number modeling of type Ia supernovae. IV. White dwarf convection. *The Astrophysical Journal* 704: 196–210. DOI 10.1088/0004-637X/704/1/196.

Zwart SP, McMillan S, Nualláin B, et al. (2008) A multiphysics and multiscale software environment for modeling astrophysical systems. In: *ICCS '08: Proceedings of the 8th international conference on computational science, part II*. Berlin: Springer-Verlag, pp. 206–216. DOI 10.1007/978-3-540-69387-1_23.