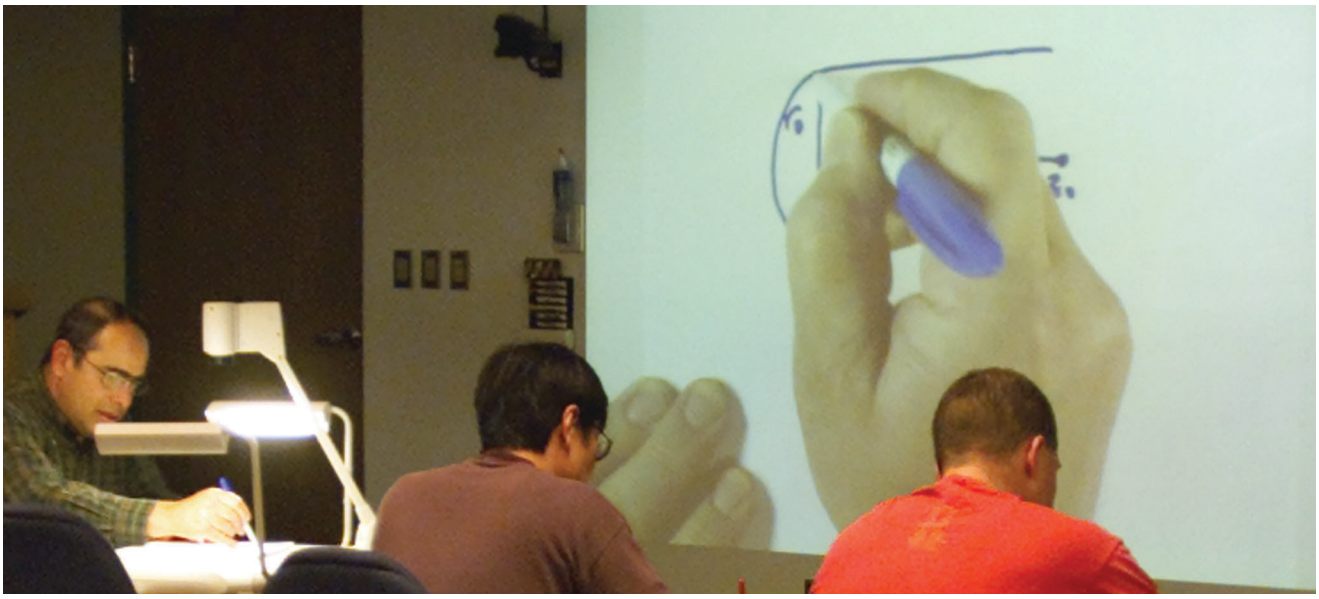

Massively Parallel Multiphysics Code Development

Jim E. Morel

The Advanced Simulation and Computing (ASCI) program is developing very large massively parallel multiphysics codes for reliably simulating nuclear weapons performance in the absence of nuclear testing. The task of developing multiphysics codes for the weapons program has always been a daunting one. A huge system of time-dependent, coupled nonlinear equations must be solved. These equations model many different types of physics. It is highly desirable, if not essential, that the solution process for such a system be decomposed into a series of steps, with each step consisting of the solution of equations associated with a single type of physics. Such an approach enables the code to be assembled with largely independent monophysics modules. This property is critical when one considers that essentially no one on a team is an expert in all the types of physics modeled in the code. Traditionally, only one or two people on a weapons code team had detailed knowledge of the coupling required between all the different types of physics in the code. Most of the teams consisted of individual experts in a single type of physics that contributed to a monophysics component. Today, only the size of the teams is different. An ASCI code team generally consists of subteams, rather than individuals, who are responsible for monophysics modules. The numerical technique that has traditionally been used to decompose the solution process into a sequence of essentially monophysics steps is still used in the current generation of ASCI codes. It is called operator splitting.

To demonstrate this concept, we need to review some basic concepts of temporal discretization. Nonlinear systems are generally solved by using a linearization process coupled with an iteration on the nonlinear terms. More specifically, the nonlinear equations are approximated with linear equations. After each solution of the linear equations, the nonlinear terms are updated. The process is then repeated until the nonlinear solution is converged. To understand operator splitting, we need consider only a set of linear equations. However, we must first review some basic concepts of temporal discretization. Although equations are generally discretized in all variables, we need not explicitly consider the other discretizations to illustrate the necessary points. For instance, let us consider a generic time-dependent linear system:

$$\frac{\partial f}{\partial t} = Af, \tag{1}$$

where f is the unknown, t is time, and A is a linear operator. A fully explicit time discretization is denoted as follows:

$$\frac{f^{n+1}-f(n)}{\Delta t}=Af^{n}, \tag{2}$$

where n is the time index, $\Delta t = t^{n+1} - t^n$ is the time step, t^n is the initial time associated with a time step, and t^{n+1} is the final time. Solving the explicit equation is generally inexpensive because one need only apply the operator A to f^n:

$$f^{n+1}=f^{n}+\Delta tAf^{n} \ . \tag{3}$$

However, explicit methods are generally unstable unless a sufficiently small time step is used. This restriction is acceptable for certain types of physics (for example, for hydrodynamics calculations with strong shocks), but it may be prohibitively expensive for others (for example, for thermal radiation transport). To obtain an unconditionally stable solution technique, one must generally use a fully implicit temporal discretization:

$$\frac{f^{n+1}-f(n)}{\Delta t}=Af^{n+1} \ . \tag{4}$$

The solution of the implicit equation is generally much more expensive than the solution of the explicit equation because one must invert an operator and apply it to f^n:

$$f^{n+1}=\left(I-\Delta tA\right)^{-1}f^{n} \ . \tag{5}$$

Suppose that we have two coupled equations. For instance, let us consider typical equations for the electron and ion temperatures in a plasma:

$$C_{ve}\frac{\partial T_e}{\partial t}=\vec{\nabla}\cdot K_e\vec{\nabla}T_e+\alpha(T_i-T_e) \ , \tag{6a}$$

and

$$C_{vi}\frac{\partial T_i}{\partial t}=\vec{\nabla}\cdot K_i\vec{\nabla}T_i+\alpha(T_e-T_i) \ , \tag{6b}$$

where T_e is the electron temperature, T_i is the ion temperature, C_{ve} and C_{vi} are the electron and ion heat capacities, respectively, K_e and K_i are the electron and ion conduction coefficients, respectively, and α is the coupling coefficient for internal energy exchange between the electron and ion fields. Modern computers can easily solve this system using a fully implicit temporal discretization:

$$\frac{C_{ve}}{\Delta t}\left(T_e^{n+1}-T_e^{n}\right)=\vec{\nabla}\cdot K_e\vec{\nabla}T_e^{n+1}+\alpha\left(T_i^{n+1}-T_e^{n+1}\right) \ , \tag{7a}$$

and

$$\frac{C_{vi}}{\Delta t}\left(T_i^{n+1}-T_i^{n}\right)=\vec{\nabla}\cdot K_i\vec{\nabla}T_i^{n+1}+\alpha\left(T_e^{n+1}-T_i^{n+1}\right) \ , \tag{7b}$$

However, this solution was not always easy to obtain. Operator splitting was once routinely used to reduce the solution of Equations (6a) and (6b) to a sequence of simpler solutions. In particular, a conduction calculation was first performed for the electrons,

$$\frac{C_{ve}}{\Delta t}\left(T_e^{n+\frac{1}{2}}-T_e^{n}\right)=\vec{\nabla}\cdot K_e\vec{\nabla}T_e^{n+\frac{1}{2}} \ , \tag{8}$$

followed by a conduction calculation for the ions,

$$\frac{C_{vi}}{\Delta t}\left(T_i^{n+\frac{1}{2}}-T_i^{n}\right)=\vec{\nabla}\cdot K_i\vec{\nabla}T_i^{n+\frac{1}{2}} \ , \tag{9}$$

followed by a local calculation of the coupling between the unknowns,

$$\frac{C_{ve}}{\Delta t}\left(T_e^{n+1}-T_e^{n+\frac{1}{2}}\right)=\alpha\left(T_i^{n+1}-T_e^{n+1}\right) \ , \tag{l0a}$$

and

$$\frac{C_{vi}}{\Delta t}\left(T_i^{n+1}-T_i^{n+\frac{1}{2}}\right)=\alpha\left(T_e^{n+1}-T_i^{n+1}\right) \ . \tag{l0b}$$

We refer to Equation (10b) as a local calculation because this particular type of coupling between temperatures leads to discrete equations that are independent in each spatial cell, as opposed to Equations (8) and (9), which involve coupling between adjacent cells. Compared with equations containing spatial coupling, local equations are generally very easy to solve and highly amenable to parallelization. If we add Equations (8) through (l0b), we obtain a set of difference equations that are "semi-implicit" in that all the operators are applied to unknowns at advanced times:

$$\frac{C_{ve}}{\Delta t}\left(T_e^{n+1}-T_e^{n}\right)=\vec{\nabla}\cdot K_e\vec{\nabla}T_e^{n+\frac{1}{2}}+\alpha\left(T_i^{n+1}-T_e^{n+1}\right) \ , \tag{11a}$$

and

$$\frac{C_{vi}}{\Delta t}\left(T_i^{n+1}-T_i^{n}\right)=\vec{\nabla}\cdot K_i\vec{\nabla}T_i^{n+\frac{1}{2}}+\alpha\left(T_e^{n+1}-T_i^{n+1}\right) \ . \tag{11b}$$

Because each solution step is fully implicit, this entire process is unconditionally stable. In general, the split solution is nearly as accurate as a fully implicit solution as long as the contributions from two or more steps are not nearly equal and opposite. If this is the case, time steps must be taken that can be extremely small relative to those required with a fully implicit discretization. Also, difficulties may be encountered in certain asymptotic limits. In recent years, an alternative to operator splitting has emerged that, in principle, can be used to solve large multiphysics systems of equations in a fully implicit manner. This technique is called the Newton-Krylov method. I will not discuss this method in detail here, but suffice it to say that it is not sufficiently mature to be used in an ASCI code project. However, it is very promising and may become the solution method of choice in the long term. In the short term, ASCI projects will continue to rely on operator splitting, and ASCI

researchers will attempt to better understand the deficiencies of operator splitting and eliminate them.

Given the previous example, it is not difficult to see that operator splitting can also be used to separate coupled multiphysics calculations into a set of monophysics solution steps together with local coupling steps. For instance, in a radiation-hydrodynamics calculation, one might first perform a hydrodynamics calculation, followed by a radiation transport calculation, followed by a local calculation of the coupling between the hydrodynamics and transport unknowns. This approach enables the development of essentially independent hydrodynamics and radiation transport software modules, together with a relatively simple module for coupling them. However, the implication here is not that the hydrodynamics and radiation transport teams can proceed completely independently of one another and then do the coupling after their respective modules are finished. Considerable planning and coordination are required before the software is written to ensure that the respective numerical treatments are compatible. For instance, if the material temperatures are assumed to be located at cell centers in the hydrodynamics equations, it is much easier to couple the modules and probably more accurate overall if the same assumption is made for the radiation transport.

The planning and coordination that must be achieved to ensure compatibility between physics modules are much more complicated with massively parallel computers than they were with serial and vector computers. The reason is that on multiprocessor distributed-memory computers, different physics modules often require different data partitionings on the processors for optimal performance. Distributed-memory machines store data on each processor. Data partitioning is simply the mapping of data to the processors on which they will be stored. In many instances, each datum can be uniquely associated with a single spatial grid cell. In such instances, all the data can be partitioned simply by partitioning the spatial grid itself, that is, by mapping each spatial cell (and hence the data associated with that cell) to a processor. Some of the information about a spatial grid partitioning is easily visualized. In particular, we can easily see what cells are mapped to the same processor by first assigning a unique color to each processor and then assigning a processor color to each cell in accordance with the cell-to-processor mapping. This information is generally referred to as the spatial domain decomposition. For instance, a typical domain decomposition for a hydrodynamics calculation on a three-dimensional (3-D) rectangular mesh is shown in Figure 1(a), and a typical domain decomposition for a radiation transport calculation on a 3-D rectangular mesh is shown in Figure 1(b). The partitionings are quite different. Thus, data that are shared by the hydrodynamics and transport calculations must be repartitioned during every time step at some point between the hydrodynamics and transport calculations. This requirement clearly complicates the coupling of physics modules. At one time, it was thought that such repartitioning would be prohibitively expensive. However, experience with ASCI codes indicates that repartitioning is not a problem as long as it occurs only between the execution of modules that do a significant amount of computational work. This is certainly the case for hydrodynamics and radiation modules.

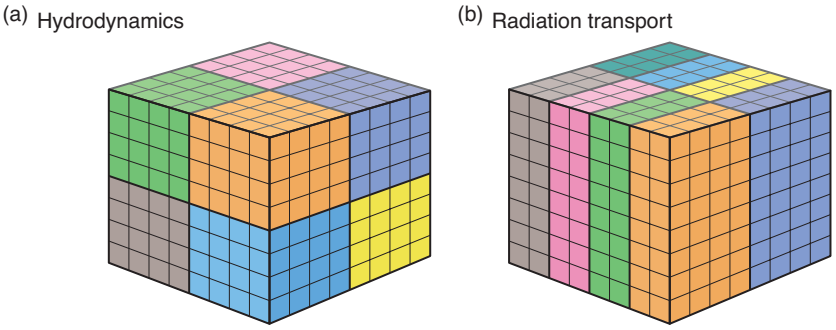

Figure 1. Spatial Domain Decomposition for Hydrodynamics vs Radiation Transport

A 3-D rectangular mesh is divided into eight computational domains (denoted by different colors), and each is assigned to a different processor. The optimal division for a hydrodynamics calculation (a) is quite different from that for radiation transport (b).

Another area in which massively parallel computing has significantly complicated multiphysics code development is the process of programming itself. On massively parallel computers, one must be concerned with moving data between processors while computing. This requirement adds another layer of complexity to the programming process that was not present with scalar and vector computers. A physicist working on an ASCI code team today requires much more computer science and advanced programming knowledge than a physicist working on a traditional serial or vector weapons code. This feature can be a problem for new hires coming onto code teams because they can require considerable training before being able to contribute effectively. Although there is a formal education program for training new designers in the weapons program, there is no formal education program to train new software developers. Efforts have been made to develop software frameworks that allow individuals to write parallel programs without a high level of computer science knowledge, but such approaches have not yet been effective for the large multiphysics programs written within the ASCI projects.

Finally, the ASCI program has been

Teaching Radiation Transport

Radiation transport, a subject rarely taught at universities, is very important to the development of the ASCI multiphysics codes for nuclear weapons. To help train young people in this field, the author initiated a graduate-level class in numerical methods for radiation transport. The class, offered for credit by the Chemical and Nuclear Engineering Department of the University of New Mexico, is taught at Los Alamos and is received simultaneously at three remote sites through a new technology, Access Grid Web-based teleconferencing. The Access Grid node at Los Alamos, one of more than 300 nodes worldwide, is run by the Advanced Computing Laboratory as part of its effort in long-distance communication. The photo shows a class in progress. On the wall are projected the classrooms at the three remote sites—University of New Mexico and Sandia National Laboratories at Albuquerque and Livermore. The inset shows node operator Cindy Sievers.

asked to deliver new code capabilities in a time frame much shorter than that associated with traditional weapons code development projects. The assumption was made that this goal would be possible because each project team would consist of several tens of individuals. However, this increase in team size was coupled with our traditional code development processes. These processes, which worked well for small teams, have failed to scale with large teams. No ASCI project team has yet found a way to efficiently utilize all its team members. This is really a management problem rather than a technical problem, but it is as difficult and as important as any technical problem faced in ASCI. Furthermore, this problem is clearly exacerbated by the fact that ASCI project teams often have no time to investigate new development processes because they are struggling to make milestones. However, they may be struggling to make milestones because they do not have adequate processes. The latest ASCI strategy at the Laboratory calls for the investigation of new code-development software environments and associated code-development processes. We hope to leverage some of the work done in this regard by other high-performance computing programs funded by the Department of Energy, such as the Scientific Discovery through Advanced Computing (SIDAC) Program. ■

Jim Morel received a B.S. in mathematics in 1972 from Louisiana State University and a Ph.D. in nuclear engineering from the University of New Mexico in 1979. He served as a nuclear research officer at the Air Force Weapons Laboratory from 1974 to 1976, was a staff member at Sandia National Laboratories from 1976 to 1984, and joined the staff at Los Alamos National Laboratory in 1984, where he is currently employed. His research interests include mimetic discretization techniques, multilevel solution techniques, and associated parallel algorithms for neutral and charged-particle transport and diffusion. Over the last 10 years, Jim has developed numerical transport and diffusion methods for 3-D unstructured meshes. He is now beginning to work on adaptive-mesh methods.