Mathematical design of multi-domain multi-physics frameworks

Bobby Philip, Oak Ridge National Laboratory, philipb@ornl.gov
Panel D: Multiphysics and multiscale couplings, Secondary: C, G

I begin with a paraphrase from an interview[1] with the recipient of the 2013 Turing Award, Dr. Leslie Lampert. In response to the question *"More recently, you have worked on ways to improve how software is built. Whats wrong with how its done now?"* Dr. Lampert responds:

"There's something about the culture of software that has impeded the use of specification. We have a wonderful way of describing things precisely thats been developed over the last couple of millennia, called mathematics. I think that's what we should be using as a way of thinking about what we build."

As members of the scientific computing community who work at the intersection of applied mathematics, computer science, and physical sciences Dr. Lampert's call to the basics seems particularly relevant. Computational plasma physics like many other disciplines requires the ability to represent a range of physical models that vary in complexity. Complexity appears for a variety of reasons including but not limited to the sophistication of the physical models, the number of spatial and temporal scales, the dimensionality of the problem, the number of physical domains, and couplings between physics, models, and domains at different scales and dimensions. Naturally this has led to a a whole "forest" of approaches that are important in their own right and range from reductionistic approaches that focus on a single physics at a particular scale to systemic approaches that attempt to consider interactions between physical systems. Invariably, as the limits of knowledge that can be extracted from a particular model are reached there is a desire to incorporate more physics, scales, dimensions, or couplings as the case may be. In practice, these extensions are often aided or hindered by the specifications or lack thereof made initially in writing computational software.

An incomplete list of computational capabilities for a plasma physics application capable of handling a member of the "forest" mentioned previously might look like this: single and multiple domain meshing, discretizations, linear algebra, linear and nonlinear solvers, time integrators, multi-physics couplings, multi-model couplings, solution transfer, uncertainty quantification, and data assimilation in addition to computer science specific questions such as communication libraries, parallel load balancing, and portability across emerging new computing architectures.

A simple start to heeding Dr. Lampert's call to handle the laundry list above might be to consider the humble map $L : U \to V$ where L could be a linear or nonlinear map with U and V being domain and range spaces. L only requires a vector $u \in U$ as its input and produces $v \in V$ as its output with $v = L(u)$, or more generally $L(u, v) = 0$. L could for example represent the forward action of a model expressed as a system of partial or ordinary differential equations (PDE's or ODE's) or an inverse map representing the action of a linear or nonlinear solver. There is a richness in this simple construct that is capable of handling much complexity. L, U, V, and their components (such as vectors) are mathematical objects for which direct representations can be constructed in software, most easily in object-oriented languages including Fortran2003 and C++. In the discussion below mathematical objects that can be directly realized in software are described.

Multi-physics models: Consider two operators $L_1(u_1; u_2)$ and $L_2(u_2; u_1)$ representing two different physics models (say the magnetic and fluid components of MHD) where $L_1 : U_1 \to V_1$ and $L_2 : U_2 \to V_2$ where u_1 is in the domain space of L_1 and a parameter for L_2 with a similar interpretation for u_2. A multi-physics model coupling both operators can be constructed by considering the operator $L = (L_1, L_2)^t : U_1 \times U_2 \to V_1 \times V_2$. If L_1, L_2, and L all obey the minimal interface

described, coupled physics simulations require no code changes on the solver or operator side to explore fully implicit or operator split approaches while continuing to maintain useful single physics calculations. However, this does require the construction of the new objects L, $U_1 \times U_2$, and $V_1 \times V_2$. The natural extension to multi-physics systems $L = (L_1, L_2, \cdots, L_n)^t : U_1 \times \cdots U_n \rightarrow V_1 \times V_2 \cdots \times V_n$ for $n > 2$ follows. For large multi-physics systems with complex strong and loose coupling between the various components where it is not often evident in advance what the best solution approaches are this minimal operator approach provides an extensible and incremental approach allowing for both research and production codes to co-exist in the same framework.

Multi-domain models: can again be constructed by considering operators L_1 and L_2 (in the case of 2 domains) with the underlying spaces on which the operators are constructed spanning different domains. L_1 and L_2 may or may not represent the same physics and may be coupled through an interface operator L_i. An example would be a fluid structure interaction problem where a plasma (plasma physics L_1) heats a material wall (thermal physics L_2) with surface reaction physics (L_i). L_i may also simply represent a projection operator that transfers solution data between the two domains. As an aside we note that formulating coupling in this fashion simplifies coupling surface and volume physics calculations. Furthermore, L_1, L_2, and L_i can potentially be multi-physics or multi-domain operators themselves thereby opening the door to multi-domain, multi-physics simulations and the exploration of a wide variety of coupling and solution techniques.

Multi-model simulations: Some problems in plasma physics require both particle and continuum model descriptions at different scales. Encapsulating the particle and continuum models within operator formulations with projection operators providing coupling is a natural way to develop independent codes that are yet capable of being coupled. Note that such an approach is also capable of providing a systematic approach to numerical homogenization and adaptive resolution across a range of scales.

Reduced models: Consider the compositional operator $L = L_1 \circ L_2 \circ L_3 : Z \rightarrow U$, where $L_3 : Z \rightarrow W$, $L_2 : W \rightarrow V$, and $L_1 : V \rightarrow U$. L could for example represent the multi-domain plasma heating example above. An operator based software approach would allow the introduction of reduced models $\tilde{L} = L_1 \circ (P \circ \tilde{L}_2 \circ R) \circ L_3 : Z \rightarrow U$ where $R : W \rightarrow \tilde{W}$ is a restriction operator to a reduced space, $\tilde{L}_2 : \tilde{W} \rightarrow \tilde{V}$ is the reduced model, and $P : \tilde{V} \rightarrow V$ is a lifting operator with minimal code disruption.

The above list hints merely at a small subset of potential opportunities with a lack of space preventing the description of what is possible when the domain and range space designs are also considered. It is easy to misconstrue that the above description is too abstract, general and impractical. Incomplete realizations of the above design in software have already enabled massively parallel multi-physics simulations across hundreds of domains and hundreds of individual solver and operator components. Returning to the list of computational capabilities we note that an operator based approach allows questions of single and multiple domain meshing, discretizations, and linear algebra to be handled within the context of constructing appropriate discrete spaces. This opens the door to coupled simulations where the individual components could be based on different meshing and discretization components. Multi-physics and multi-domain models, linear and nonlinear solvers, and time integrator components all can be handled elegantly through an operator based design and implementation. Without belaboring the point further we note that the rich underlying and overlaying structures of fields, topological, vector, and inner product spaces and operator spaces among others represent a refined, rigorous, and verifiable specification and design that if translated directly into computational software would lead to extensible software applications.

[1] http://www.technologyreview.com/news/525621/three-questions-for-leslie-lamport-winner-of-computings-top-prize