

The design and development of modular and adaptive algorithms

J. W. Banks, W. D. Henshaw, LLNL

How do we construct physics based simulation tools¹ that can be made to efficiently use a future computer whose makeup is not known exactly, but whose general construction can be inferred from prevailing hardware trends? These trends inform our best estimates of the likely composition of such a next generation machine, and the outlook is that this computer will be massively multi-core (including embedded accelerators such as GPUs), will penalize excessive communication, and may experience performance irregularities and frequent faults. Our position is therefore:

Proposition: *To operate effectively in this challenging environment, mathematical research is required to construct new modular algorithms that can be decomposed into a large number of loosely coupled components, themselves consisting of multiple subcomponents or tasks, whose execution can be automatically and adaptively orchestrated in order to further the overall simulation.*

Significant new math research on developing decoupled modular strategies should be a priority and will likely lead to much better utilization of future machines. To accomplish this goal the following items should be addressed;

- increased mathematical modularity of the simulation tool,
- flexible, dynamic methods that adapt to the environment, and
- a high-level algorithmic description that enables automatic generation of low-level implementations and dynamic algorithm selection.

Increasing the modularity simulation tools through mathematical decomposition is an important applied math problem that should be addressed. Consider the example of a fluid-structure problem consisting of compressible fluid flow with an elastic body. Broadly speaking there are two primary simulation approaches for such an FSI problem: partitioned and monolithic. In a partitioned approach, the solvers for each piece are isolated from each other and coupled only through the interface. This is in contrast to a fully-coupled, monolithic approach where the entire system is advanced by a single unified solver, typically by an implicit method. In a partitioned solver, each physical process is considered in isolation, so each solver constitutes a confined module. Partitioned techniques naturally increase algorithmic concurrency, reduce global synchronizations, and fit well within a modular fault tolerance strategy where global resilience is achieved by hardening isolated subcomponents and then combining these in a hierarchical manner. There are many examples of modular partitioning including approximate factored schemes, alternating-direction-implicit (ADI) methods, deferred correction, etc. Whatever the case, by introducing modularity, complex simulations are reduced to smaller subcomponents whose interconnectedness is more easily manipulated to take full advantage of a new generation of computer (for example overlapping communication with computation, restarting failed computations due to hardware interruptions, or incorporating local time stepping).

¹our focus is primarily partial differential equations (PDEs)

Partitioned schemes have some significant challenges including maintaining numerical stability, achieving good convergence rates, and maintaining accuracy at interfaces, all of which can be addressed by applied math research. For example, we have recently demonstrated that by using detailed mathematical and numerical analysis, stable and accurate partitioned solvers can be created for compressible FSI problems [1, 2, 3, 4]. Our work is now pushing these results into incompressible FSI regimes and more. Other applications are likely to see similar benefit from a renewed push toward increased modularization.

We can create flexible, dynamic methods that adapt to the environment by taking advantage of increased mathematical modularity. It seems likely that future machines will be highly heterogenous in the sense that different cores and/or pieces of the communication network may operate slightly differently than their counterparts on different parts of the machine. As a result, different algorithms and/or different implementations of the same algorithm are likely to perform differently on various pieces of the machine. In order to obtain optimal performance it will be important to dynamically select the best fit for each part of the machine. Flexibility is key here, and mathematical analysis should be performed to decide which combinations can be permitted and which cannot. For example, it may be that some part of the machine is unresponsive for some time, and the remainder of the computation needs data from the unresponsive piece to continue. Rather than hold up the whole calculation, we might use a low cost model (such as extrapolation) for some time and then sync back up at some later time. The stability implications of such an approach are clearly significant, and detailed numerical analysis is needed to understand the limits of the overall approach.

Developing a high-level algorithmic description that enables automatic generation of low-level implementations and dynamic algorithm selection will provide computer scientists and mathematicians a common framework that can be used to develop scalable and portable simulation tools. Higher level descriptions of a problem retain important information about the nature of the computation that can be used when mapping the simulation onto a machine. In fact the same problem may need to be instantiated differently on different subsets of the computer in order to obtain optimal performance. For example it is extremely unlikely that the intent of an application is to solve the matrix equation $Ax = b$. To be sure, this will be a critical step in many simulation tools, but the set of solution strategies that may be appropriate depends significantly on the nature of the continuous problem. For example, elliptic problems are likely to be able to take advantage of multigrid, while hyperbolic problems may have trouble when treated with the same techniques. As a second example, it may be more efficient to interleave multiple ensemble calculations for a UQ analysis than to run them independently; for example, a lack of perfect scalability in one ensemble calculation may lead to idle processors that could be usefully employed for a second ensemble calculation. These choices will need to be made automatically and dynamically. We will therefore need to devise a high-level language to express the basic mathematics of the problem being solved as well as an outline of the discretization being used. This information can then be made available to the runtime environment and the whole system can be dynamically optimized in a holistic manner.

References

- [1] J. W. Banks, B. Sjögreen, A normal mode stability analysis of numerical interface conditions for fluid/structure interaction, *Commun. Comput. Phys.* 10 (2) (2011) 279–304.
- [2] J. W. Banks, W. D. Henshaw, D. W. Schwendeman, Deforming composite grids for solving fluid structure problems, *J. Comput. Phys.* 231 (2012) 3518–3547.
- [3] B. Sjögreen, J. W. Banks, Stability of finite difference discretizations of multi-physics interface conditions, *Commun. Comput. Phys.* 13 (2) (2013) 386–410.
- [4] J. W. Banks, W. D. Henshaw, B. Sjögreen, A stable FSI algorithm for light rigid bodies in compressible flow, *J. Comput. Phys.*