THE ADAPTIVE MULTISCALE SIMULATION INFRASTRUCTURE - METHODS AND SOFT-TISSUE EXAMPLE SIMULATION

WILLIAM R. TOBIN *, VICTOR W.L. CHAN*, CATALIN R. PICU*†, VICTOR H. BAROCAS‡, AND MARK S. SHEPHARD*

Abstract. [Discuss multi-scale and frameworks, show how AMSI does multi-scale, and how it is different than others. Introduce the soft-tissue problem. Discuss hierarchical multi-scale and relate it to well-known multi-scale models, and what requirements it imposes. Discuss AMSI design decisions, the implementation of AMSI, and how to use AMSI to implement the soft tissue simulation. Show some performance measures relevant to the implementation of this sort of simulation, and finally produce a roadmap on how to apply AMSI to other simulations (hierarchical and concurrent).]

1. Introduction.

1.1. Multi-scale Simulation. The development of new numerical simulations has proliferated rapidly in the last several decades. The massive increases in computational power of massively parallel machines has facilitated the implementation and execution of numerical simulations of unprecedented scale and fidelity. However, when it is the case that the governing physical mechanism of interest in a simulation occurs at a scale orders removed from the desired resolution of the solution, even the massive increases in computational power of the last decades is insufficient in many cases. Multi-scale simulation techniques offer a method of achieving accurate results on a domain orders of magnitude removed from the underlying phenomena guiding the physical action of the simulation.

Multi-scale simulation broadly falls into two categories: concurrent (partitioned-domain) and hierarchical problems. These categories are most readily distinguished by the relationships between the interacting scales in a problem.

In concurrent problems the domain of each scale is modeled using similar physical dimensions: the standard units used to describe the scale will typically fall within several orders of magnitude of each other. Interface surfaces or regions between the scale domains are used to transfer relevant physical tensor field information between scales. The transfer of relevant tensor field values allow the scales to influence one another and gives rise to multi-scale behaviors in the problem simulation. Often values in concurrent simulations will not require the use of up- or down-scaling operations to transfer tensor values.

The Domain Decomposition Methods (DDM) [21] and the Quasicontinuum Methods (QM) [17] both allow the implementation of concurrent multi-scale problems. DDM allows the implementation of multi-physics continuum-to-continuum coupling while QM is used to couple atomistic models to continuum regions in a macro-scale domain.

Hierarchical problems are characterized by strongly-separated scales in which the physical dimensions of the scale are typically 6 or more orders of magnitude separated. The domain relationship between two such scales is often reduced to the existence of micro-scale simulations occurring at various points throughout the larger-scale simu-

*Scientific Computational Research Center, Rensselaer Polytechnic Institute, Low Center for Industrial Innovation, CII-4011, 110 8th Street, Troy, NY 12180.

†Department of Mechanical, Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute, Jonsson Engineering Center, Rm. 2049, 110 8th Street, Troy, NY 12180.

‡Department of Biomedical Engineering, University of Minnesota, 7-105 Nils Hasselmo Hall, 312 Church Street SE, Minneapolis, MN 55455.

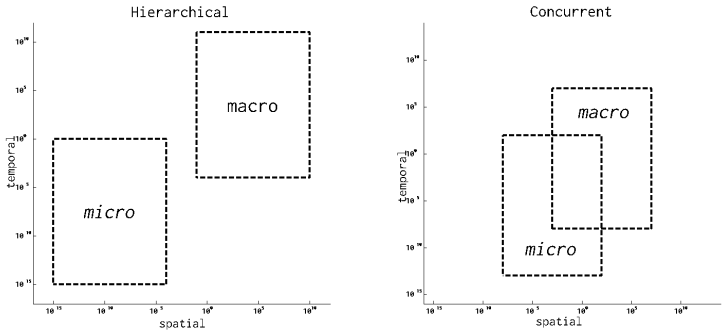

FIG. 1. *Scale separation maps for hierarchical and concurrent multi-scale problems, respectively.*

lation. The tensor field couplings which give rise to multi-scale behavior occur at each point in the macro-scale simulation at which a micro-scale simulation is embedded. The local value of tensor field data is used to establish conditions at the micro-scale, which in turn produces a tensor field value relevant to the evolution of the macro-scale simulation. Values in hierarchical simulations are more likely to require up- and down-scaling operations to account for differences between the coupled scales.

Hierarchical problems can be implemented with the use of information passing schemes in which a well-defined set of parameters is transmitted between the simulation scales [22, 8, 10]. This approach allows the replacement of a constitutive model commonly used in continuum mechanics problems with a full single-scale simulation conducted at a distinct points in the macro-scale problem.

The Heterogeneous Multiscale Methods (HMM) [23, 24] are methods used to solve multi-scale problems with either localized regions of macroscopic model failure or problems without a known macroscopic model entirely. HMM is well-suited for application to hierarchical problems as it does not require the existence of a macro-scale constitutive model, allowing the micro-scale simulations to fully guide system behavior.

An example of scale separation maps [5] associated with these multi-scale problem categories can be seen in figure 1. Further discussion of the taxonomy of multi-scale modeling in [20] is a useful primer on the various categories of multi-scale models, with accompanying examples for each category to further elucidate the concepts.

In the rest of this paper we will discuss currently existing multi-scale frameworks including particulars of their implementations and use cases. We will then discuss the Adaptive Multi-scale Simulation Infrastructure (AMSI) tools we have developed including our intent in development and targeted problem types, the design of our system, and the subsequent implementation of that design. We will further discuss the application of our infrastructure to the development of a multi-scale soft biological tissue simulation. We then discuss the performance of the soft tissue simulation and how this relates to the AMSI multi-scale coupling approach. Finally we will discuss our plans for future development and extension of the capabilities of our infrastructure.

2. Multi-scale Frameworks. There are numerous multi-scale frameworks currently in development intended to facilitate the implementation of multi-scale simulations on HPC machines. We do not intend to provide a full survey of such frameworks in this section but rather to discuss several frameworks which exhibit features common to many frameworks existing in the multi-scale space at this moment.

Software intended to support multi-scale simulations needs to enable the introduction of new models, algorithms, and data structures [16]. Further, a set of well-defined modules with distinct interfaces must be implemented, especially as this concerns data ownership and simulation state. State data must be query-able through interfaces in order for coupled codes to maintain consistency throughout the simulation. Further, implementing general code, or providing mechanisms to extract important underlying data in general formats is necessary. This forces multi-scale application developers to adhere to specific data structures and algorithms, or write their own wrapper code in order to translate specialized data structures into more general formats. These issues and rationales for multi-scale software development are further discussed in[16].

The Uintah Problem Solving Environment (PSE) [4] is a framework developed at the University of Utah intended to allow the intuitive combination of parallel simulation components into novel scientific applications, including but not limited to multi-scale simulations. Each component in the Uintah system is designed to solve a PDE on a structured AMR grid, and must adhere to a C++ API used to establish connections between components. All components are decomposed into a set of discrete tasks, each of which has a well-defined set of required input and output data. The information about these tasks is expressed in the form of a directed acyclic graph (DAG) [3], and is used by the Uintah system to schedule task execution across the HPC system. The parallel communication necessary to provide task input and output occurs through a data warehousing structure which controls access to local and global data.

The Mesh-Based Parallel Code Coupling Interface (MpCCI) [15] is a library which facilitates the coupling of simulation codes based on the underlying domain grid over which simulation data (specifically tensor field values) are distributed. Support for different grid formats is provided by MpCCI, as well as different methods of data interpolation when needed to communicate data between simulation scales with mismatched grids. MpCCI uses a code-coupler component object in order for two single-scale application codes to communicate with eachother [9]. A coupler object is implemented to transform values from a producing single-scale code into values that can be used by a consuming single-scale code.

In the time since MpCCI was developed over a decade ago it has seen sufficent adoption to warrant the organization of several user forums and is currently used to support the implementation of multi-physics codes. There have also been coupler objects implemented to allow many industry-standard simulation tools to be coupled through MpCCI including Abaqus, ANSYS Fluent, MATLAB, and others [1].

The `MUSCLE-HPC` [2] code is a partial implementation of the `MUSCLE-2` toolkit [6] for use on HPC machines. The multi-scale simulation in `MUSCLE-HPC` is constructed from a set of submodels, each of which is implemented by inheriting from an abstract C++ class which provides an API for initialization, boundary condition updates, numerical solution iteration, and several data query functions. Coupling together of submodels into a multi-scale model is done through the use of an interface of Conduits which the class API has methods to express the set of input and output Conduits for the submodel. Further the `MUSCLE-HPC` runtime environment requires that a machine core acting as a server is devoted to running a special Manager class to register client submodels and facilitate the creation of parallel communication channels between the submodels.

3. AMSI. The Adaptive Multi-scale Simulation Infrastructure (AMSI) a set of libraries providing support for the implementation and execution of dynamic numer-

ically multi-scale simulations on massively-parallel HPC machines. AMSI is designed to ease the incorporation of legacy single-scale simulation codes into new multi-scale simulations. This is accomplished by minimizing the insertion of calls to the AMSI systems in a legacy simulation code to only those locations associated with the desired multi-scale coupling.

In contrast to the approach taken in MpCCI, the single-scale codes used in AMSI need not possess a spatial discretization of the problem domain. Nor is there any requirement that AMSI directly query the domain data structures. Thus AMSI does not require that scale-coupling data be associated with any spatial discretization as is required in [15]. This protects the capabilities provided by AMSI against a shift in community adoption of new simulation models or different domain representations.

[contrast uintah, muscle]

An AMSI user can integrate their codes using AMSI without the need to refactor their data structures or interfaces in order to accommodate the AMSI runtime, as is the case when using multi-scale frameworks designed to operate on a well-specified API the user must expose to the framework. AMSI requires only minimal placement of multi-scale coupling calls at those locations in legacy codes associated with the coupling operations used for multi-scale interaction. All interfaces to the AMSI system operate on built-in data types or are implemented to accept generic data types, limiting any need to refactor existing codebases to contend with specialized data types and inheritance schemes [18].

AMSI is not tied to any particular physics model, domain, or fields implementation or representation, providing operations and data structures to model coupling on an abstract level, while facilitating and supplying less abstract operations (or hooks for such operations) where possible.

There are many open questions in the development and verification of multi-scale modeling approaches [12]. AMSI attempts to provide the tools to approach these questions through the development of new multi-scale models and simulations from existing single-scale simulations without placing assumptions on the multi-scale model or simulation implementation that would preclude exploration of an open question.

4. Model Design. The design of the simulation coupling operations utilizing the AMSI multiscale library is informed by two abstract models. The first is the Abstract Scale Model (ASM), which describes a generalized single-scale simulation. The second is the Abstract Coupling Model (ACM), which describes a generic coupling of two such simulations. These models provide a workflow through which a simulation developer can produce a new multi-scale simulation on an HPC machine using the facilities provided by the AMSI libraries. Further, these models guided many development decisions during the implementation of the AMSI multiscale library.

[expand:] By instantiating these models for a particular problem, a developer will consider all selection decisions required by the proposed multi-scale taxonomy discussed in [11].

The AMSI model workflow additionally includes considerations for the hierarchy of implementation operations that go into the production of a new multi-scale code.

The design of a Multiscale Modeling Language (MML) is discussed in [13]. Our approach is compatible with such a language though we do not directly necessitate the use of any specialized languages in the construction of multi-scale simulations with AMSI.

[work in citation [14]]

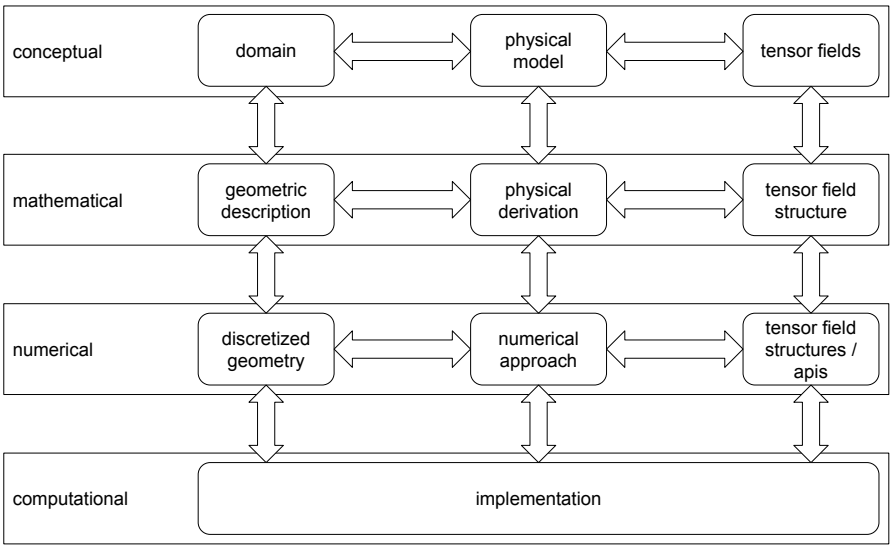

FIG. 2. *The abstract single-scale model used to inform the development of a multi-scale coupling model.*

4.1. Abstract Scale Model. The Abstract Scale Model (ASM) (illustrated in figure 2) provides a hierarchical model of the implementation specifics of a single-scale simulation. The model is oriented on the principle of most abstraction to least abstraction along the vertical axis. Each level in the abstraction hierarchy requires information in three different areas: the physics being simulated, the domain (type) over which the simulation is to take place, and the physical tensor fields required to describe the problem. By specifying or implementing the information required at each node in the ASM a developer will fully describe and produce a single-scale simulation.

The conceptual physical model is often the starting point in the development of an ASM, as it informs many of the decisions made during the specification of the rest of the model. This high-level description is often specified in terms of the primary physical property(ies) being modeled, e.g. displacements, temperature, trajectories, etc, as well as (typically) one or more conservative laws. The specification of the physical model most directly effects the conceptual domain and conceptual tensor fields. The conceptual domain incorporates both the temporal and spatial domains of the problem, which provides sufficient information to plot the scale on a scale-separation graph. [Additionally the conceptual domain includes the specification of e.g. continuum vs. atomistic approaches.] The conceptual tensor fields are directly influenced by the physical model and includes every primary and supporting field required to simulate the physical model as specified by the chosen conservative laws.

[add in some references:] The mathematical physical derivation is a specification of the chosen form of the conservative law(s) from the conceptual physical model and accompanying mathematical requirements. The mathematical geometric description of the domain incorporates the specification of the method of mathematical representation of a domain for the problem, e.g. for a continuum problem a CAD format or grid assumptions. The mathematical tensor field structure for the required fields is specified, including symmetries which allow for tensor order reductions to simplify the mathematical formulations in the physical derivation.

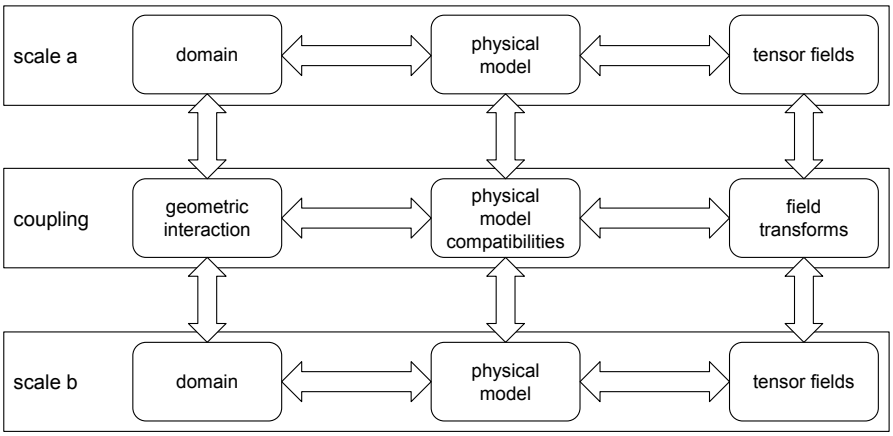

FIG. 3. *The abstract multi-scale coupling model used to guide the design of AMSI.*

The numerical approach includes the approach to solving the mathematical physical derivation this includes the construction and solution of the discretized model. The discretized geometry specifies the representation of the geometric domain the numerical approach will query and operate on, this includes for instance meshes and grids for continuum problems and discrete particles for MD problems. The tensor field structures and APIs are specified over the discretized geometry and are queried for use in the numerical approach to produce a solution which is also expressed in terms of a tensor field structure.

The computational implementation of an ASM requires the combination of all numerical specifications in order to produce a final simulation.

4.2. Abstract Coupling Model. The ACM is derived from a model initially discussed in [19].

The Abstract Coupling Model (ACM) (seen in figure 3) provides a model of the coupling of two distinct realizations of an ASM (fig 2). The hierarchy of each coupled ASM must be taken into account during the specification of the primary nodes in the ACM. It is convenient to view the ACM as a top-down view of two ASMs and the coupling layer between them. Each of the conceptual, mathematical, and numerical layers must still be considered when specifying nodes in the ACM. Specifying and implementing the ACM fully will result in a valid multi-scale coupling operating between two single-scale simulations.

The physical model compatibility requires the specification of the relationship between the two physical models. This includes any laws relating the physical quantities modeled in the two ASMs, and may require intermediate operations including derived values in order for the two physical models to influence each other. Proceeding down the ASM hierarchy this further requires a particular mathematical derivation, as well as the specification of any numerical operations required to produce all quantities needed to satisfy the mathematical derivation.

The geometric interaction of problem domains includes the specification of each domain on a scale-separation map, as well as the nature of the domain relationship with respect to e.g. overlapping regions/interfaces or full locality of sub-domains for strongly-separated scales. This specifies whether the multi-scale problem is concurrent or hierarchic as discussed in sections 1.1. Going down the hierarchy this also includes

any mathematical operations required to map spatially between the domains which can include e.g. inverting mappings, interpolation operations, tensor field queries at selected locations. At the numerical level the operations necessitated by the mathematical relationships must be implemented.

The field transforms in a coupling require the specification of all operations required for influences from a field associated with a scale to be incorporated into a field on the other scale(s) in the ACM. At the conceptual level this includes the full set of tensor fields on each scale along with known relevant relationships between the fields. The specification of these operations is primarily mathematical and numerical in nature, and is influence heavily by the physical model compatibility and geometric interactions.

5. AMSI Model Implementation.

6. Parallel Design.

6.1. AMSI Design. The AMSI multiscale library provides the facilities to instantiate, manage, and execute dynamic scale coupling operations on otherwise uncoupled single-scale simulation codes. AMSI specifically targets the use-case of a developer taking existing single-scale simulation codes and combining them to produce a new multi-scale code while minimizing intervention in the legacy codebases.

The core of the AMSI multiscale systems operates at the numerical and implementation levels of the ASM and ACM abstraction hierarchy. While developers use a top-down approach to specify and construct a simulation, initial development on AMSI multiscale has focused on providing functionality in a bottom-up fashion. The intent of this approach is to provide a useful set of functionality for developers before working up the abstraction hierarchy to provide facilities for operations required at those levels.

At the lowest-level, the AMSI multiscale library is concerned with the instantiation and management of the individual single-scale codes used in a multi-scale simulation. The parallel structure of HPC machines requires that in order to couple multiple single-scale codes into a multi-scale code, the relationship of those individual single-scale codes to the machine must be modeled by AMSI multiscale.

The set of threads of execution in a given simulation and their relationship with the physical hardware of the machine is called the parallel execution space (roughly equivalent to MPI_COMM_WORLD) of the simulation. AMSI multiscale models the parallel execution space of the whole multi-scale simulation as well as each single-scale code in the multi-scale system, through the use of a Scale.

6.2. AMSI Scales. The Scale is a data structure modeling the relationship between the computational level of a single ASM used in the construction of a multi-scale simulation and the parallel execution space of the HPC machine the simulation is executing on. The relationship between the Scale and the parallel execution space is specified by the user using an AMSI multiscale configuration file, further discussed in section 7. All Scales present in a multi-scale simulation are instantiated and configured at the beginning of a multi-scale simulation.

Several methods for determining the partitioning of the parallel execution space and assigning the partitions to Scales in a simulation are provided. Scale allocations may be determined by total process count, process ratios between multiple Scales, and additionaly may take advantage of machine introspection operations such as using the `hwloc` [7] hardware locality library in order to take advantage of machine structure for parallel communication operations.

Also during initialization, a main-like function is associated with an individual Scale. A developer may simply reuse the main function of the existing single-scale simulation, however the function is additionally passed an MPI_COMM associated with the Scale on which parallel operations for the single-scale code modeled by the Scale are to take place.

6.3. AMSI Couplings. The Coupling is a data structure modeling the relationship of units of coupling data between two distinct Scales. The Coupling in AMSI multiscale is not assumed to be symmetric so in order to establish a two-way coupling between Scales a Coupling must be produced for each direction of communication. While it is often the case that a coupling is bi-directional, this is not always the case [11], so this form is preferred. Each Coupling exists on the union of the parallel execution spaces associated with both Scales being coupled, and specifies on Scale as the producer and the other as the consumer.

The Coupling operates at the computational level of the ACM, but is directly informed by specifications at the numerical level. Individual units of scale-coupling data are supplied to the Coupling, e.g. tensor field values located at points as specified by the domain relationship in the ACM. These may be raw values or values derived locally, i.e. the tensor field transformation operations may by applied to the coupling data on either end of the coupling. This allows for minimization of communication overhead whenever possible.

A unit of coupling data generically refers to any data type as the precise implementation or data structure of the data is not assumed by the Coupling or AMSI multiscale until the communication phase of the coupling operation.

The parallel discretization of the numerical domains of the multi-scale problem informs the parallel locality of both the origin and the destination of each unit of scale-coupling data. At present this mapping is taken care of by the developer. Facilitating the automation of the construction of this mapping is the next step in advancing the development of AMSI multiscale up the ASM/ACM abstraction hierarchy.

7. AMSI Parallel Implementation. [cite the in-progress neuron paper as the most up-to-date reference for biotissue]

Software intended to suppport multi-physics simulations needs to "enable the introduction of new models, algorithms, and data structures" [16]. Further, a set of well-defined modules with distinct interfaces must be implemented, especially as this concerns data ownership and simulation state. State data must be query-able through interfaces in order for coupled codes to maintain consistency throughout the simulation. Further, implementing general code, or providing mechanisms to extract important underlying data in general formats is necessary. This forces multi-physics application developers to adhere to specific data structures and algorithms, or write their own wrapper code in order to translate specialized data structures into more general formats. These issues and rationales for multi-scale software development are further discussed in [16].

Toward this end AMSI provides a simple API used for multi-scale coupling and scale-sensitive load-balancing operations, and makes no requirement on the application code providing any interfaces for AMSI usage. This allows intervention in legacy code only in locations where multi-scale operations are to be implemented, and where replacement of global collective operations with scale-collective operations is required in order to prevent parallel deadlock.

[Show a multi-scale simulation initialization in AMSI and how construct and execute a Coupling using code snippets.]

8. Soft-Tissue Implementation. Show how AMSI was used with the existing single-scale simulations to construct the biotissue problem. Discuss the tensor fields involved in the coupling and the transformations required to coupled the physical models together (deformation gradient and stress/stress derivatives), the geometric coupling which is known a-priori due to the strongly-separated nature of this multi-scale simulation since it is hierarchical.

Discuss the control flow of the single-scale macro and micro simulations. Discuss the modifications to the control flow using AMSI to couple the simulations together into a multi-scale simulation.

Include control flow diagrams for both individual scales as well as the combined multi-scale simulation.

9. AMSI Performance Measures. Discuss that currently scales must operate on non-overlapping partitions in the parallel execution space, requiring that one scale is idle while the other executes. Address how to minimize this idle time.

What constitutes a good set of results? It likely mostly depends on how we describe our difference from the existing frameworks and what our key goals in AMSI are

- We mostly focus on the ease of implementing a multi-scale simulation from existing simulations - but this is hard to develop a metric for

- We also want computational efficiency, and while we can calculate our overall compute efficiency for any run we don't really have anything to compare against

10. Development of Additional Multi-scale Simulations.

10.1. Hierarchical Simulations. This should be relatively straightforward, just need problem to discuss.

The production of CouplingData objects for hierarchical multi-scale problems is particularly straightforward as the domain relationship is well-established a-priori.

10.2. Coupled-Domain Simulations. Discussion of obstacles being worked on towards concurrent multi-scale with discussion of at least one particular problem.

The production of a CouplingData object derived from the relationship between the domains in a concurrent partitioned-domain problem is a complex operation currently undergoing development.

REFERENCES

[1] *Mpcci web page*, 2016, http://www.mpcci.de/.

[2] M. B. Belgacem and B. Chopard, *Muscle-hpc: A new high performance api to couple multiscale parallel applications*, Future Generation Computer Systems, 67 (2017), pp. 72–82.

[3] M. Berzins, J. Beckvermit, T. Harman, A. Bezdjian, A. HUMPHREY, Q. MENG, J. SCHMIDT, and C. WIGHT, *Extending the uintah framework through the petascale modeling of detonation in arrays of high explosive devices*, Submitted to SIAM Journal on Scientific Computing, (2015).

[4] M. Berzins, J. Luitjens, Q. Meng, T. Harman, C. A. Wight, and J. R. Peterson, *Uintah: a scalable framework for hazard analysis*, in Proceedings of the 2010 TeraGrid Conference, 2010, p. 3.

[5] J. Borgdorff, J.-L. Falcone, E. Lorenz, C. Bona-Casas, B. Chopard, and A. G. Hoekstra, *Foundations of distributed multiscale computing: Formalization, specification, and analysis*, Journal of Parallel and Distributed Computing, 73 (2013), pp. 465–483.

[6] J. Borgdorff, M. Mamonski, B. Bosak, K. Kurowski, M. B. Belgacem, B. Chopard, D. Groen, P. V. Coveney, and A. G. Hoekstra, *Distributed multiscale computing with muscle 2, the multiscale coupling library and environment*, Journal of Computational Science, 5 (2014), pp. 719–731.

[7] F. Broquedis, J. Clet-Ortega, S. Moreaud, N. Furmento, B. Goglin, G. Mercier, S. Thibault, and R. Namyst, *Hwloc: A generic framework for managing hardware affinities in hpc applications*, in Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, PDP '10, Washington, DC, USA, 2010, IEEE Computer Society, pp. 180–186, https://doi.org/10.1109/PDP.2010.67, http://dx.doi.org/10.1109/PDP.2010.67.

[8] M. J. Buehler, *Concurrent scale coupling techniques: From nano to macro*, Lecture Series, 3 (2006).

[9] F. Delalondre, C. Smith, and M. S. Shephard, *Collaborative software infrastructure for adaptive multiple model simulation*, Computer Methods in Applied Mechanics and Engineering, 199 (2010), pp. 1352–1370.

[10] J. Fish, *Multiscale methods: bridging the scales in science and engineering*, Oxford University Press on Demand, 2010.

[11] V. Gravemeier, S. Lenz, and W. A. Wall, *Towards a taxonomy for multiscale methods in computational mechanics: building blocks of existing methods*, Computational Mechanics, 41 (2008), pp. 279–291.

[12] A. Hoekstra, B. Chopard, and P. Coveney, *Multiscale modelling and simulation: a position paper*, Phil. Trans. R. Soc. A, 372 (2014), p. 20130377.

[13] A. G. Hoekstra, E. Lorenz, J.-L. Falcone, and B. Chopard, *Toward a complex automata formalism for multiscale modeling*, International Journal for Multiscale Computational Engineering, 5 (2007).

[14] G. Ingram, I. Cameron, and K. Hangos, *Classification and analysis of integrating frameworks in multiscale modelling*, Chemical engineering science, 59 (2004), pp. 2171–2187.

[15] W. Joppich and M. Kürschner, *Mpccia tool for the simulation of coupled applications*, Concurrency and computation: Practice and Experience, 18 (2006), pp. 183–192.

[16] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, et al., *Multiphysics simulations challenges and opportunities*, International Journal of High Performance Computing Applications, 27 (2013), pp. 4–83.

[17] R. E. Miller and E. B. Tadmor, *The quasicontinuum method: Overview, applications and current directions*, Journal of Computer-Aided Materials Design, 9 (2002), pp. 203–239.

[18] S. J. Plimpton and J. D. Gale, *Developing community codes for materials modeling*, Current Opinion in Solid State and Materials Science, 17 (2013), pp. 271–276.

[19] M. Shephard, M. Nuggehally, B. FranzDale, C. Picu, J. Fish, O. Klaas, and M. Beall, *Component software for multiscale simulation*, Bridging the Scales in Science and Engineering, (2010), pp. 393–421.

[20] E. B. Tadmor and R. E. Miller, *Modeling materials: continuum, atomistic and multiscale techniques*, Cambridge University Press, 2011.

[21] A. Toselli and O. B. Widlund, *Domain decomposition methods: algorithms and theory*, vol. 34, Springer, 2005.

[22] E. Weinan, *Principles of multiscale modeling*, Cambridge University Press, 2011.

[23] E. Weinan, B. Engquist, et al., *The heterognous multiscale methods*, Communications in Mathematical Sciences, 1 (2003), pp. 87–132.

[24] E. Weinan, B. Engquist, X. Li, W. Ren, and E. Vanden-Eijnden, *Heterogeneous multiscale methods: a review*, Commun. Comput. Phys, 2 (2007), pp. 367–450.