A Model-to-Model Interface for Concurrent Multiscale Simulations

P. E. Antonelli,^{1,2} K. M. Bryden,^{1,3} and R. LeSar^{1,2‡}

¹Ames Laboratory, Ames, IA 50011-3020

²Department of Materials Science and Engineering, Iowa State University, Ames, IA 50011-2300

³Department of Mechanical Engineering, Iowa State University, Ames, IA 50011-2274

Abstract

We present a low-level model-to-model interface that will enable independent models to be linked into an integrated system of models. The interface is based on a standard set of functions that contain appropriate export and import schemas that enable models to be linked with no changes to the models themselves. These ideas are presented in the context of a specific multiscale materials problem that couples atomistic-based molecular dynamics calculations to continuum calculations of fluid flow to examine the influence of interactions of the fluid with an adjacent solid on the fluid flow.

Keywords:

[‡] Corresponding author. lesar@iastate.edu

1. Introduction

Multiscale modeling is ubiquitous in science and engineering, arising in many and quite divergent domains, from the design of materials to simulations of power plants. All of these systems are governed by important features that control their behavior at multiple scales of length and time. Describing such systems often depends on having independent models based on differing physics for each scale.[1,2]

Integrated modeling,[3,4] which can be defined as modeling that “includes a set of interdependent science-based components (models, data, and assessment methods) that together form the basis for constructing an appropriate modeling system”,[3] provides a structure to develop and organize the relevant information about a multiscale system and to apply that information to model the behavior of that system in response to external forces. The challenges of creating such integrated modeling systems are common to many disciplines. The environmental modeling community, for example, has been very active in this area, driven by their need to coordinate between various disciplines and to integrate a wide range of models that are often developed autonomously and independently from the integrated modeling efforts, and generally by experts who may not be tightly connected to the overall effort.[3,4] Inclusion of new models and codes into an integrated modeling system thus often requires coordination between disparate groups, which can inhibit addition of new capabilities into the integrated scheme. As discussed in more detail below, we have similar challenges in multiscale materials modeling.

In this paper we present an approach to integrated modeling based on the development of a model-to-model interface that enables independent models to be more readily linked into an integrated system with no changes to the models themselves. We examined a number of existing technologies for linking models, all from environmental modeling. While these systems have not been applied to systems with the range of scales found in materials, their basic structures provide an excellent starting point for creating integrated multiscale materials models. The models we examined include the Bespoke Framework Generator,[5] Earth System Modeling Framework - Flexible Modeling System,[6] OASIS (Ocean Atmosphere Sea Ice Soil),[7] and CSDMS (Community Surface Dynamics Modeling System).[8] All of these systems require

models to have *initialize*, *run*, *finalize*, *get*, and *set* functions (examples are described below) for basic control over models. Also generally required by these couplers is a description of how the models are connected together, which is done in the form of an XML file or some other type of configuration file. OASIS and CSDMS have an additional advantage of having GUI's that allow placing the models and their connections into the interface for visualization and automated configuration file generation. CSDMS also uses Babel [9] to allow model writing in different languages (C++, C, Fortran, XML, Python, and Java are supported) whereas the other couplers are limited to models all being in the same language.

After consideration of the attributes of the various approaches, we have chosen to adapt the basic methodology of the CSDMS approach, which is based on the use of a low-level model-to-model interface, for linking materials models together. The purpose of this paper is to give an example of how to construct such an interface within a multiscale problem, with a focus on the types of information that must be passed from one model to another.

The biggest challenge for computational materials science is the extreme range of length and time scales that govern materials behavior, ranging from the Å and sub-picoseconds of atomistic behavior to the meters and years of materials in engineered applications. In between those extremes lie complex sets of physical phenomena that depend on the overall material behavior of interest and the type of material. We show a simple example of this range of mechanisms in Figure 1, highlighting the various scales that govern the mechanical behavior of metallic materials.[10] Consideration of other types of materials or other properties would lead to a similar, but not identical, figure.

Looking more closely at Figure 1, we see that at each scale there is a structural “unit” that dominates the physical processes at the given length and timescales. These units are the entities whose dynamics define the physics of interest at each scale. Typically, a set of models of materials behavior is created for each of the various scales.[2,11] These models are generally developed and used by different groups. In the traditional view of computational materials science, this hierarchy of models and simulations, each describing a specific scale and its associated phenomena, are linked to create a multiscale description of materials behavior. Typically, information is passed sequentially from scale to scale, an approach that is often

referred to as information passing or sequential multiscale.[1] Inherent in the information-passing paradigm is the lack of inverse models that would allow us to predict the needed structures and properties at the small scale from desired properties at a larger scale – information flows only one way. It is common that information passing is done by separate groups of researchers, using codes specific to the scales of their particular interests.

While the information-passing approach has proved useful for many systems and properties, it is, at its core, suspect in a number of ways. Information passing assumes that there is a separation of scales, which is not always clear. It is inherently a coarse-graining procedure, which loses information at each step, especially about the distribution of properties other than their mean. Indeed, this latter statement reflects the fundamental flaw of this approach – it assumes that we can start from fundamental laws and build up to a description of a macroscopic system, which is not obviously true.[12]

Another approach, which is often referred to as “concurrent multiscale”, [1] links simulations at differing scales directly within a single integrated framework. This approach has the advantage of enabling the flow of information between scales, which eliminates many of the uncertainties inherent in the information-passing paradigm. For recent examples, see [13-15]. As in sequential multiscale simulations, models must be developed and applied at multiple scales, with the added complication of developing appropriate interfaces between the models that enable them to be used concurrently. As we will discuss in detail below, the integrated approach described here provides for an efficient construction of integrated multiscale materials models.

We will present these ideas in the context of a specific problem, involving the interaction between an atomistic scale and a continuum scale, to serve as an example of the types of problems inherent in multiscale modeling of materials. Specifically, we use molecular dynamics to examine the interactions between a fluid and an underlying solid to set the boundary condition between a continuum fluid (described using the Lattice Boltzmann method) and the solid. Our focus is not on fluid properties, per se, but rather on how one connects an atomistic scale to a continuum scale in the context of creating a model-to-model interface to link together independent multiscale models.

We describe the basic methodologies in Section 2, followed by a description of our implementation of a basic model-to-model interface in Section 3, in which we discuss modifications to the CSDMS approach necessary for our test problem. Section 4 is a summary of results from an application of model, with comments about the efficiency with the model interface. Concluding remarks are given in Section 5.

2. Coupled Multiscale Model of Surface-Fluid Interactions in Couette Flow

The goal of this work is to provide a method for coupling multiscale simulations using a low-level model-to-model interface. We focus on an example problem in which we examine the role of surface interactions on Couette flow, which is the laminar flow of a viscous fluid held between two plates, with one plate moving at some velocity relative to the other, fixed, plate. In traditional computational solutions to Couette flow, the velocity of the fluid at the stationary plate (which we will refer to as the slip velocity) is assumed to be zero and a linear increase of velocity is seen between the two plates. In this sample problem, our goal is to examine the role of fluid-surface interactions on determining the slip velocity by incorporating realistic descriptions of the interactions between atoms in the fluid and those on the surface of the solid. We model the system using a lattice Boltzmann method for the continuum fluid flow coupled to a molecular dynamics simulation of the interaction between the atoms of the fluid and the underlying surface. The models operate at different scales and are based on different physics so provide a good case for understanding information mediation in multiscale simulations. We note that this problem has been used previously as an example of coupled multiscale simulations, though without the model integration strategy introduced here.[e.g., 16]

2.1 Lattice Boltzmann (LB) method

The lattice Boltzmann (LB) method [17] was introduced in the 1990s to address deficiencies in the lattice-gas cellular automaton model for fluid flow.[18] In the LB method, the single particles that move around the lattice in the lattice-gas method are replaced with single particle distribution functions, $f(\vec{x}, \vec{e}, t)$, in which f is the probability of finding a fluid particle at position $\vec{x}$ at time t with velocity $\vec{e}$. The probability distributions are discretized on a lattice

(with positions $\vec{x}$) and represented as $f_i(\vec{x}, t) = f(\vec{x}, \vec{e}_i, t)$, in which i indicates a direction to the nearest points in the lattice (as described below), $\vec{e}_i \Delta t$ is the displacement vector along direction i , and Δt is the time step.

The macroscopic quantities of density ρ and velocity $\vec{e}$ are found by evaluating the moments of f at each lattice site. With the requirement that the probabilities be normalized, i.e., that $\sum_i f_i(\vec{x}, t) = 1$, we have

$$\begin{aligned} \rho(\vec{x}, t) &= \rho_o \sum_i f_i(\vec{x}, t) \\ \rho(\vec{x}, t) \vec{u}(\vec{x}, t) &= \rho_o \sum_i \vec{e}_i(\vec{x}, t) f_i(\vec{x}, t) \end{aligned} \quad (1)$$

in which the sum is over the directions in the lattice and ρ_o is the density of the system (an input parameter). The equation of motion for the probability distributions can be shown to be [17]

$$f_i(\vec{x} + \vec{e}_i \Delta t, t + \Delta t) = f_i(\vec{x}, t) + \frac{1}{\tau} (f_i^{eq}(\vec{x}, t) - f_i(\vec{x}, t)), \quad (2)$$

where $f_i^{eq}(\vec{x}, t)$ is a local equilibrium distribution resulting from collisions, Δt is the time step and τ is a relaxation time. The equilibrium distributions are given by

$$f_i^{eq}(\vec{x}, t) = \omega_i \rho \left(1 + \frac{\vec{e}_i \cdot \vec{u}}{c_s^2} + \frac{(\vec{e}_i \cdot \vec{u})^2}{2c_s^4} - \frac{u^2}{2c_s^2} \right), \quad (3)$$

in which ω_i are a set of directional weights normalized to unity and c_s is the speed of sound. The ω_i are effectively quadrature weights in the numerical integration over a continuous distribution.[19]

In Figure 2 we show the grid used in this work. This lattice is generally referred to as the D3Q19 lattice (3 dimensions and 19 directions).[20] Shown in the figure are the lattice directions and velocity vectors, with the velocity vectors given by

$$\vec{e}_i = \begin{cases} (0,0,0) & i = 0 \\ (\pm c, 0, 0), (0, \pm c, 0), (0, 0, \pm c) & i = 1 - 6 \\ (\pm c, \pm c, 0), (0, \pm c, \pm c), (\pm c, 0, \pm c) & i = 7 - 18 \end{cases}, \quad (4)$$

in which $c = \Delta x / \Delta t$ and Δx is the grid size.[21] The speed of sound is given by $c_s^2 = c^2 / 3$ for the D3Q19 lattice. The weights are

$$W_i = \begin{cases} 1/3 & i = 0 \\ 1/18 & i = 1 - 6 \\ 1/36 & i = 7 - 18 \end{cases} \quad (5)$$

such that $\sum_{i=0}^{18} W_i = 1$. To account for constraints on the particle densities introduced by using content-velocity boundary conditions at the top and bottom of the simulation, an artificial force, Q , is applied to the unknown components of the particle density function.[22]

The solution to the LB equations has two steps, collision and advection. The collision equation is

$$f_i^{out}(\vec{x}, t) = f_i^{in}(\vec{x}, t) + \frac{1}{\tau} (f_i^{eq}(\vec{x}, t) - f_i^{in}(\vec{x}, t)) \quad (6)$$

and the advection equation

$$f_i^{in}(\vec{x} + \vec{e}_i \Delta t, t + \Delta t) = f_i^{out}(\vec{x}, t), \quad (7)$$

in which f_i^{out} denotes the outgoing (after collision) distribution functions while f_i^{in} denotes the incoming (before collision) distribution functions. Note that the LB calculations are all done in reduced units with unit densities, unit time steps, and unit distances. The calculations are scaled to specific units as specified below.

2.2 Molecular dynamics (MD)

We employed a constant-volume molecular dynamics (MD) simulation to model the interaction of a fluid and a solid. We solved the equations of motion using the velocity Verlet algorithm [23] and controlled the temperature using the Nosé-Hoover thermostat.[24,25]. Details of such simulations are given elsewhere.[10,26]

Lennard-Jones potentials were used to describe the interactions between the atoms in the fluid and between the atoms in the fluid and the solid. Specifically the potential between atoms in the fluid takes the form [27]

$$f(r) = 4 \epsilon \left[\left(\frac{S}{r} \right)^{12} - \left(\frac{S}{r} \right)^6 \right]. \quad (8)$$

Details of the choice of potential parameters are described below. A cutoff distance of $r_c = 2.72S$ is used. The MD simulations were run in standard reduced units, with the reduced total, kinetic and potential energies given by $E^* = E/\epsilon$, $K^* = K/\epsilon$, and $U^* = U/\epsilon$, respectively. Other reduced quantities are: distance: $r^* = r/\sigma$; pressure: $P^* = P \sigma^3/\epsilon$; temperature: $T^* = k_B T/\epsilon$; density: $\rho^* = \rho \sigma^3$; and time: $t^* = t/t_o$, where $t_o = \sigma (m/\epsilon)^{1/2}$, where m is the mass of the atom.

The constant velocity boundary condition at the top of the system continuously adds energy to the system, which, if left unchecked, would lead to viscous heating and a steady increase in the temperature. The resultant non-uniform temperature profile can be studied either by solving the heat equation with a source term representing viscous forces [28] or by using molecular dynamics simulations with a thermostat applied to the wall atoms.[29] A complete treatment of the viscous heating would require the incorporation of heat flow into the fluid flow simulation, either through a thermal Lattice Boltzmann model [30] or by coupling an additional model for the heat flow to both the Lattice Boltzmann and molecular dynamics models. For this paper, in which the goal is to examine the challenges inherent in linking models across scales, we chose to simplify the problem and include only a Nosé-Hoover thermostat on the fluid atoms, which we apply to the x -components of the velocities, perpendicular to both the flow direction and the height of the simulation cell.

2.3 Interface between lattice Boltzmann (LB) and molecular dynamics (MD)

As noted above, our goal is to couple a lattice Boltzmann simulation of continuum Couette flow with a molecular dynamics simulation used to model the interactions of the fluid with the underlying surface. These simulations are done in three dimensions, using the basic grid shown schematically in Figure 3. The system is divided into vertical cells with flow in the cells being calculated with the LB method. In a traditional simulation of Couette flow, the driving velocity

in the y direction would be set at the upper surface ($z = 1$ in Figure 3) and a zero-velocity boundary condition would be used at the bottom ($z = 0$) boundary. In the present simulation, we employ a MD simulation in the bottom cell to set the velocity boundary condition for the LB calculation.

The MD simulation cell is shown schematically in Figure 4. The boundary condition at the bottom edge arises from the influence of the interaction of the fluid atoms with a set of stationary atoms in a face-centered cubic (fcc) structure. The strength of the interaction between the fluid atoms and the solid atoms, the well depth ε_s , was varied to examine the role it plays on the slip velocity. The top atoms (light gray) have a density and interactions appropriate for the fluid but are kept in a fixed fcc structure and are given a velocity in the y direction that comes from the LB calculation.

The basic procedure is as follows: an LB calculation is run assuming a zero-velocity boundary condition at the bottom; the calculated velocity from the LB calculation at the top of the bottom cell is used to set the velocity at the top boundary of the MD calculation; the velocity at the bottom edge of the MD cell (the slip velocity) is determined (as described below); and the slip velocity sets the bottom boundary conditions for an LB calculation. The system is iterated to self-consistency.

2.4 Information transfer between models

In our approach, the LB and MD models have been written as autonomous codes that need input to run and that provide output. For the LB method, the needed input are the distance and time scales (Δx and Δt), the actual density, and the relaxation time τ . For MD, the needed input are the potential parameters (σ , ε , and ε_s) as well as atomic positions, temperature, cell dimensions, number of atoms, and time step, which are given in the code in reduced units.

In addition to these parameters, the boundary conditions must be set. The LB method requires the velocity at the top surface (the driving velocity) and bottom surface (the slip velocity). The driving velocity is set as an input, while the MD calculations provide the slip velocity. The MD calculations require the average velocity at its top surface, which is provided by the LB calculation.

One of the challenges in any multiscale simulation based on methods that describe different physical processes is to ensure a match between the fundamental thermodynamic, mechanical and transport properties described by each method. In general, the methods are based on different physical models and thus the information is calculated in very different ways. In the present case, we must ensure that the LB and MD calculations are modeling the same fluid, i.e., that each fluid has the same density, effective temperature, and viscosity. Details on making that connection are presented in the Appendix.

3. The Basic Model Interface (BMI)

Our goal is to create a standard model-to-model interface by which we can link autonomous models into an integrated modeling system for multiscale simulations. A detailed discussion of the reasons behind such an approach and more details of how to construct these interfaces is given elsewhere by the CSDMS group.[8] The basic goal of the CSDMS system is to create a common interface to link computer codes that have been developed by disparate members of their team to describe different aspects of the earth-surface system into an integrated simulation. They refer to these codes as “components,” which are equivalent to what we have been referring to as models. We will continue with the word model in the following discussion.

We emphasize that we are not discussing an interface between a user and a computer program, such as through a GUI. Rather, we are constructing an interface between models. The interface has no data associated with it, but rather is a specific set of common functions that define how information is transferred to and from various models. This interface thus allows the models to be connected into an integrated modeling system.

We use the word “model” in a very broad way. Each model is a computer code, which is written in some computer language and requires some input and yields some output, i.e., a “model” is an “information processor” that takes information in and sends new information out. The model may involve calculations that will be solved on a grid (or not) and will be written in some units (e.g., meters), which may or may not be consistent across all the models. The model could be a simple algebraic equation or it could be a complex computer simulation or a database - each

model acts as a source of information. The model interface makes communication between models independent of the models themselves, making integration of the models easier.

We believe that such an approach is essential for the efficient construction of integrated multiscale models in a environment in which models are widely different (e.g., lattice Boltzmann and molecular dynamics) and are likely to be developed and maintained by different groups. The benefits to this approach are that these groups can develop their models in any computer language; the models can be written in any sets of units; and the models can be calculated on any grid (or no grid). Each model can have multiple interfaces, so that we could link the same model in different ways to achieve multiple functionalities. For example, we could provide a set of interfaces to a molecular dynamics code that has many functionalities (constant volume and energy versus constant temperature versus constant stress, etc.). Thus this one model could be used in multiple ways by multiple people with no change in the fundamental computer code and can be reused in multiple integrated models without any special adaptations. The most important benefit is that constructing such autonomous models facilitates effective cooperation between groups, allowing each group to do what it does best - creating new models without having to focus on developing a new framework.

Our starting point is the CSDMS approach,[8] which enables a model to communicate with other models in a way that is accessible to the developers of those models. The CSDMS scheme is to do this in a structured way, as shown schematically in Figure 5. The basic model interface (BMI) is a set of functions that is attached to each model to enable it to fit into a second-level interface called the component model interface (CMI). The BMI uses very simple data types and is easy to implement, which facilitates its use by the developer community.[31] The BMI provides all the information that enable the model to be “plugged into” the integrated modeling application. The functions in the CMI allow the models to communicate and share data with other models regardless of computer language.[8] Service components are background packages that all models use to provide basic compatibility between languages, etc. The framework provides the means by which the modeler can assemble the models into a connected application. We note that in this paper we are using the same programming language for all models and so are focused on creating a low-level basic model interface.

Modeling materials has a number of unique challenges, including a tight coupling between scales as described in Figure 1. Our work is thus centered in part on developing a better understanding of model-to-model communication in multiscale materials simulations, i.e., on the mediation of information transfer between models across scales. Our approach has been to start with a relatively simple hybrid system of multiscale models (described in Section 2) to examine key aspects of information mediation for linked (concurrent) multiscale simulations, including

- compatibility of information transfer between models (i.e., does information have the right units, attributes, etc.)
- boundaries between models (how do we match boundary conditions)
- convergence of the overall solution, especially with highly disparate models
- stability of the solution

The Basic Model Interface (BMI) consists of a standardized set of functions that allow access to a model's input and output variables in a standard way.[31] Communication between models is handled mainly by the use of *get_()* and *set_()* functions that return and change the value of variable respectively. The full name of these functions will include the data type and rank of the variable. Appropriate use of these functions allows access to any of the models's input or output variables. While the models are allowed to do their internal calculations in whatever systems of units is most convenient, the values returned by *get_()* or input into *set_()* should be in some accepted set of units. In our case, we use SI units.

A list of functions and their descriptions is found in Table 1. We note that *get* and *set* functions exist for variables of every combination of data type and rank and that we only list a few of them in Table I. While many of these functions are similar or identical to those from the BMI, [8,31] new functions have been added for information about the potential boundary conditions that can be applied to a model and what input and output variables to use with a specific boundary condition. Specifically, the function *get_boundary_condition_names()* returns types of boundary conditions that a model is capable of enforcing and *get_boundary_var_names(string)* returns a list of names of variables associated with a given boundary condition. The function *match_units(model *)* has also been added and is used to ensure that two models are simulating the same material under the same conditions. This function enables the model calling it to

1
2
3
4 examine the state variables of the input model and modify its own state variables to match them,
5
6 as discussed in the Appendix.
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

Table I. List of BMI-like Functions

function	purpose
void initialize(string input_file, string identifier)	allocates memory for model and sets input variables
void run(int time_steps)	runs model for number of time steps based on value of time_steps
void finalize()	deallocates memory for model and prints output to a file
vector <string> get_input_var_names()	returns list of input variables
vector <string> get_output_var_names()	returns list of output variables
vector <string> get_boundary_condition_names()	returns list of usable boundary conditions
vector <string> get_boundary_condition_var_names(string boundary_condition)	returns list of variables to use to enforce given boundary condition
string get_var_type(string variable)	returns data type of variable
string get_var_units(string variable)	returns units of variable
int get_var_rank(string variable)	returns rank of variable
double get_0d_double(string)	returns value of a zeroth rank floating point variable
vector <double> get_1d_double(string)	returns a first rank floating point variable
void set_2d_double_at_index(string, double, int, int)	set the value of a second rank floating point variable at a specified index
int get_3d_int_at_index(string, int, int, int)	return the value of a third rank integer variable at a specified index
vector < vector < vector < vector < string> > > > get_4d_string(string)	return a fourth rank string variable
void match_units(model *)	matches the values of variables in two models to put them into the same state

The framework in Figure 5 provides access to linking autonomous models within a run-time environment. To demonstrate the benefits of this approach, however, we are using a simpler system, in which there is a “controlling model” that serves the role as both the CMI and framework. The MD code and LB code are implemented as subclasses of the controlling model, so they inherit all of the functions, while the implementation for each function is specific to a given model. The BMI enables the models to communicate and share data with each other regardless of the language, though in our case the two models are written in a common language (C++). We emphasize again that since the models communicate through the functions in Table I (and others), it is immaterial that they are on a different grid (or no grid at all), use different units, or use a different time stepping scheme. The boundaries between models are controlled through the model-to-model interface. The controlling model creates and initializes a object of each model class, *md* and *lb*, and uses *match_units* to set the *density*, *temperature*, and *viscosity* to be the same in both models. It then connects the LB/MD simulations as described above.

4. Results of the Model

We developed and implemented a BMI-like interface to link the LB and the MD models, with the goal of (1) establishing a structure for connecting multiscale materials models and (2) demonstrating that this structure can be used to solve a specific problem, which, in this case, is to evaluate the effects of surface interactions at a fluid-solid interface. Here we describe the simulations in more detail and present some results. We note that detailed comparisons were made between an original monolithic code and the BMI-linked application, with no loss in computational speed or accuracy arising from use of the BMI-linked code.

Table II: Physical parameters input to each model[‡]

<i>Lattice Boltzmann</i>	<i>Molecular Dynamics</i>
$\Delta x = 5.78 \text{ nm}$	$N = 2400$
$\Delta t = 4.363 \times 10^{-11} \text{ s}$	$m_a = 6.63 \times 10^{-26} \text{ kg}$
$\rho_o = 1374 \text{ kg/m}^3$	$\varepsilon = 120 \text{ K} = 1.656 \times 10^{-21} \text{ J}$
$\tau = 1$	$\sigma = 0.34 \text{ nm}$
	$a_{MD} = 17 \sigma = \Delta x$
	$V_{cell} = 1.93 \times 10^{-25} \text{ m}^3$
	$T = 1.3 \varepsilon/k_b = 156 \text{ K}$
	$\delta t = 0.001 t_o = 2.2 \times 10^{-15} \text{ s}$

[‡] Δx , Δt , ρ_o , and τ are defined in Eq. (1), (2) and (5). N is the number of freely flowing fluid atoms in the molecular dynamics cell, m_a is the mass per atom, ε and σ are the potential parameters defined in Eq. (9), a_{MD} is the size of molecular dynamics cell ($= \Delta x$), V_{cell} is the volume of the molecular dynamics cell (and the LB grid volume), T is the temperature, and δt is the time step in the solution to the equations of motion. t_o is the reduced unit of time in the calculation, as described in the text.

The basic parameters in the modeled system are given in Table II. The interaction potential between the fluid atoms and the underlying solid atoms (as shown in Figure 4), ε_s , is varied. The velocity profile in the y direction from an MD simulation is found by determining the average velocity in small volume at different z values in the MD cell (indicated in Figure 4). These are then fitted with a line and the slip velocity (the value at the surface) is determined, which is passed to the LB model as the boundary condition at the bottom surface. The LB simulations were run for 1000 time steps before passing information to the MD calculations, which were then run for 20000 time steps to equilibrate the fluid with the new boundary condition and 600,000 time steps to create the velocity profile and to determine the slip velocity, which is passed to the LB calculation. This process continues for 20 cycles. The average of the slip velocity from the last 10 cycles is taken as the slip velocity and its standard deviation is taken as a measure of the error of the method.

We examined two types of substrate surfaces. In Figure 6a we show a “smooth” surface, consisting of the [100] surface of a face-centered cubic lattice. In Figure 6b, a “rough surface” is created by removing 1/2 of the surface atoms from Figure 6a, keeping a fourfold symmetry. We note that the interaction energy between these atomic-scale surfaces and the fluid is not

sufficient, in general, to lead to zero slip velocity, which is why we are using a solid density at an unphysical value of four times that of the fluid.[16] To obtain non-zero slip velocities for the atomically rough surface we had to increase the driving velocity at the top of the Couette flow relative to that used for a smooth surface (30 LB units, or 3974 m/s for the rough surface and 0.75 LB units or 99.3 m/s for the smooth surface).

In Figure 7, we show the calculated slip velocity as a function of the number of cycles in the iterative procedure. Calculations were done with $\epsilon_s/\epsilon = 0.2$ for the rough surface and 0.6 for the smooth surface. As noted above, the velocities with the smooth surface are small enough that numerical noise is relatively larger than that for the rough surface. That said, the results converge reasonably well for both, with larger relative error for the smooth surface. Note that the surface velocities are much smaller than the driving velocities (at the top of the LB model). For example, for the smooth surface, the ratio of the slip velocity to the driving velocity in Figure 7a is about $6/99 \sim 6\%$ and for the rough surface in Figure 7b, the ratio is about 0.8%. From a continuum perspective the assumption of a zero slip velocity may be a reasonable boundary condition, depending on the driving velocity and the strength of the fluid solid interaction.

In Figure 8, we show the converged slip velocity as a function of ϵ_s/ϵ for the smooth and rough surfaces. In both cases, the stronger the interaction energy between the fluid atoms and the solid atoms, the lower the slip velocity, which is certainly as one would expect. For the smooth surface (Figure 8a), the net interactions between the substrate surface and the atoms in the fluid were quite small, and very small net fluid velocities were sufficient to drive the atoms past the surface. Moving to an atomically rough surface changed the dynamics of the solid-liquid interface dramatically (Figure 8b), greatly increasing the effects of the surface on the fluid flow at the interface.

6. Summary and Conclusions

The results from the simulations of Couette flow are consistent with expectations and with previous work,[16] with the stronger the interaction energy between the fluid atoms and the solid atoms, the lower the slip velocity (see Figures 7 and 8). We found that an surface with

roughness at an atomic scale enhanced the interactions between atoms in a fluid and the surface than that found with an atomically-smooth surface. The iterative approach taken to link the models was satisfactory, though one could imagine a number of other coupling schemes.

The important part of this paper is that we established a basic model-to-model interface for linking autonomous multiscale computer models. We specifically examined key aspects of the mediation of the information between the models. A challenge for materials modeling is that the physical behavior at a specific set of length and time scales generally depends on physical processes at smaller scales, which can lead to complex boundary conditions between models. Because of this complexity, we focused on the key issues of

- compatibility of information transfer between models (i.e., does information have the right units, attributes, etc.)
- boundaries between models (how do we match boundary conditions for widely different length and time scales)
- convergence of the overall solution, especially with highly disparate models
- stability of the solution, especially with highly disparate models

We introduced new functionality to the Basic Model Interface of the CSDMS group [8] to handle these issues. Specifically, we introduced new functions to force compatibility between the LB and MD models to consistently represent fluids in the same state (detailed in the Appendix) and to set the boundaries between the models (detailed in Section 3). Convergence and stability were managed in the “controlling model” and were a product of the choice of an iterative link between scales. More sophisticated calculations will most likely require more sophisticated boundary conditions, which are under investigation.

We are currently adding the model-to-model interface to other models (e.g., heat flow and a phase-field model) for application to problems of more relevance to materials (e.g., solidification). The goal is to create a library of models that we can link together as needed to form an integrated model system for a specific application that requires no further modification of the models in the library.

We note that a more effective approach to developing this type of integrated model system would be to develop autonomous models with ontological and semantic independence.[32]

1
2
3
4 Autonomy refers to models existing and running independently of other models. Ontological and
5
6 semantic independence mean that each model's description of its own data and how the code is
7
8 written is independent of how other models work and how they describe their data. Models
9
10 could then be developed independently of one another and a way to link them together as
11
12 necessary would be employed.
13
14

15 16 **Funding** 17

18
19 This work was funded by the US Department of Energy, Office of Fossil Energy and was
20
21 performed at the Department of Energy's Ames Laboratory under Contract No. DE-AC02-
22
23 07CH11358.
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

References

1. J. Fish, Bridging the scales in nano engineering and science, J. Nanopart. Res. 8 (2006) 577.
2. J. H. Panchal, S. R. Kalidindi, D. L. McDowell, Key computational modeling issues in Integrated Computational Materials Engineering, Comput.-Aided Des. 45 (2013) 4-25.
3. G. F. Laniak, G. Olchin, J. Goodall, A. Voinov, M. Hill, P. Glynn, G. Whelan, G. Geller, N. Quinn, M. Blind, S. Peckham, S. Reaney, N. Gaber, R. Kennedy, A. Hughes, Integrated environmental modeling: A vision and roadmap for the future, Environ. Modell. Softw. 39 (2013) 3-23.
4. D. J. Muth and K. M. Bryden, An integrated model for assessment of sustainable agricultural residue removal limits for bioenergy systems, Environ. Modell. Softw. 39 (2013) 50-69.
5. C. W. Armstrong, R. W. Ford, G. D. Riley, Coupling integrated Earth system model components with BFG2, Concurr. Comp.-Pract. E. 21 (2009) 767-791.
6. V. Balaji, The FMS Manual: A developer's guide to the GFDL Flexible Modeling System (2002). URL: https://www.gfdl.noaa.gov/cms-filesystem-action/tools_and_data/fms/FMSManual.pdf (downloaded 18 October 2015)
7. R. Redler, S. Valcke, H. Ritzdorf, OASIS4, a coupling software for next generation Earth system modeling, Geosci. Model Dev. 3 (2010) 87-104.
8. S. D. Peckham, E. W. H. Hutton, B. Norris, A component-based approach to integrated modeling in the geosciences: the design of CSDMS, Comput. Geosci. 53 (2013) 3-12.
9. Babel Homepage, URL: <https://computation.llnl.gov/casc/components/index.html#page=home>, (downloaded 29 December 2015).

10. M. F. Ashby, Physical modeling of materials problems, Mater. Sci. Tech. 8 (1992) 102–111.
11. R. LeSar, Introduction to computational materials science: Fundamentals to applications, Cambridge University Press, Cambridge UK, (2013).
12. P. W. Anderson, More is different, Science 177 (1972) 393-396 (1972)
13. S. Ghosh, Adaptive hierarchical-concurrent multiscale modeling of ductile failure in heterogeneous metallic materials, JOM-J. Min. Met. Mat. S. 67 (2015) 1.
14. Q. Tong, S. Li, From molecular systems to continuum solids: a multiscale structure and dynamics, J. Chem. Phys. 143 (2015) 064101.
15. F. J. Vernery, M. Kabiri, An adaptive concurrent multiscale method for microstructured elastic solids, Comput. Method Appl. M. 241-244 (2012) 52-64.
16. N. Asproulis, M. Kalweit, D. Drikakis, A hybrid molecular continuum method using point wise coupling, Adv. Eng. Softw. 46 (2012) 85–92.
17. S. Chen, G. D. Doolen, Lattice Boltzmann method for fluid flows, Annu. Rev. Fluid Mech. 30 (1998) 329-364.
18. U. Frisch, B. Hasslacher, Y. Pomeau, Lattice-gas automata for the Navier-Stokes equation, Phys. Rev. Lett. 56 (1986) 1505–1508.
19. B. Chopard, M. Droz, Cellular Automata Modeling of Physical Systems, Cambridge University Press (1998).
20. D. d’Humières, I. Ginzburg, M. Frafczyk, P. Lallemand, L. Luo, Multiple-relaxation-time lattice Boltzmann models in three dimensions Philos. T. Roy. Soc. A 360 (2002) 437-451.
21. C. Körner, E. Attar, P. Heinl, Mesoscopic simulation of selective beam melting processes, J. Mater. Process. Tech. 211 (2011), 978-987.

- 1
2
3
4 22. C. F. Ho, C. Chang, K. H. Lin, C. A. Lin, Consistent Boundary Conditions for 2D and 3D
5 lattice Boltzmann simulations, CMES-Comp. Model. Eng. 44 (2009) 137-155.
6
7
8
9 23. L. Verlet, Computer “experiments” on classical fluid. I. thermodynamical properties of
10 Lennard-Jones molecules, Phys. Rev. 159 (1967) 98–103.
11
12
13 24. S. Nosé, A molecular dynamics method for simulations in the canonical ensemble, Mol.
14 Phys. 52 (1984) 255–268.
15
16
17 25. W. G. Hoover, Canonical dynamics: equilibrium phase-space distributions, Phys. Rev. A
18 31 (1985) 1695–1697.
19
20
21
22 26. D. Frenkel, B. Smit, B. (2002). Understanding molecular simulation: From
23 algorithms to applications (2nd ed.), Academic Press, San Diego, CA (2002).
24
25
26 27. J. E. Jones, On the determination of molecular fields. - II. From the equation of
27 state of a gas, P. Roy. Soc. A-Math Phy. 106 (1924) 463-477.
28
29
30
31 28. J. N. Reddy, An introduction to continuum mechanics: With applications,
32 Cambridge University Press, New York (2008).
33
34
35 29. X. Yong, X., L. T. Zhang, Thermostats and thermostat strategies for molecular
36 dynamics simulations of nanofluidics, J. Chem. Phys. 138 (2013) 084503.
37
38
39 30. X. He, S. Chen, G. D. Doolen, A novel thermal model for the Lattice Boltzmann
40 method in incompressible limit, J. Comput. Phys. 146 (1998) 282-300.
41
42
43
44 31. CSDMS Basic Model Interface. URL: http://csdms.colorado.edu/wiki/BMI_Description
45 (downloaded 18 October 2014)
46
47
48 32. K. M. Bryden, A proposed approach to the development of federated model sets, In:
49 Ames, D.P., Quinn, N.W.T., Rizzoli, A.E. (Eds.), Proceedings of the 7th International
50 Congress on Environmental Modelling and Software, June 15-19, 2014. San Diego,
51 California, USA, 2014. ISBN: 978-88-9035-744-2, (2014).
52
53
54
55
56
57 <http://www.iemss.org/sites/iemss2014/proceedings.php>
58
59
60
61
62
63
64
65

- 1
2
3
4 33. R. L. Rowley, M. M. Painter, Diffusion and viscosity equations of state for a Lennard-
5
6 Jones fluid obtained from molecular dynamics simulations, Int. J. Thermophys. 18,
7
8 (1997) 1109-1121.
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

Table AI. Equivalence of quantities between lattice Boltzmann and molecular dynamics.

Property	Lattice Boltzmann	Molecular Dynamics
Density	$r = r_o \sum_{i=0}^{18} f_i = 1374 \frac{\text{kg}}{\text{m}^3}$	$r = \frac{4m_a}{(1.7S)^3} = 1374 \frac{\text{kg}}{\text{m}^3}$
Temperature	T = 156 K (arbitrary)	T = 156 K (thermostat)
Viscosity	$\eta = \frac{r}{3} \frac{Dx^2}{Dt} (t - 0.5) = 1.75 \times 10^{-4} \frac{\text{kg}}{\text{m s}}$	$\eta^* = 1.934$ $\eta = \eta^* \frac{\sqrt{m_a e}}{S^2} = 1.75 \times 10^{-4} \frac{\text{kg}}{\text{m s}}$
Velocity Conversion Factor [‡]	$v_{LB} = g_{MD \rightarrow LB} v_{MD}$ $g_{MD \rightarrow LB} = \frac{Dx}{Dt} \frac{dt}{a_{MD}} = 0.0493$	$v_{MD} = g_{LB \rightarrow MD} v_{LB}$ $g_{LB \rightarrow MD} = 1/g_{MD \rightarrow LB} = 20.3$

[‡] Factor to convert velocities in the molecular dynamics calculations to those in the lattice Boltzmann simulations, and *vice versa*.

Figure captions:

Figure 1: Length and time scales in materials science adapted from Ashby [10] On the left, we indicate the important unit structure at each scale, in the middle, the approximate length and time scales, and at the right, the approach used to simulate the material's mechanical behavior. We note that the approaches used at each scale are generally distinct to that scale and, most often, are developed and used by different research groups.

Figure 2. D3Q19 lattice showing the lattice directions. The numbering scheme is described in the text. [18]

Figure 3. Schematic view of the Lattice Boltzmann grid, with the region in which molecular dynamics is used shown in gray.

Figure 4: Schematic two-dimensional view of the molecular dynamics cell (a 2D projection). The dark colored atoms at the bottom represent the solid. The shaded atoms at the top boundary are given a velocity set by the results of the LB calculation. The rectangles indicate volumes over which averages are taken to determine the velocities as a function of z . Details are in the text.

Figure 5. Relationship between the components of the CSDMS schema. Adapted from [7].

Figure 6. Atomic-level structures for the two surfaces used in our study. (a) The [100] surface of a face-centered cubic lattice. Note that it is smooth at an atomistic scale. (b) The surface from (a) with 1/2 of the atoms removed. Note that the surface structure has a fourfold symmetry.

Figure 7. The slip velocities as a function of iterative cycle for (a) the atomically-smooth surface of Figure 6a and (b) the atomically-rough surface of Figure 6b.

Figure 8. The slip velocities v_s (the average fluid velocity at the substrate surface) as a function of the strength of the interatomic interaction (ϵ_s/ϵ) between the atoms in the fluid and those in the solid substrate. (a) The atomically-smooth surface of Figure 6a. (b) The atomically-rough surface of Figure 6b.

Unit	Length Scale	Time Scale	Mechanics
Complex Structure	10^3 m	10^6 s	Structural Mechanics
Simple Structure	10^1 m	10^3 s	Fracture Mechanics
Component	10^{-1} m	10^0 s	Continuum mechanics
Grain Microstructure	10^{-3} m	10^{-3} s	Crystal plasticity
Dislocation Microstructure	10^{-5} m	10^{-6} s	Micro-mechanics
Single Dislocation	10^{-7} m	10^{-9} s	Dislocation Dynamics
Atomic	10^{-9} m	10^{-12} s	Molecular Dynamics
Electron Orbitals	10^{-11} m	10^{-15} s	Quantum Mechanics

Figure 1: Length and time scales in materials science adapted from Ashby [10] On the left, we indicate the important unit structure at each scale, in the middle, the approximate length and time scales, and at the right, the approach used to simulate the material's mechanical behavior. We note that the approaches used at each scale are generally distinct to that scale and, most often, are developed and used by different research groups.

Figure 2. D3Q19 lattice showing the lattice directions. The numbering scheme is described in the text. [18]

Figure 3. Schematic view of the Lattice Boltzmann grid, with the region in which molecular dynamics is used shown in gray.

Figure 4: Schematic two-dimensional view of the molecular dynamics cell (a 2D projection). The dark colored atoms at the bottom represent the solid. The shaded atoms at the top boundary are given a velocity set by the results of the LB calculation. The rectangles indicate volumes over which averages are taken to determine the velocities as a function of z . Details are in the text.

Figure 5. Relationship between the components of the CSDMS schema. Adapted from [7].

(a)

(b)

Figure 6. Atomic-level structures for the two surfaces used in our study. (a) The [100] surface of a face-centered cubic lattice. Note that it is smooth at an atomistic scale. (b) The surface from (a) with 1/2 of the atoms removed. Note that the surface structure has a fourfold symmetry.

(a)

(b)

Figure 7. The slip velocities as a function of iterative cycle for (a) the atomically-smooth surface of Figure 6a and (b) the atomically-rough surface of Figure 6b.

(a)

(b)

Figure 8. The slip velocities v_s (the average fluid velocity at the substrate surface) as a function of the strength of the interatomic interaction (ϵ_s/ϵ) between the atoms in the fluid and those in the solid substrate. (a) The atomically-smooth surface of Figure 6a. (b) The atomically-rough surface of Figure 6b.