ORAL REFERENCE: ICF100913OR

SOFTWARE METHODOLOGIES FOR MULTISCALE DESCRIPTIONS OF DEFECTS, DEFORMATION AND FRACTURE

C. R. Myers[1], T. Cretegny[2], N.P. Bailey[2], C.-S. Chen[1], A.J. Dolgert[2], L.O. Eastgate[2], E. Iesulauro[3], A. R. Ingraffea[3], M. Rauscher[2], and J.P. Sethna[2]

1 Cornell Theory Center, Cornell University, Ithaca, NY 14853 USA
2 Laboratory of Atomic and Solid State Physics, Cornell University, Ithaca, NY 14853 USA
3 Department of Civil and Environmental Engineering, Cornell University, Ithaca, NY 14853 USA

ABSTRACT

We examine some of the software implications of multiscale modeling of material deformation and fracture. Research in this field is aided through the development of flexible software frameworks for managing disparate degrees-of-freedom, constructing complex models, and interrogating simulation data at various scales. We describe some of the highlights of our Digital Material system for multiscale materials modeling, and discuss its use in the study of intergranular fracture in polycrystals.

KEYWORDS

Multiscale modeling, fracture, software engineering, Digital Material, grain boundary decohesion, molecular dynamics

INTRODUCTION

Computational mechanics, in general, and computational fracture mechanics, in particular, are becoming increasingly multidisciplinary. This trend is driven to a large extent by a desire and need to model material structures at many length and time scales, in order to more faithfully capture processes of relevance to material deformation and failure. Modeling across scales, however, introduces a new set of challenges, such as the need to develop frameworks for composing software components and material descriptions, manage disparate computational degrees-of-freedom in a consistent fashion, and construct algorithms that can exploit multiscale representations.

In this paper, we highlight some of the software challenges posed by multiscale modeling of materials deformation and failure, and discuss some of the approaches we have taken to build useful computational problem-solving environments for the study of such problems. We will discuss an application example using some of these tools, namely the modeling of decohesion of grain boundaries in polycrystalline materials.

MULTISCALE MODELING OF MATERIAL DEFORMATION AND FAILURE

Multiscale modeling is in many ways a problem of optimization, balancing the desire for greater fidelity (gleaned from explicit inclusion of smaller-scale degrees-of-freedom) with the need for compact and efficient modeling forms (typically embodied in more coarse-grained descriptions). One is striving, in some sense, for the smallest possible set of modeling degrees-of-freedom that are capable of describing the phenomena of interest. Conventional single-scale models, such as descriptions of continuum fields evolving according to constitutive laws, are able to offer compact representations, but often at the expense of having ad hoc constitutive models characterized by a large and unsystematic set of adjustable parameters. One of the benefits of multiscale modeling is that the space of material behaviors being approximated by these ad hoc constitutive models can be factored into a geometric piece and a functional piece. Collections of smaller-scale structures, described in turn by their own constitutive laws (which one hopes are more fundamental and less ad hoc than the more coarse-grained constitutive laws), are allowed to organize themselves geometrically in order to produce an effective constitutive description at larger-scales. The recently developed quasicontinuum method [1], for example, can be thought of as an extension of the finite-element method for solving continuum models of material response, where atoms are introduced and allowed to organize under their own mutual self-interaction in order to provide better constitutive descriptions of materials containing lattice defects. In many situations, however, it is far preferable to perform multiscale modeling implicitly, by finding a reduced-order description of smaller-scale processes for inclusion at larger scales. One approach of ours has been to focus on the development of appropriate *functional forms* that are capable of capturing the smaller-scale behavior [2]. In many cases, these functional forms have nonanalytic behavior (e.g., near bifurcations, or near configurations with high symmetry) that will dictate what small-scale information is most crucial for accurate modeling. It is important to study coarse-grained theories to try to determine what information from smaller scales is needed, rather than simply "throwing over the wall" information from small scales for use elsewhere.

Modeling of the behavior of materials across scales, however, differs from standard multiscale techniques in that the relevant material degrees-of-freedom vary widely from scale to scale. (Conventional numerical methods designed to address problems with structure on many scales benefit from the self-similarity of those methods across scales. Techniques such as multigrid [3] and multiresolution wavelet analyses [4] introduce bases or other discrete representations of continuum fields which represent equivalent types of information at each scale.) Fracture mechanics, for example, has been studied computationally at many scales, involving a whole menagerie of material structures, including electrons, atoms, vacancies, dislocations, voids, dislocation structures, grains, grain boundaries, and various continuum fields (see Figure 1).

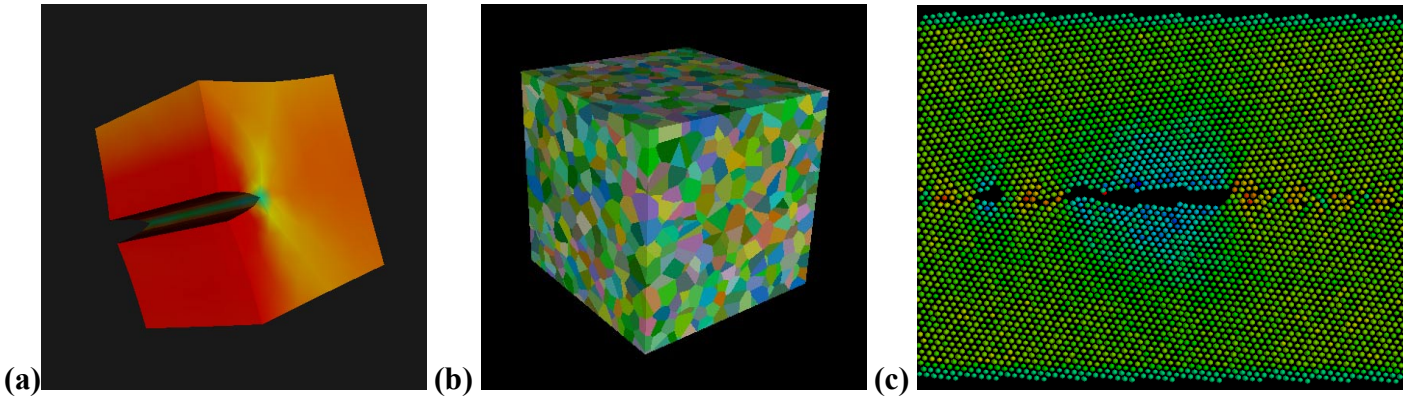

Figure 1. Models of materials at various scales. (a) Continuum fields at the macroscale; (b) Polycrystalline grain assemblies at the mesoscale; (c) Atoms near a grain boundary at the nanoscale.

Because of such diversity and heterogeneity, management of these degrees-of-freedom is significantly more complex than in methods such as multigrid. Furthermore, the fundamental algorithms required to efficiently

pass information across scales are not well-understood. Therefore, it is imperative that one construct software environments that provide flexible and expressive mechanisms for composing material descriptions and numerical kernels in order to experiment with different classes of models, algorithms, descriptions, etc.

This problem of *component composition* therefore becomes paramount. In order to explore a variety of material structures in a variety of different contexts, it is important to develop appropriate computational abstractions so that components may be reused. We would like to be able to synthesize, for example, the relevant pieces of code for molecular dynamics (MD) modeling with those for finite-element modeling in order to implement the quasicontinuum (QC) method, without having to maintain two separate bases of code for MD and QC. In addition, we would like to be able to compose high-level geometric descriptions (say, parameterizing a polycrystalline collection of grains with different lattice orientations) with underlying numerical models in order to build complex applications out of constituent pieces.

SOFTWARE ENGINEERING FOR MULTISCALE MODELING: DIGITAL MATERIAL

We have been addressing some of these issues through the development of *Digital Material,* an amalgam of software frameworks designed to support multiscale investigations of material structure and response [5]. This effort has many facets. Considerable activity has gone into developing a framework for atomistic simulations (e.g., MD, QC, etc.), which we will describe in more detail below. One application of the atomic-scale modeling framework is to provide quantitative input to mesoscale finite-element models of grain boundary fracture in polycrystals [6]. A different set of problems revolves around the use of phase-field models to study problems of interfacial evolution, including the formation and propagation of cracks [7]. Yet another area of investigation centers around the characterization of texture (orientation and misorientation distribution functions) in polycrystals [8], with one goal of relating texture to microcrack initiation and growth. Finally, we continue to work to build software bridges between these tools and more conventional finite-element software systems for investigating crack growth in continuum models [9].

The Digital Material system is built upon a set of broad software design philosophies. First, we have sought to develop abstractions that separate material structures from algorithms or numerical models that act on those structures. This is important because a central feature of multiscale modeling is that a given material structure can play different roles in different contexts, depending on how it fits into a larger material model. Second, in order to support the sort of flexible and expressive software composition described above, we make use of extensions to object-oriented design techniques such as *design patterns* [10] to encapsulate aspects of our programs that need to change. Design patterns deal with the collaboration between sets of computational objects. In our system, design patterns are used, for example, to enforce the separation between material structures and the

```
from MD3D import *
from BiCrystalInitializer import *

potential       = EMTPotential('Cu')
latticeConstant = potential.GetLengthScale() * sqrt(2)
lattice         = FCCLattice(latticeConstant)
atoms           = PrimitiveListOfAtoms()
cutoff          = potential.GetCutoffDistance()
neighborLocator = CellNeighborList(cutoff, .05*cutoff)

atoms.SetNeighborLocator(neighborLocator)
atoms.SetMass(63.54)

pbc = SimplePeriodicBoundaryConditions()
atoms.SetBoundaryConditions(pbc)

initializer = SymmetricalTiltGBInitializer(lattice,
            [lengthX, gbLength, minWidthZ],
            interfacePlane = [2, 2, 1],
            normalPlane = [1, -1, 0], bc)
initializer.Create(atoms)
print 'The number of atoms is :', atoms.GetNumber()
```

Figure 2. Sample Python script showing the initialization of some of the components used in an atomistic simulation of grain boundary decohesion.

algorithmic tools that act to modify or interrogate those structures. While some of the standard design patterns (e.g., observers) are appropriate for large-scale scientific computing, other patterns need to be developed in recognition of the constraints imposed by numerical modeling. In particular, the need for high computational performance has led us to phrase many of our basic abstractions in terms of aggregates of material objects (e.g., collections of atoms), so that those aggregates can be acted on efficiently without incurring a large computational overhead. Finally, we recognize the value of having high-level, interpreted control of our simulations, in order to support rapid prototyping, component composition, and interactive interrogation of simulation data. To achieve this, we use Python [11], an interpreted, object-oriented programming language, to drive many of our computational kernels and to "glue" together sets of computational objects. SWIG [12] is another system that assists in this process: SWIG generates the necessary glue/wrapper code for Python to talk to underlying C++ objects and functions, based on C++ interface declarations (e.g., class definitions and function prototypes). An example Python script for setting up an atomistic simulation of grain boundary fracture is shown in Figure 2. From within Python, users are able to manipulate references to compiled C++ objects as if they were native Python objects.

APPLICATION EXAMPLE: GRAIN-BOUNDARY FRACTURE

As mentioned, one of our goals is to provide quantitative input to mesoscale finite-element models of grain boundary fracture in polycrystals [6]. At the mesoscopic scale, the polycrystal is modeled via finite elements where the geometrical and physical properties of the grains are individually taken into account. At that scale, grain boundaries are modeled with interface elements, whose behavior is governed by a *cohesive zone model* (CZM), which relates the traction on the interface to its opening. We are conducting atomistic simulations of grain boundary decohesion, in part to provide appropriate CZMs for the mesoscale modeling. This work aims to develop not only cohesive models for a single grain boundary, but functional forms describing entire families of grain boundaries as parameterized by the orientations of the adjacent grains.

The main quantities we are interested in are the energy release rate, the peak stress and the range of the cohesive zone model. The energy release rate is the integral of the traction-separation curve, and represents the amount of energy per unit length that is dissipated in the crystal while the crack tip propagates. For a grain boundary in 3 dimensions, the CZM parameters will be functions of 5 independent degrees of freedom (DOFs): 3x2 rotational DOFs describing the orientation of the two adjacent grains, minus 1 DOF associated with invariance due to rotation of both grains about the normal to the interface. (We may choose to add other parameters, such as temperature, which play a role in the nature of the decohesion). Often one considers only 3 parameters, describing the interface misorientation. In doing so one assumes that the orientation of the interface does not play a significant role. This is a great simplification, but is hard to justify. For example, the limit of zero misorientation corresponds to the single crystal, for which lattice effects make crack propagation properties very anisotropic. On the other hand, special grain boundaries with reduced numbers of DOFs, such as symmetrical tilt grain boundaries (STGBs), can be usefully studied.

We have performed some preliminary simulations of STGBs, such as that depicted in Figure 3(a). These boundaries allow the use of periodic boundary conditions in the plane of the interface (typically around 10 atomic layers thick). In the orthogonal direction, we enforce prescribed displacement boundary conditions through the imposition of constraints on the outer two layers of atoms at either boundary. (Fixed displacement conditions on the outer layer results in a pinning of emitted dislocations at the outer boundary. Alternatively, constraining the center-of-mass motion of the outer layers allows such dislocations to exit the system. Switching between different types of constraints is straightforward in our system since constraints are specified in separate objects that are attached to the group of atoms being simulated.) We have performed dynamical simulations with a Verlet algorithm that periodically rescales the velocities of the atoms such that a fixed temperature can be prescribed.

We want to relate the opening of the interface and the traction in the surrounding bulk. Defining the opening of the grain boundary is a delicate task because from the atomistic point of view, there is no distinction between atoms that belong to the bulk and atoms that belong to the grain boundary. Briefly, we assume that on the external boundaries of the grains the stress is uniform. This assumption calls for a grain size large compared to the emergent inhomogeneities in the grain boundary. We also assume that the displacement of the atoms far from the interface can be written as $\overrightarrow{u_n} = (\mathbf{1}+\boldsymbol{\varepsilon})\cdot\overrightarrow{u_{n,0}} + \vec{\delta}$, where n is the index of the atom with position $\overrightarrow{u_{n,0}}$ in the unstrained grain boundary, $\boldsymbol{\varepsilon}$ is the strain tensor, and $\vec{\delta}$ is defined as the grain boundary opening. (More precisely, we can sum up the contributions to $\vec{\delta}$ from each grain). That the above decomposition can be done must be checked on a case-by-case basis. The actual determination of $\vec{\delta}$ and $\boldsymbol{\varepsilon}$ can be done by linear regression. The relevant points should be located as far as possible from the interface but not too close to the external constrained surface. We expect some inhomogeneities in the directions orthogonal to the GB, so we compute an average of $\vec{\delta}$ over the entire simulation box.

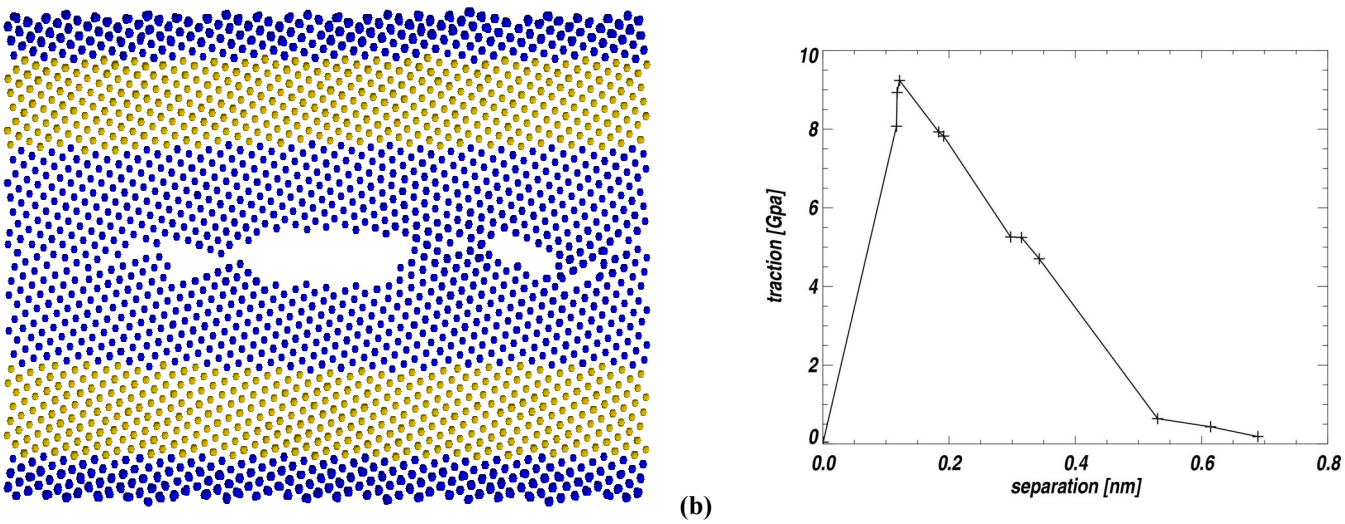

Figure 3. Decohesion of a [441] symmetric tilt grain boundary (i.e., the plane of the boundary is a [441] plane for each grain). (a) Snapshot from an MD simulation, highlighting those atoms used in the linear regression to determine the traction-separation relation. (b) Traction-separation curve derived from MD simulation.

A lingering difficulty is that the CZMs they provide are not, in and of themselves, so useful for the mesoscale finite element models. Typically the energy release rate and the range of the cohesive zone model are too small and the peak stress is too high. We are currently exploring reasons for these discrepancies. For example, the size of the atomistic system is very small compared to the typical size of the mesoscale interface elements: in an MD box the GB length is about 10 nm, while the width and thickness of the grains is of the order of 3 nm. With these dimensions the allowed ductility is quite small: the maximal opening will typically be on the order of a nanometer. One of our goals is to understand how the size of the damage zone scales with the dimensions of the atomistic simulation. Additionally, the properties of polycrystals crystals are often dominated by defects. Real materials will be substantially more heterogeneous than our simulation samples, and the large peak stress we measure in MD might naturally be explained by the lack of inelastic behavior driven by defects in the sample (e.g., acting as sites for the nucleation of dislocations or microvoids).

FUTURE DIRECTIONS

There are several directions in which this work ought to be extended. One important goal revolves around the notion of *adaptivity*. Our current system is built to allow researchers to experiment with different sorts of multiscale representations and algorithms, so that more formal and general multiscale methods can be identified. Our system encourages the development of heuristic approaches, which one would eventually like

to automate, so that the system itself could adaptively select the appropriate level of resolution to address a certain problem. Adaptivity needs to be broadly planned for at multiple levels: at the *application* level (where one might switch between models being solved based on the nature of the solution), at the *algorithm* level (where one might switch between different methods used to solve the problem at hand), and at the *system* level (where one might need to balance computational loads in response to changing resource requirements, or migrate to different platforms in response to changing resource availability).

Even with advanced software engineering techniques, the barriers to composing complex and adaptive applications are still substantial, and we must consider what other sorts of software generation technologies are available to assist in that construction process. One such system that we are beginning to explore is the *Loci* framework, developed by Ed Luke [13]. Loci is a system for synthesizing the control flow of large and complex applications by deducing that flow from dependency information among the computational components being assembled. While this technology is still under development and being extended to handle larger classes of numerical models, there is promise that a system such as Loci might automate the construction of complex, distributed, multiscale applications from high-level problem specifications.

ACKNOWLEDGEMENTS

We gratefully acknowledge support for this research from NSF awards 9873214 and 0085969, with additional infrastructure support through NSF award 9972853.

REFERENCES

1. Shenoy, V.B. Miller, R. Tadmor, E.B., Rodney, D., Phillips, R., and Ortiz, M. (1999). *Journal of Mech. Phys. Solids*. 47, 611.
2. Bailey, N.P., Sethna, J.P., and Myers, C.R. (2000). *Mat. Res. Soc. Symp. Proc.* 578, 249.
3. Brandt, A. (1977). *Mathematics of Computation* 31, 333.
4. Arias, T.A. (1999). *Reviews of Modern Physics* 71, 267.
5. Digital Material, http://www.tc.cornell.edu/Research/Multiscale/DigitalMaterial
6. Iesulauro, E., Dodhia, K., Cretegny, T., Chen, C.-S., Myers, C.R., and Ingraffea, A.R. (2001). "Continuum-atomistic modeling for crack initiation and propagation in polycrystals", to appear in Proceedings of the 10th International Conference on Fracture.
7. Eastgate, L.O., Sethna, J.P., Rauscher, M., Myers, C.R. and Chen, C.-S. (2001). "Modeling Crack Propagation: A Phase-Field Approach", to appear in Proceedings of the 10th International Conference on Fracture.
8. Myers, C.R., Loge, R.E., Arwade, S.R., Miller, M.P., and Dawson, P.R., in preparation.
9. Carter, B., Chen, C.-S., Chew, L.P., Chrisochoides, N., Gao, G.R., Heber, G., Ingraffea, A.R, Krause, R., Myers, C.R., Nave, D., Pingali, K., Stodghill, P., Vavasis, S., and Wawrzynek, P.A., (2000). In: *Lecture Notes in Computer Science*, pp. 443-449, Vol. 1800, J. Rolim (Ed.), Springer-Verlag, Heidelberg.
10. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA.
11. Python, http://www.python.org
12. SWIG, http://www.swig.org
13. Luke, E. (1999). In: *Proceedings of ISCOPE'99*, pp. 142-153, Matsuoka, S., Oldehoeft, R.R., Tholburn, M. (Eds.), Springer-Verlag, Heidelberg.