Journal of Computational Science 4 (2013) 401–411

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

An algorithm for massively parallel dislocation dynamics simulations of small scale plasticity

Kenneth W. Leiter, Joshua C. Crone, Jaroslaw Knap*

Computational Science and Engineering Branch, RDRL-CIH-C, US Army Research Laboratory, Aberdeen Proving Ground, MD 21005-5066, USA

ARTICLE INFO

Article history:
Received 30 October 2012
Received in revised form 8 February 2013
Accepted 17 February 2013
Available online 15 March 2013

Keywords:
Dislocation dynamics
Finite element method
Parallel computing

ABSTRACT

Accurate modeling of dislocation motion in bounded bodies is essential for the goal of obtaining desired properties, for example electronic or optical, of many microelectronic devices. At present, we lack high fidelity computer codes for such modeling that efficiently utilize modern parallel computer architectures. In contrast, many dislocation simulation codes are available for periodic or infinite bodies. In principle, these codes can be extended to allow for dislocation modeling in finite bodies. However, such extension may involve an additional solver to be employed, coupled with a dislocation simulation code. We present an algorithm for development of parallel dislocation simulation capability for bounded bodies based on such coupling. Subsequently, we analyze the performance of the algorithm for a demanding dislocation dynamics model problem.

Published by Elsevier B.V.

1. Introduction

Behavior of engineering materials is governed by the presence of defects. Among various material defects, dislocations play a pivotal role as primary carriers of inelastic deformation. While cooperative dislocation motion is reasonably well understood and described in macroscopic materials [1,2], its understanding for nano-scale materials is still lacking. A case in point is dislocation motion in microstructurally thin films [3,4]. A microstructurally thin film is a film whose characteristic dimension is comparable to the characteristic microstructural size. Today, most films that comprise integrated circuits, microelectronic devices or magnetic storage media are examples of microstructurally thin films. Understanding how microstructural thin films deform is critical to obtain desired electronic or optical properties of devices [5–7]. Dislocation motion is crucial in this deformation process.

Among countless modeling techniques of material science, dislocation dynamics [8–10] is most likely best suited to accurately describe motion of dislocations in thin films. While there exist several actively developed dislocation dynamics computer codes, including microMegas [11], Tridis [12], Micro3d [13], PARANOID [9,10], and ParaDiS [14], most can only handle simulations of macroscopic materials or simple bounded bodies, such as boxes or cylinders. Moreover, no existing dislocation dynamics code has demonstrated the ability to handle general bounded bodies with high dislocation content on modern parallel computer hardware.

The treatment of small scale plasticity by means of dislocation dynamics commonly requires solution of a boundary value problem of elasticity [15]. Although it is possible to implement routines to solve the boundary value problem directly into an existing dislocation dynamics code [13], efficient parallel implementation may be difficult because of the disparate treatment of the two problems. For example, different domain decompositions can be expected for the two problems in parallel computing environments, which may necessitate a significant amount of communication between them. Furthermore, methods of solving boundary value problems of elasticity are well established and efficient parallel computer codes based on these methods already exist [16,17]. Therefore, a potentially better strategy than direct implementation, and the one we explore in this article, is to couple an existing parallel dislocation dynamics code to an existing parallel boundary value problem solver. This choice has the advantage of introducing minimal modifications to the codes and promotes encapsulation of the distinct problems.

Efficient coupling of disparate computer codes spanning length and time scales is an active area of research in scientific computing [18]. The desire to couple codes is motivated by a need to capture more sophisticated scientific phenomena and take full advantage of large modern computational platforms as they approach the exascale. Over the years, many methodologies have been proposed to address the issue of efficient coupling [19–21]. One of these methodologies, the so-called cooperative parallelism approach, uses remote method invocation to spawn and communicate with numerous child programs. This approach has been successfully employed to couple a coarse parallel finite element application with a fine constitutive material model for multiscale simulation [22].

* Corresponding author. Tel.: +1 410 278 0420.
E-mail address: jaroslaw.knap@us.army.mil (J. Knap).

1877-7503/$ – see front matter. Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.jocs.2013.02.002

Cooperative parallelism appears ideal for loosely coupled applications where the number of child programs is unknown and where child programs can fail or be terminated by the parent. In addition, the cooperative parallelism methodology may be well suited for coupling codes with non-deterministic and dynamic communication patterns and where communication is short [23].

Cooperative parallelism, however, may not be well suited as a coupling choice for our application. The communication pattern between the dislocation dynamics and boundary value problem solver is deterministic and persistent. Moreover, a connection-oriented approach may be desirable since the coupled codes may exchange significant amounts of data.

Other coupling methodologies exist, but are not well-suited to modern parallel hardware [24] or are overly application specific for our needs [25,26]. For example, the Intercomm [25] and model coupling toolkit [26] projects provide mechanisms for communication and interpolation of data between two grid-based applications. In contrast, we require a more general coupling approach. To satisfy our requirements for a general coupling approach capable of efficiently handling communication of large amounts of data, we employ distributed shared memory as the coupling mechanism between a dislocation dynamics code and a boundary value problem solver [27]. In this article we describe the coupling algorithm in detail and evaluate parallel performance of the application for handling large dislocation dynamics simulations of small scale plasticity.

2. Dislocation dynamics

In recent years, meso-scale simulations based on dislocation dynamics have been the focus of many researchers [28,11,15,9,10,12,29,13,30,31,8,14,32]. While some minor differences between individual approaches exist, their general features are quite common. Usually, dislocation lines within a linear elastic body are discretized into a set of connected line segments. Subsequently, forces on degrees of freedom associated with these discretized dislocation segments are computed. These forces originate from dislocation line segments interactions via their stress fields and drag on dislocation motion. The drag forces on dislocation line segments are calculated in terms of their respective velocities by recourse to a drag function. The resulting ordinary differential equations are then integrated in time allowing for updated dislocation segment positions to be obtained. It is important to emphasize that during the course of their motion dislocation segments are permitted to intersect with one another. These intersections must be properly accounted for and are ordinarily handled by a set of well-defined topological operations. A concise representation of the main components of a dislocation dynamics simulation is shown in Fig. 1. An in-depth description of dislocation dynamics is beyond the scope of this article. We summarize, however, the main points of the theory below. In our summary we closely follow Bulatov and Cai [8] and Arsenlis et al. [14].

2.1. Representation of dislocation lines

In the classical theory of dislocations (c.f. [33,34]) individual dislocations are treated as lines contained within a linear elastic body $\mathcal{B} \subset \mathbb{R}^3$. These dislocation lines are constrained to terminate at surfaces of the body, $\partial\mathcal{B}$, but are otherwise free to move within $\mathcal{B}$. With each point along a dislocation line we associate the Burgers vector, $\mathbf{b} \in \mathbb{R}^3$, representing the direction of the local distortion of the crystal lattice related to the dislocation. The practicality of dislocation dynamics hinges upon a simplified treatment of these dislocation lines. To this end, each dislocation line is approximated by a set of straight dislocation segments $\mathcal{D} = \{s_1, s_2, s_3, \ldots\}$. In turn, each dislocation segment, s_i, is treated as an ordered pair of vertices, i.e. $s_i = (v_i^1, v_i^2)$, with the corresponding Burgers vector, $\mathbf{b}_i$. We moreover assume that v_i^2 and v_{i+1}^1 are equivalent, i.e. correspond to the same vertex. In addition, while vertices are allowed to belong to multiple dislocation segments, each dislocation segment may only belong to a single discretized dislocation line.

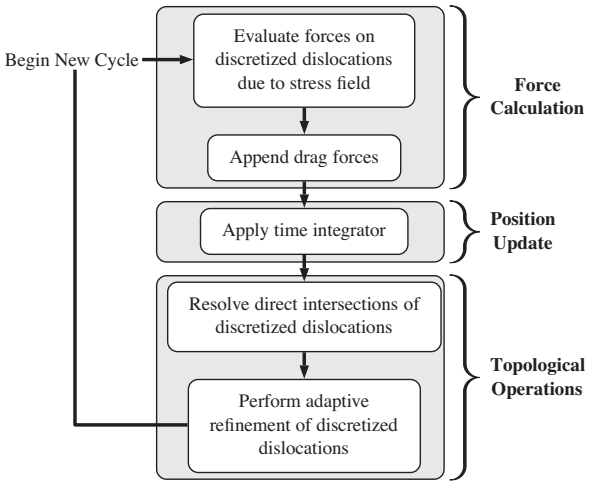

Fig. 1. The main components of a dislocation dynamics simulation.

With the above representation at hand, the coordinates of a point $P \in s_i$ along a discretized dislocation line $\mathcal{D}$, $\mathbf{x}(P)$, are readily computed by linear interpolation

$$\mathbf{x}(P) = \mathbf{x}_i^1 (1 - \xi) + \mathbf{x}_i^2 \, \xi, \tag{1}$$

where $\mathbf{x}_i^1, \mathbf{x}_i^2$ are, respectively, the coordinates of vertices v_i^1, v_i^2, and $\xi \in [0, 1]$ denotes the value of the parametric coordinate along s_i corresponding to P.

We emphasize that a choice of dislocation line representation is not, by any means, limited to the one introduced above. As a matter of fact, other representations have been successfully employed in dislocation dynamics simulations (e.g. [29,9]).

2.2. Computation of forces

The presence of dislocation lines in $\mathcal{B}$, along with applied tractions and displacements on $\partial\mathcal{B}$, induces at every point in $\mathcal{B}$ stress $\sigma \in Sym(\mathbb{R}^3)$ [33–35]. $Sym(\mathbb{R}^3)$ is the space of symmetric second-rank tensors over $\mathbb{R}^3$. This stress field acts on discretized dislocation lines yielding a force density (per unit length) $\mathbf{f}_i^{PK}$ at a point $P \in s_i$ with coordinates $\mathbf{x}(P)$

$$\mathbf{f}_i^{PK}(\mathbf{x}) = [\sigma(\mathbf{x})\mathbf{b}_i] \times \mathbf{t}_i, \tag{2}$$

where $\mathbf{b}_i$ is the segment Burgers vector, $\mathbf{t}_i$ is the tangent direction of s_i, and $\times$ is the vector (cross) product in $\mathbb{R}^3$. $\mathbf{f}_i^{PK}$ can be integrated along s_i and apportioned to each of the two segment vertices according to

$$\mathbf{f}_i^1 = |s_i| \int_0^1 \mathbf{f}_i^{PK}(\xi) \;\; (1 - \xi) \;\; d\xi, \tag{3}$$

$$\mathbf{f}_i^2 = |s_i| \int_0^1 \mathbf{f}_i^{PK}(\xi) \;\; \xi \;\; d\xi. \tag{4}$$

Here, $|s_i|$ denotes the length of s_i and ξ is the parametric coordinate along s_i. Individual segment force contributions (3) and (4) are accumulated at each vertex giving rise to a net vertex force.