Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 219 (2006) 608–621

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

# A parallel algorithm for 3D dislocation dynamics

Zhiqiang Wang [a,*], Nasr Ghoniem [a], Sriram Swaminarayan [b], Richard LeSar [b]

[a] *University of California – Los Angeles, Los Angeles, CA 90095-1597, USA*
[b] *University of California, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*

Received 14 April 2005; received in revised form 4 December 2005; accepted 4 April 2006
Available online 2 June 2006

**Abstract**

Dislocation dynamics (DD), a discrete dynamic simulation method in which dislocations are the fundamental entities, is a powerful tool for investigation of plasticity, deformation and fracture of materials at the micron length scale. However, severe computational difficulties arising from complex, long-range interactions between these curvilinear line defects limit the application of DD in the study of large-scale plastic deformation. We present here the development of a parallel algorithm for accelerated computer simulations of DD. By representing dislocations as a 3D set of dislocation particles, we show here that the problem of an interacting ensemble of dislocations can be converted to a problem of a particle ensemble, interacting with a long-range force field. A grid using binary space partitioning is constructed to keep track of node connectivity across domains. We demonstrate the computational efficiency of the parallel micro-plasticity code and discuss how O(N) methods map naturally onto the parallel data structure. Finally, we present results from applications of the parallel code to deformation in single crystal fcc metals.
© 2006 Elsevier Inc. All rights reserved.

*Keywords:* 3D dislocation dynamics; Parallel algorithm; Single crystal plasticity; Large scale simulation

## 1. Introduction

Dislocations are curvilinear defects in materials and are considered as the primary carrier of the plastic deformation. Dislocation microstructure evolution and dislocation pattern formation in deformed materials are crucial in determining the mechanical properties of these materials. Full understanding of the physical origin of plasticity requires fundamental knowledge of these phenomena, such as the occurrence of slip bands and cell structures.

Dislocation theory has been extensively studied and used to explain plasticity since 1930's. In the past decades, with the development of fast computers, a new methodology, called dislocation dynamics (DD), has been developed and successfully applied to study many of the fundamental aspects of plasticity of materials at the microscale [1–3]. In DD simulations, the equations of motion for dislocations are numerically solved to determine the evolution and interaction of dislocations. Microstructure details are a direct observation in the simulation and can be used to validate experimental results.

* Corresponding author. Tel.: +1 505 606 0069.
*E-mail address:* zhiqiang@lanl.gov (Z. Wang).

Many DD models exist and most of them use straight segments to connect discrete dislocation nodes on a dislocation loop [1,4,5]. This leads to singularities when the self-force is calculated at intersection of two segments, because the self-force is a function of the curvature of the dislocation line. Fine meshing of the dislocation line with many segments is required specially for strong interactions, which demands additional expensive computations. To eliminate these limits, the parametric dislocation dynamics (PDD) model has been developed [6,7].

In the PDD, dislocations are represented as spatial curves connected through dislocation nodes. Fig. 1 shows a curved dislocation loop divided into segments, and a specific segment of the curve between two nodes. A third order shape function for the segment helps maintain $C^2$ continuity of dislocation lines at dislocation nodes and remove the singularity that exists in dislocation models with linear segments. Given a parameter $w$ from 0 to 1 between two nodes, let the position vectors at two nodes ($w = 0$ and $w = 1$) be $\mathbf{P}(0)$, $\mathbf{P}(1)$, and the tangent vectors at two nodes be $\mathbf{T}(0)$, $\mathbf{T}(1)$, then the position and tangent vectors for any point on the dislocation segment can be written as:

$$\mathbf{P}(w) = \sum_{i=1}^{4} \mathscr{C}(w)_i q_i; \quad \mathbf{T}(w) = \sum_{i=1}^{4} \mathscr{C}'(w)_i q_i, \tag{1}$$

where $\mathscr{C}_i$ are the shape functions for the cubic spline, and $q_i$ are generalized coordinates as follows:

$$q_1 = \mathbf{P}(0); \quad q_2 = \mathbf{T}(0); \quad q_3 = \mathbf{P}(1); \quad q_4 = \mathbf{T}(1). \tag{2}$$

We have shown that with this parametric description, the displacement and stress fields from an infinitesimal dislocation segment can be written in compact vector forms as [6]:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}w} &= \frac{|\mathbf{b}||\mathbf{T}|V}{8\pi(1-\nu)R}\left(\frac{(1-\nu)V_1/V}{1+\mathbf{s}\cdot\mathbf{g}_1} + \left[(1-2\nu)\mathbf{g}^1 + \mathbf{g}_1\right]\right), \\ \frac{\mathrm{d}\sigma}{\mathrm{d}w} &= \frac{\mu V|\mathbf{T}|}{4\pi(1-\nu)R^2}\left[(\mathbf{g}^1\otimes\mathbf{g}_1 + \mathbf{g}_1\otimes\mathbf{g}^1) + (1-\nu)(\mathbf{g}^2\otimes\mathbf{g}_2 + \mathbf{g}_2\otimes\mathbf{g}^2) - (3\mathbf{g}_1\otimes\mathbf{g}_1 + \mathbf{I})\right], \end{aligned} \tag{3}$$

where

$$\mathbf{g}_1 = \frac{\mathbf{P}(w)}{|\mathbf{P}(w)|}; \quad \mathbf{g}_2 = \frac{\mathbf{T}(w)}{|\mathbf{T}(w)|}; \quad \mathbf{g}_3 = \frac{\mathbf{b}}{|\mathbf{b}|}, \tag{4}$$

and

$$\mathbf{g}^i \cdot \mathbf{g}_i = \delta^i_j, \tag{5}$$

$\mathbf{b}$ is the Burgers vector for a dislocation loop, $\delta^i_j$ is the Kronecker delta, $V = (\mathbf{g}_1 \times \mathbf{g}_2)\cdot\mathbf{g}_3$, $V_1 = (\mathbf{s}\times\mathbf{g}_1)\cdot\mathbf{g}_2$ and $\mathbf{s}$ is an arbitrary unit vector.

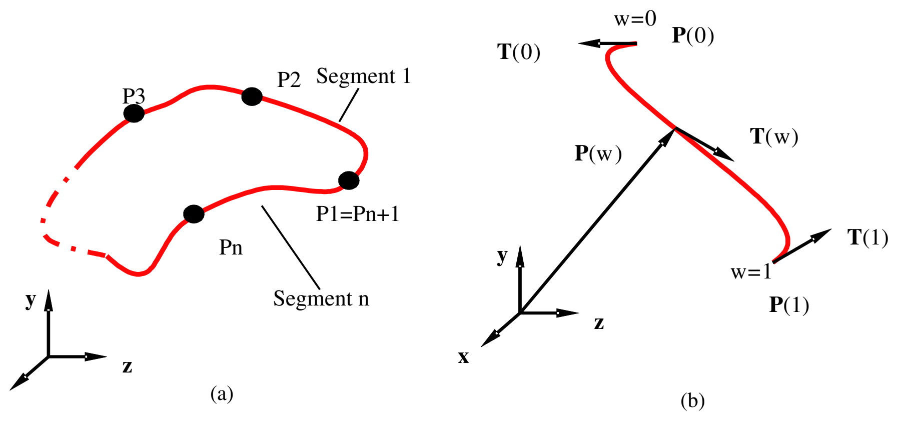

Fig. 1. Parametric representation of dislocation lines. (a) A dislocation loop is divided into segments connected at dislocation nodes; (b) a curved dislocation segment between two nodes.

Details of the derivations of the equation of motion (EOM) for a generalized loop can be found in [7]. Here, we give a brief review of the process to familiarize readers. The EOM is based on a variational principle for Gibbs free energy, derived from irreversible thermodynamics as:

$$\oint_{\Gamma} (f_k^{\mathrm{t}} - B_{\alpha k} V_\alpha)\delta r_k |\mathbf{ds}| = 0, \tag{6}$$

where $f_k^{\mathrm{t}} = f_k^{\mathrm{pk}} + f_k^{\mathrm{s}} + f_k^{\mathrm{o}}$ is the component $k$ of the total force acting on a dislocation from the summation of $f_k^{\mathrm{pk}}$, the Peach–Koehler force, $f_k^{\mathrm{s}}$, the self-force, and $f_k^{\mathrm{o}}$, the Osmotic force [7]. $B_{\alpha k}$ is the resistive matrix, and $V_\alpha$ is the velocity of the dislocation and $r_k$ is the position of a point on the dislocation.

By dividing dislocation loops into $N_s$ segments, Eq. (6) can be written as:

$$\sum_{j=1}^{N_s} \int_j \delta r_i (f_i^{\mathrm{t}} - B_{ik} V_k) |\mathbf{ds}| = 0. \tag{7}$$

From Eq. (1), we have:

$$\begin{aligned}
r_i &= \sum_{j=1}^{N_{\mathrm{DF}}} \mathscr{C}_{im}(w) q_m, \\
\delta r_i &= \sum_{j=1}^{N_{\mathrm{DF}}} \mathscr{C}_{im}(w) \delta q_m, \\
V_k = r_{k,t} &= \sum_{n=1}^{N_{\mathrm{DF}}} \mathscr{C}_{kn} q_{n,t}, \\
|\mathbf{ds}| = (r_{l,w} r_{l,w})^{1/2}\, \mathrm{d}w &= \left( \sum_{p,s=1}^{N_{\mathrm{DF}}} q_p \mathscr{C}_{lp,w} \mathscr{C}_{ls,w} q_s^{1/2} \right) \mathrm{d}w
\end{aligned} \tag{8}$$

with $q$ as the local coordinates of nodes on each segment. Plugging equation (8) back to Eq. (7), we have

$$\sum_{j=1}^{N_s} \int_0^1 \sum_{m=1}^{N_{\mathrm{DF}}} \delta q_m \mathscr{C}_{im}(w) \left[ f_i^{\mathrm{t}} - B_{ik} \sum_{n=1}^{N_{\mathrm{DF}}} \mathscr{C}_{kn} q_{n,t} \right] \times \left( \sum_{p,s=1}^{N_{\mathrm{DF}}} q_p \mathscr{C}_{lp,w} \mathscr{C}_{ls,w} q_s \right)^{1/2} \mathrm{d}w = 0. \tag{9}$$

To reduce the complexity of this equation, we define two terms, an effective force $f_m$ and an effective resistivity $\gamma_{mn}$, for each segment as:

$$\begin{aligned}
f_m &= \int_0^1 f_i^{\mathrm{t}} \mathscr{C}_{im}(w) \left( \sum_{p,s=1}^{N_{\mathrm{DF}}} q_p \mathscr{C}_{lp,w} \mathscr{C}_{ls,w} q_s \right)^{1/2} \mathrm{d}w, \\
\gamma_{mn} &= \int_0^1 \mathscr{C}_{im}(w) B_{ik} \mathscr{C}_{kn}(w) \left( \sum_{p,s=1}^{N_{\mathrm{DF}}} q_p \mathscr{C}_{lp,w} \mathscr{C}_{ls,w} q_s \right)^{1/2} \mathrm{d}w.
\end{aligned} \tag{10}$$

As a result, Eq. (9) is reduced to

$$\sum_{j=1}^{N_s} \left[ \sum_{m=1}^{N_{\mathrm{DF}}} \delta q_m \left( f_m - \sum_{n=1}^{N_{\mathrm{DF}}} \gamma_{mn} q_{n,t} \right) \right] = 0. \tag{11}$$

If we write

$$\begin{aligned}
\sum_{j=1}^{N_s} \sum_{m=1}^{N_{\mathrm{DF}}} \sum_{n=1}^{N_{\mathrm{DF}}} [\delta q_m \gamma_{mn} q_{n,t}] &= \sum_{k=1}^{N_{\mathrm{tot}}} \sum_{l=1}^{N_{\mathrm{tot}}} \delta \mathscr{Q}_k \Gamma_{kl} \mathscr{Q}_{l,t}, \\
\sum_{j=1}^{N_s} \sum_{m=1}^{N_{\mathrm{DF}}} [\delta q_m f_m] &= \sum_{k=1}^{N_{\mathrm{tot}}} \delta \mathscr{Q}_k \mathscr{F}_k,
\end{aligned} \tag{12}$$

where $Q_k$ are global coordinates of degrees of freedom of segments, which may be the position or tangent at a specific node. $[\Gamma_{kl}]$ is the global resistivity matrix, $\mathscr{F}_k$ is the global force vector, and $N_{\mathrm{tot}}$ is the total number of degrees of freedom. Realizing that the virtual displacements are totally arbitrary, we have the final equation of motion in form as

$$\mathscr{F}_k = \sum_{l=1}^{N_{\text{tot}}} \Gamma_{kl} \mathscr{Q}_{l,t}. \tag{13}$$

After solving Eq. (13), numerical integration methods are applied to update the generalized coordinates and hence the dislocation shape. Thus, microstructure evolutions can be simulated.

In our PDD simulations, to accommodate plastic deformation, the dislocation density increases through the multiplication of Frank–Read sources. These are dislocations pinned at the two ends. Details of the PDD model and applications can be found in Refs. [2,7,6].

PDD and other DD simulations provide a promising way to link the microstructure and the macroscopic properties of materials. One of the interesting applications is to use DD to explain the strain hardening of materials. Although progress has been made by PDD and general DD methods to explain the nature of plastic deformation [1,8–10], there are several challenges that limit their extensive utilization. First, in order to get a representative macroscale plastic behavior, the collective behavior of a large dislocation system must be considered. The typical size scales are microns which may contain tens of thousands of dislocations. This presents a huge demand for computational power that cannot be fulfilled by a single processor. Parallel simulation is naturally selected as an alternative approach, in which a large problem is divided into small pieces and solved by different individual processors.

Dislocations have long-range elastic interactions such that all the dislocation neighbors have to be taken into account for interaction calculation. This makes the computational burden scale as $O(N^2)$, which becomes prohibitive for large $N$. Employing a "cut-off distance" in the simulation with long-range force fields is known to produce spurious results [11,12]. Much effort has been put into developing algorithms that reduce that burden. Many of the more important methods, developed independently by a number of groups, are based on the concept of a *hierarchical tree*. While different methods of constructing tree structures and handling of interaction potentials have been used, all of them share two important common characteristics. First, they utilize a hierarchical-tree data structure, and second, they directly compute the force on an individual particle from nearby particles while the force from remote particles is calculated by an approximate method. Among these methods are the Barnes–Hut (BH) method [13] and fastmultipole method (FMM) [14]. The BH technique builds its data structure via a hierarchical subdivision of space into cubic cells (for 3-D problems) and an oct tree (quad tree for 2-D) is built. The subdivision of volume is repeated until the cells at the lowest level of the tree contain at most one or no particles. Each node of the tree in BH method represents a physical volume of space and the total mass and the center-of-mass within the volume is stored at the node. The BH method has a computational burden of $O(N \ln N)$. Far-field interactions are usually calculated with a low-order multipole expansion, though often just the zeroth-order "charge–charge" term is employed. The FMM method is based on the same oct-tree structure as the BH method, though with no restriction on having just one (or no) particles in the leaf nodes. By a clever management of the interactions, however, the FMM reduces the computational effort to $O(N)$. In addition to its computational efficiency, the error in the FMM can be reduced to machine accuracy by keeping enough terms in the multipole expansions. A multipole acceptance criteria (MAC) is employed to determine what kind of interaction (direct or approximate) should be included for each particle. Parallel formulations for both the BH and FMM methods have been developed [15–19].

It is possible to apply similar hierarchical strategy to DD simulations. However, another important challenge for DD simulations is that dislocations are curvilinear defects that run through a lattice. This makes DD simulation more complicated than particle problems. While we can divide space into subdivisions, the connectivity between dislocation segments have to be maintained if the dislocation is across the boundaries of these subdivisions. In the meantime, dislocation systems are heterogeneous, i.e., there are regions of dense dislocation density and regions that are essentially empty of dislocations. This makes it difficult to divide the system into sub-systems with similar problem sizes for parallel simulations to maintain the load balancing for each working processor.

Because of these challenges, very few parallel DD simulations have been implemented. One implementation in Lawrence Livermore National Lab has been reported [20]. In their implementation, dislocations are considered as nodes connected through straight line segments. Besides the difference between the algorithm used in their implementation and the reported algorithm here, one main difference is that in their implementation, one node may have more than two arms due to the annihilation of two or more segments. This may create very complex microstructures that could be artificial according to their topological rules. In our model, each dislocation retain its identity even in strong interactions and its topological configuration is fully controlled by the force acting on the dislocation without defined rules, which is more physically based.

In this paper, we wish to present a parallel PDD implementation in order to expand the capabilities of PDD simulations for a wide range of physical problems. First, the PDD model is extended to a particle formalism, which transfers the line problem of dislocations to a particle like problem. Thus, similar hierarchical strategies for particles can be applied to PDD simulations without much modification. This reduces the complexity in treating line defects across computational domain borders. Then, details of the algorithm implementation, especially sub-tree structures defined to form a hierarchical tree in the parallel PDD, are explained. The process to construct the hierarchical tree is a dynamical one that can accommodate the heterogeneous dislocation distribution, and it is explained in detail. We also develop procedures to reduce the storage and communication burdens. It will also be shown that long range dislocation interactions can be incorporated by natural combinations of the tree data structure with the dislocation multipole expansion method (MEM) [21], which is similar to the well known Greengard–Rokhlin method [14] and leading to achieve O($N$) computation. Finally, the performance of the implementation is illustrated by considering a large scale application of strain hardening of single crystal copper.

## 2. Dislocation particles

In this section, a mathematical model is developed such that dislocation calculations can be made similar to particle simulations by representing dislocations as a set of dislocation particles, which are then treated as particles.

We have known that in Eq. (13), the global resistivity matrix $\Gamma_{kl}$ and global force matrix $\mathscr{F}_k$ are assembled from local effective resistivity $\gamma_{mn}$ and effective force $f_m$ of each segment. The procedure is just like the assembling of the global stiffness matrix in standard finite element method (FEM). With careful study, it is found that the global resistivity matrix $\Gamma_{kl}$ is banded and sparse. The only none-zero elements are the elements with unequal indices of $k$ and $l$. Here the EOM can be written in a different matrix form as:

$$\begin{bmatrix} * & * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & * & 0 & 0 & 0 \\ 0 & 0 & S_{i-1} & S_i & S_{i+1} & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * & * \end{bmatrix} \begin{bmatrix} * \\ Q_{i-1,t} \\ Q_{i,t} \\ Q_{i+1,t} \\ * \end{bmatrix} = \begin{bmatrix} * \\ F_{i-1} \\ F_i \\ F_{i+1} \\ * \end{bmatrix}, \tag{14}$$

where $S_i$ and $F_i$ are resistivity and force elements arranged from the global resistivity matrix and global force vector for a dislocation node $i$, respectively. They can be calculated for each node with only the information of the direct neighbor segments of that node. The generalized coordinates of each node (the dislocation particle) are given by $Q_i$. Eq. (14) is solved at each time step. However, instead of constructing and solving the global vector $\mathscr{F}_k$ and global matrix $\Gamma_{kl}$ for one dislocation loop, we can calculate individual $S_i$ and $F_i$ for each dislocation node based on its connecting dislocation segments and solve for each individual node using the Gauss–Jacobi or Gauss–Seidel iteration method [22]. We give these individual node the name of dislocation particle. The solution for a dislocation particle $i$ at time $t + \delta t$, where $t$ is the current time and $\delta t$ the time step, can be obtained through a $p$ step iteration:

$$S_i Q^p_{i,t+\delta t} = F_i - (S_{i-1} Q^{p-1}_{i-1,t} + S_{i+1} Q^{p-1}_{i+1,t}), \tag{15}$$

where $p$ is the iteration step, and $Q^0_{i,t} = Q^{(t)}_{i,t}$ with $Q^{(t)}_{i,t}$ the solution at time $t$. This equation can be solved independently for each dislocation particle ($i$) at time $t + \delta t$ with the assumption that the solution for its connected dislocation particles ($i - 1$ and $i + 1$) at time $t$ has been obtained.

To illustrate how $S$ and $F$ are calculated, we give a simple example. Suppose we have a dislocation Frank–Read source with four nodes on it, which forms three segments, we write the local resistivity for segment $j$ as $\gamma^j_{mn}$, and the effective force for segment $j$ as $f^j_m$, the EOM for this system is written as:

Table 1
Illustration for the calculation of resistivity and force elements for dislocation particles

| | $S_{i-1}$ | $S_i$ | $S_{i+1}$ | $F_i$ |
|---|---|---|---|---|
| $\mathcal{Q}_1$ | 0 | $\gamma^1_{11}$ | $\gamma^1_{12}$ | $f^1_1$ |
| $\mathcal{Q}_2$ | $\gamma^1_{21}$ | $\gamma^1_{22}+\gamma^2_{11}$ | $\gamma^2_{12}$ | $f^1_2+f^2_1$ |
| $\mathcal{Q}_3$ | $\gamma^2_{21}$ | $\gamma^2_{22}+\gamma^3_{11}$ | $\gamma^3_{12}$ | $f^2_2+f^3_1$ |
| $\mathcal{Q}_4$ | $\gamma^3_{21}$ | $\gamma^3_{22}$ | 0 | $f^2_3$ |

$$\begin{bmatrix} \gamma^1_{11} & \gamma^1_{12} & 0 & 0 \\ \gamma^1_{21} & \gamma^1_{22}+\gamma^2_{11} & \gamma^2_{12} & 0 \\ 0 & \gamma^2_{21} & \gamma^2_{22}+\gamma^3_{11} & \gamma^3_{12} \\ 0 & 0 & \gamma^3_{21} & \gamma^3_{22} \end{bmatrix} \begin{bmatrix} \mathcal{Q}_{1,t} \\ \mathcal{Q}_{2,t} \\ \mathcal{Q}_{3,t} \\ \mathcal{Q}_{1,t} \end{bmatrix} = \begin{bmatrix} f^1_1 \\ f^1_2+f^2_1 \\ f^2_2+f^3_1 \\ f^2_3 \end{bmatrix}. \tag{16}$$

Comparing with Eq. (14), we have $S_{i-1}$, $S_i$, and $S_{i+1}$ for $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_3$, and $\mathcal{Q}_4$ as shown in Table 1.

Eq. (15) is now of the same form as that for the EOM of a system of interacting particles, where each dislocation node $i$ can be considered as a virtual particle, with the attributes identifying the information of its position, tangent, connection points on the dislocation loop, the glide plane and Burgers vector of the loop. The dislocation problem becomes a particle problem with the special consideration of connection between dislocation particles. Thus, the implementation of the parallelization will be essentially the same as for systems of point particles.

## 3. DD code parallelization

The primary objective of DD parallelization is to divide the computational domain, which is the physical volume containing the dislocation systems, into different sub-domains, and to solve the EOM (Eq. (15)) of dislocation particles (henceforth called DPs) in each sub-domain independently on a single processor. In this way, the amount of work for each processor is reduced and the speed of calculation is improved. The sub-domains are not required the have the physical same size or shape. The only two requirements are that the DPs in each sub-domain should be as close as possible and each sub-domain has a similar number of DPs. The first criterion ensures that the closest neighbors of most dislocation DPs reside on the same processor such that they are not transferred from a different processor when they are needed. In this way, communication is kept at the lowest level. The latter criterion ensures that all processors have a similar amount of work such that load balancing is realized for best performance.

In the parallel implementation, we handle all communications between processors with the standard message passing interface (MPI) [23]. We also assume a master–slave computing structure, in which one computer processor is defined as master, to handle I/O, data management, etc., while the others (slaves) perform the main calculations.

Since we can now take advantage of previously-developed parallel algorithms for heterogeneous systems for our model of dislocation DPs, we have chosen to do this using a binary space partitioning scheme. To facilitate the distribution, we map the computational domain and DPs on to a tree structure that has three parts on each processor: a global part, a local part and a ghost part. The partition process of the computational domain is done along with the building up of the tree structure. We now describe the tree structure and its creation in detail.

The global tree is built by the master processor and is distributed to each slave processor, i.e., each processor has an identical global tree. The local tree is built on one processor based on the local group of DPs that are assigned to the processor. The average number of DPs on the processor is $N_{\mathrm{ave}} = N_{\mathrm{total}}/N_{\mathrm{p}}$, where $N_{\mathrm{total}}$, $N_{\mathrm{p}}$ are the total number of DPs and the number of processors, respectively. The real number of DPs on each processor is made close to $N_{\mathrm{ave}}$ for load balancing. The ghost tree on one processor is transferred from other processors, which has close neighbor information. This architecture helps to increase the speed of the master processor by requiring that it builds only the global tree. It also reduces the use of memory on each processor without building a tree for the entire set of DPs.

### 3.1. The global tree

The process to build a global tree is the main procedure to partition the domain and to preserve locality. This procedure is done as follows and is shown schematically in Fig. 2. Randomly selected DPs are assigned to a single level tree, which represents the entire domain. Since we allow at most $N_{\mathrm{ave}}$ DPs per leaf of the global tree, we need to split the domain into two when the $N_{\mathrm{ave}} + 1$ DPs have been assigned to the domain. We do this by splitting the domain along the distance of largest separation between two possible particle groups, which ensures that most of the close DPs are together and not separated into different groups. The two newly created sub-domains are then made children of the tree level that contained the original DPs. We then pick more DPs and assign them to a sub-domain. If the number of DPs in that sub-domain is larger than $N_{\mathrm{ave}}$, that sub-domain is then split into two along the direction of maximum separation and two leaves are assigned to the tree level where the split took place. We continue this process until the domain has been partitioned into a binary tree with a number of $N_{\mathrm{p}}$ leaves. The top level of the tree represents the entire domain, and every level of the tree is made up exactly of all the sub-domains under it. In the mean time, each node of the tree contains information on the center of the domain/sub-domain, and the size of this domain/sub-domain. In addition, each of the lowest level leaves contain the attributes, specified in Section 2, for $N_{\mathrm{ave}}$ DPs. Thus, all the information of the whole domain and its partitions has been fully recorded by the global tree data structure. Finally, each leaf of the global tree and the group of DPs attached to this leaf are assigned to one of the $N_{\mathrm{p}}$ processors.

### 3.2. The local and ghost trees

The local tree is the second part of the hierarchical tree. After a group of DPs have been assigned to it, each processor performs a similar process to build a binary tree for these local DPs, assuming that each leaf of the local tree has only one DP. Next, the completed local tree is attached to a leaf of the global tree, knowing that the global tree leaf is assigned to this processor. Each processor is responsible for calculating forces and motion of the DPs resident in the local tree. At this step, all DPs in the leaves of the global tree have been passed down to the leaves of the local tree.

Although most of the closest neighbors of local DPs have been preserved on the same processor due to the nature of this partitioning algorithm in building the global tree, a small fraction of the neighbors reside on other processors and they are required to be transferred to the local processor. Thus, and in general, parts

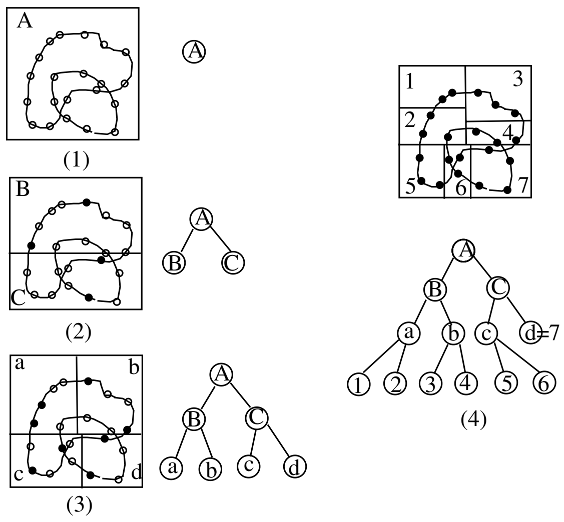

Fig. 2. The dynamic recursive domain partitioning process along with the corresponding procedure to build the global tree. Each node of the tree represents a specific domain or sub-domain. Filled circles represent dislocation particles that are randomly picked and included in the domain.

of the local trees on other processors are transferred. These are called as ghost trees. These ghost trees are attached to the global leaves corresponding to the processors where they come from. To satisfy the criterion for transfer as ghost to another processor, a local tree node has to be physically close to the sub-domain that is assigned to that processor. The critical value used to measure the distance between the two is 2–3 times the cut-off distance, $L_C$, for direct interaction. With this specification, the size of ghost information does not change much with the overall system size because it is restricted to a certain distance. Thus, the amount of communications does not increase dramatically with the system size.

In summary, the local tree contains the information for the "on-processor" DPs, i.e., dislocation particles for which information is available without any parallel communication. The ghost tree is the "off-processor" data or information needed to calculate forces for the on-processor DPs that reside on other processors, and has to be gathered. Once the above three parts of the tree are constructed and combined together, each processor has an essential tree, as shown in Fig. 3, which has all necessary information.

### 3.3. Dislocation neighbor search and interactions

Close dislocation interactions are calculated according to the stress fields in Eq. (3). Long range dislocation interactions can be calculated by using the multipole expansion method (MEM) [21]. In the MEM, the stress field resulting from a dislocation ensemble within a volume $\Omega$, which is a distance away from the interacting dislocation, can be written as:

$$\sigma_{ij} = \frac{\mu\Omega}{8\pi}\sum_{t=0}^{\infty}\frac{1}{t!}\left[R^{\text{o}}_{,mppa_1\ldots a_t}\left(\epsilon_{jmn}\langle\zeta_{nia_1\ldots a_t}\rangle + \epsilon_{imn}\langle\zeta_{nja_1\ldots a_t}\rangle\right) + \frac{2}{1-\nu}\epsilon_{kmn}R^{\text{o}}_{,ijma_1\ldots a_t}\langle\zeta_{nka_1\ldots a_t}\rangle \right.$$
$$\left. - \frac{2}{1-\nu}\delta_{ij}\epsilon_{kmn}R^{\text{o}}_{,ppma_1\ldots a_t}\langle\zeta_{nka_1\ldots a_t}\rangle\right], \qquad (17)$$

where $\mathbf{R}^{\text{o}}$ is a vector connecting the field point where the stress is evaluated and the center of the volume, $\langle\zeta_{ijk\ldots}\rangle$ represent the dislocation moments defined in Ref. [21]. These moments depend only on the distribution of the dislocation microstructure within the volume. They can be evaluated for each volume independently. After the moments are determined, the stress field and interaction forces on other dislocations that are sufficiently well separated from the volume $\Omega$ are easily obtained.

The MEM can be easily matched to the tree structure, because each node of the tree structure represents a volume containing some distribution of dislocations, and the moments for the dislocation distribution of the volume can be easily calculated after the local tree has been built. Thus, in the following calculations, long range interactions between dislocations can be computed from these moments.

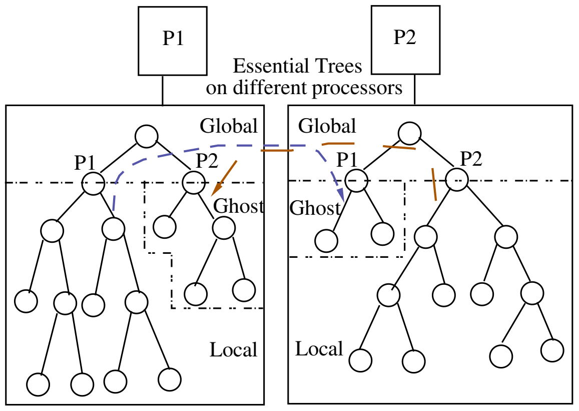

Fig. 3. The essential tree structure on each processor. Each essential tree is a combination of three sub-tree structures. The global tree keeps processor-level domain decomposition information, the ghost tree keeps information transferred from other processors, and the local tree keeps information of local dislocation particles.

After the tree structure has been built on each processor, dislocation neighbors are searched by the tree traversal process based on the close neighbor criterion (CNC), which is described as follows. If any two dislocation segments represented by dislocation particles are within a critical distance ($L_C$), these two segments are direct neighbors and their interaction is calculated directly, otherwise the interaction is calculated by using the MEM. Based on detailed analysis of dislocation interactions, the value of $L_C$ is chosen to be around $100a$ [24], where $a$ is the lattice constant. The tree traversal process is recursive, and for each dislocation particle, the process always begins from the root tree node. The distance between the dislocation particle under consideration and the limits of the domain represented by the tree node is first calculated. If this distance is less than $L_C$, then we descend one more level into the tree. If we descend to a leaf before we are more than $L_C$ away, then we need to calculate the interaction directly, and we add that node to the list of nodes that have to be calculated directly. If the distance is greater than $L_C$, the interaction between the current dislocation particle and the entire subtree below that level is calculated using a multipole expansion of the force term. We then add the multipole moments of dislocations in the entire subtree below that level to the list of subtrees for multipole expansion calculations. After the traversal process, dislocation neighbors are listed for each dislocation particle and become available for further calculations.

### *3.4. Updating tree information*

Once the tree structure is completed and dislocation neighbors are identified, each processor can begin to solve the equations of motion. Dislocations then move and their distributions in space changes. Hence, the tree structure needs to be rebuilt and the dislocation neighbor-lists need to be reconstructed. In this process, load balancing is also maintained. Although building the global tree can be done in O($N$) operations (because it creates $2N-1$ tree nodes which is on the order of $N$), traversing the tree to build neighbor lists can be done in O($N\log(N)$) operations. It is not necessary to communicate information every time step to rebuild the tree. Also because we have chosen a larger distance than $L_C$ in determining ghost information, we have enough information to perform calculations for a number of time steps before the neighbor lists are updated. We choose the number of time steps for updating information such that dislocations do not move out the ghost zone range. According to dislocation mobility, stress and time steps in the simulation, a rough estimation for this number is around 50–300.

## 4. Performance and simulation results

A parallel computer code was constructed on these principles, and named DeMecha, for general simulations of defect mechanics. The parallel code DeMecha was tested on the UCLA ISIS Beowulf cluster, which has 90 dual processor AMD Opteron, 246 2.0 GHz 32-Bit processors. In the first test case, up to 60 computer processors were used for a number of 600 simulated curved dislocations in the simulation. Each dislocation has 10 DPs, and each DP has 6 degrees of freedom (DOF) (3 for position and 3 for the tangent). The total number of degrees of freedom is 36,000. Let us define a speedup factor $S=\frac{t_1}{t_N}$, where $t_1$ and $t_N$ are the computation times for 1 and $N$ processors, respectively. Fig. 4 shows that the speedup factor increases sublinearly with the number of nodes, and is about 44 when 60 nodes are utilized.

To determine the communication efficiency of the parallel DD code, each processor was assigned the same number of degrees of freedom. In the second and third simulations, 150 and 300 degrees of freedom were used. The number of processors was then increased, and the communication efficiency was determined. The "communication efficiency, $\eta_C$" is defined as the ratio of the speedup factor to the total number of processors used, i.e. $\eta_C = S/N$. Because the computational overhead associated with processor-to-processor communication increases with the number of processors, $\eta_C$ is expected to decrease. This is seen in Fig. 5. However, the figure also shows that $\eta_C$ remains above 85%, irrespective of the number of degrees of freedom per processor.

A large scale simulation was performed on a total of 60 processors for a uniaxial tension test of fcc single crystal copper to predict the stress–strain curve of copper. The external load is applied along the [1 0 0]-direction at a constant strain rate of $c = 100\ \mathrm{s}^{-1}$. The initial dislocation density is $\rho = 1\times10^{7}\ \mathrm{cm/cm}^3$. These initial dislocations are randomly distributed Frank–Read sources. The simulation volume of material is $10\ \mu\mathrm{m}\times 10\ \mu\mathrm{m}\times 10\ \mu\mathrm{m}$, and periodic boundary conditions are applied. In the simulation, the material constants used

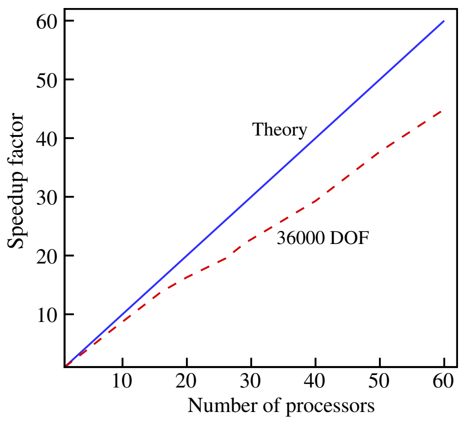

Fig. 4. The dependence of the speedup factor on the number of Beowulf cluster processors for the UCLA-microplasticity DD code.

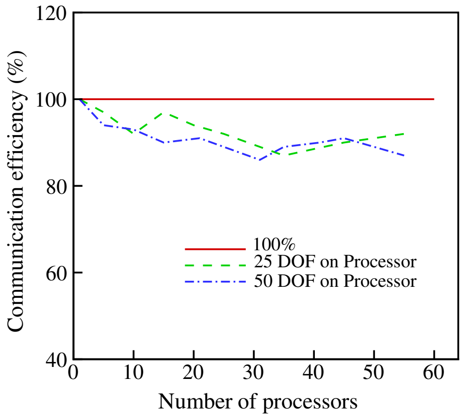

Fig. 5. Dependence of the communication efficiency of the UCLA-microplasticity parallel code on the number of processors. The number of degree of freedom on each processor is fixed.

for copper are: dislocation mobility $M = 10^4$ Pa$^{-1}$ s$^{-1}$, shear modulus $\mu = 50$ GPa and Possion ratio $v = 0.31$. The time step used is $\delta t = 10^{-12}$ s.

In order to obtain the measured stress in the simulation, we notice that

$$c = \dot{\epsilon}_{11} = \dot{\epsilon}^{\mathrm{e}}_{11} + \dot{\epsilon}^{\mathrm{p}}_{11}, \tag{18}$$

where $\epsilon^{\mathrm{e}}$ and $\epsilon^{\mathrm{p}}$ are elastic and plastic strain rate, respectively. The plastic strain rate is calculated from dislocation motions as:

$$\dot{\epsilon}^{\mathrm{p}} = -\frac{1}{2V}\sum_{i=1}^{N}\oint_{0}^{l^{(i)}} v^{(i)}[\mathbf{n}^{(i)}\otimes\mathbf{b}^{(i)} + \mathbf{b}^{(i)}\otimes\mathbf{n}^{(i)}]\,\mathrm{d}l, \tag{19}$$

where $N$ is total number of dislocation loops, $l^{(i)}$ is the length of dislocation $i$, $V$ is the volume of the simulated material, $\mathbf{n}$ is a unit vector defined as $\mathbf{v}\times\xi$, $\mathbf{v}$ and $\xi$ are the velocity and the tangent vectors of the dislocation loop at a point on the dislocation line, respectively. The elastic strain rate in the [1 0 0] direction is defined as:

$$\dot{\epsilon}^{\mathrm{e}}_{11} = \frac{\dot{\sigma}_{11}}{E}, \tag{20}$$

where $E$ is Young's modules and $\boldsymbol{\sigma}$ the stress tensor. With the definition $\dot{\sigma}_{11} = \frac{\sigma^{t+1}_{11} - \sigma^{t}_{11}}{\delta t}$, Eqs. (18)–(20) lead to

$$\sigma^{t+1}_{11} = \sigma^{t}_{11} + E\delta t(c - \dot{\epsilon}^{\mathrm{p}}_{11}). \tag{21}$$

Obtaining the stress from Eq. (21) and the strain as $\varepsilon = c\delta t$.

The simulated stress–strain curves are shown in Fig. 6(a). The results of the simulations clearly show that elastic deformation at lower stresses are followed by extensive plastic deformation. The hardening rate for the

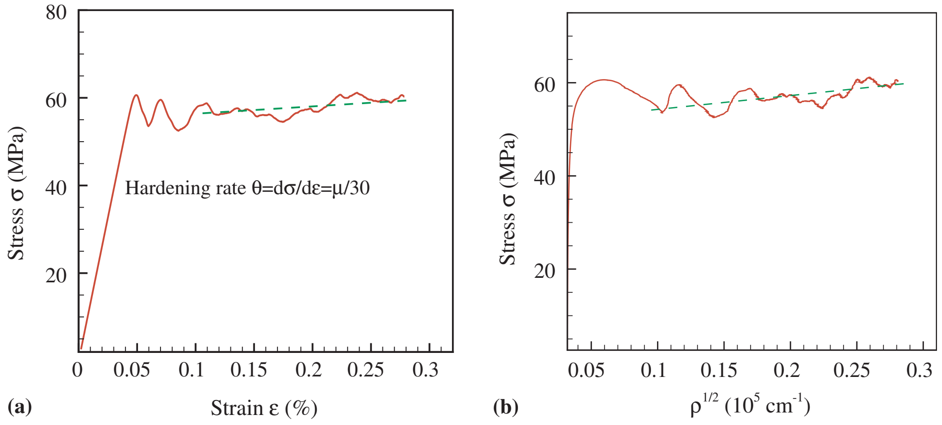

Fig. 6. (a) Predicted stress–strain relationship for single crystal copper, and (b) the relationship between the dislocation density and applied stress for single crystal copper.

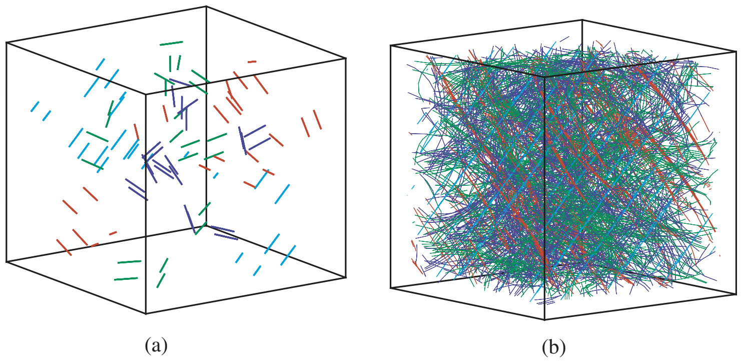

Fig. 7. Simulation of plastic deformation of a 10 μm × 10 μm × 10 μm volume. (a) Initial microstructure of dislocation Frank–Read sources, (b) microstructure at 0.3% strain. Different colors in the figure are used to represent dislocations on different glide planes, green = (1 1 1), cyon = $(1\,\bar{1}\,1)$, blue = $(\bar{1}\,\bar{1}\,1)$, red = $(\bar{1}\,1\,1)$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

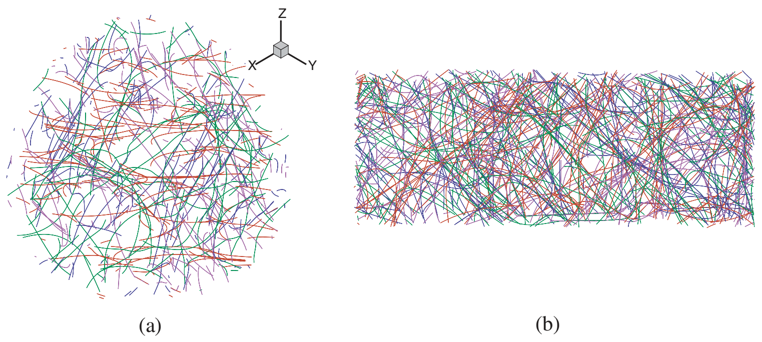

Fig. 8. [1 1 1] A slice of 1.5 μm, cut through the center of the simulation volume, showing the formation of complex 3D microstructure: (a) view along the [1 1 1]-direction, (b) view perpendicular to the [1 1 1]-direction.

plastic deformation $\theta = \frac{d\sigma}{d\epsilon}$ is found to be $\approx \frac{\mu}{30}$, where $\mu$ is the shear modulus of copper. This indicates that the crystal is in the beginning of the well-known stage II work hardening regime [25]. An approximate linear relationship between the stress and the square root of the dislocation density is shown in Fig. 6(b), once an initial transient is surpassed. The simulated dislocation microstructure is shown in Figs. 7–10. These microstructures are clearly complex, and are indicative of the collective motion of dislocation ensembles. Dislocation patterns such as dislocation-rich and dislocation-poor regions shown in Fig. 8 may indicate the incipient formation of dislocation cell structures. In Fig. 9, slip-band formation is shown, which is similar to experimental observations. Short-range interactions of dipoles and junctions are shown in Fig. 10, resulting in spatial heterogeneities of dislocation distributions. This may be used to explain microstructure evolution and pattern formation, as well as to prove the capability of the code in simulating large-scale collective motion of dislocations.

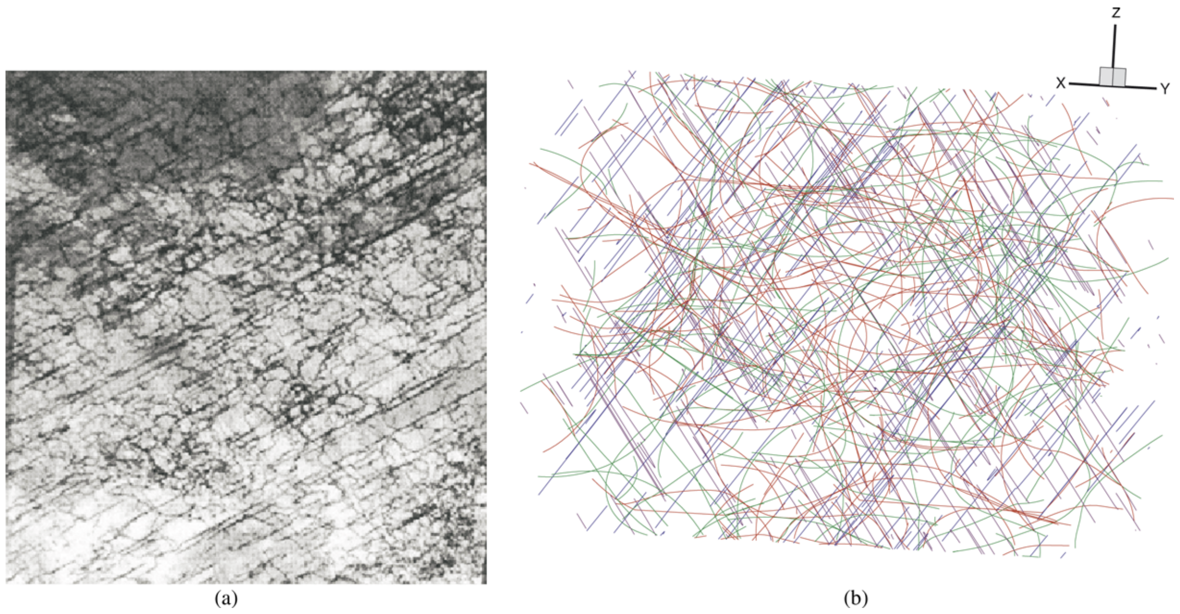

Fig. 9. Comparison between experimentally-observed and simulated dislocation microstructure. (a) Dislocation microstructure in Cu at a strain of $0.3 \times 10^{-3}$ (Riso National Lab, Denmark, 1998), (b) simulated results, view along the [1 1 0]-direction.

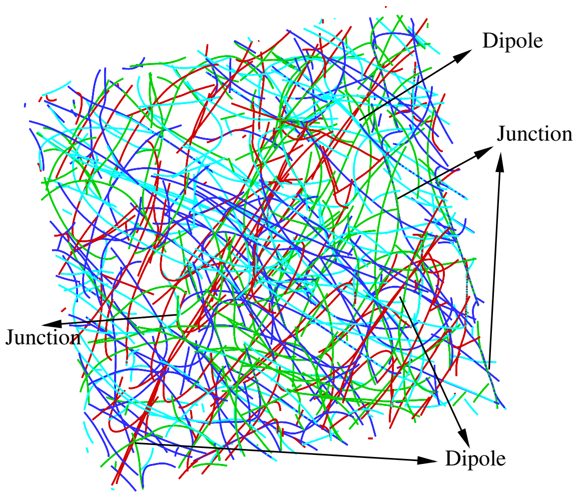

Fig. 10. Dislocation structures of typical short-range interactions (dipoles and junctions).

## 5. Conclusions

A parallel computer code DeMecha for large scale simulations of material deformation at the microscale is developed on the basis of discrete dislocation dynamics. Dislocations are represented as “dislocation particles”, which enables the utilization of particle interaction algorithms, while maintaining the continuous nature of dislocation lines as space curves. Thus, the complex problem of the dynamics of an ensemble of 3-D curved dislocation is transformed, for the first time, to the equivalent and well-established problem of particle dynamics with a long-range interaction field (e.g., charged particles, plasmas, quantum systems, gravitational systems, etc). A Hierarchical tree data structure has been designed to represent the computational domain and enable efficient partitioning and distribution of the computational load. The developed parallel simulation code, DeMecha, shows significant performance enhancements with speedup factors up to 44 for 60 parallel processors. Also, good control of communication and load balancing is achieved, with communication efficiencies in excess of 85%. The code was applied to simulate the elasto-plastic deformation of single crystal fcc copper subjected to uniaxial tension in Stage-II hardening regime. The results indicate that the collective behavior of dislocations is essential in explaining the experimentally observed microstructure and strain hardening curves. Moreover, the stress–strain relationship for single crystal copper in stage-II hardening is completely determined from simulations without any ad hoc assumption. The developed parallel algorithm should enable more realistic simulations of material deformation on the basis of microscopic physics.

## Acknowledgments

The work of R. LeSar and S. Swaminarayan was performed under the auspices of the United States Department of Energy (US DOE under contract W-7405-ENG-36) and was supported by the Division of Materials Science of the Office of Basic Energy Sciences of the Office of Science of the US DOE. The work of Z. Wang and N. Ghoniem was supported by Los Alamos National Laboratory under Contract # 54280-001-03 2F and by the US Department of Energy, Office of Fusion Energy, through Grant DE-FG02-03ER54708.

## References

[1] B. Devincre, L.P. Kubin, Simulations of forest interactions and strain hardening in fcc crystals, Modelling Simul. Mater. Sci. Eng. 2 (1994) 559.
[2] N.M. Ghoniem, L.Z. Sun, Fast sum method for the elastic field of 3-d dislocation ensembles, Phys. Rev. B 60 (1) (1999) 128.
[3] D. Weygand, L.H. Friedman, E. Van der Giessen, A. Needleman, Discrete dislocation modeling in three-dimensional confined volumes, Mater. Sci. Eng. A 309–310 (2001) 420.
[4] M. Rhee, J. Stolken, V. Bulatov, T. Rubia, H.M. Zbib, J. Hirth, Dislocation stress fields for dynamic codes using anisotropic elasticity: methodology and analysis, Mater. Sci. Eng. A 309–310 (2001) 288.
[5] K.W. Schwarz, Simulation of dislocations on the mesoscopic scale. i. Methods and examples, J. Appl. Phys. 85 (1) (1999) 108–119.
[6] N.M. Ghoniem, J. Huang, Z. Wang, Affine covariant–contravariant vector forms for the elastic field of parametric dislocations in isotropic crystals, Philos. Mag. Lett. 82 (2) (2002) 55–63.
[7] N.M. Ghoniem, S.H. Tong, L.Z. Sun, Parametric dislocation dynamics: a thermodynamics-based approach to investigations of mesoscopic plastic deformation, Phys. Rev. B 61 (2000) 913.
[8] V.V. Bulatov, H.M. Zbib, M.J. Tang, Crystal plasticity from dislocation dynamics, MRS Bull. 26 (2001) 1991.
[9] N.M. Ghoniem, J.M. Huang, Computer simulations of mesoscopic plastic deformation with differential geometric forms for the elastic field of parametric dislocations: review of recent progress, J. Phys. 11 (5) (2001) 53.
[10] N.M. Ghoniem, R. LeSar, A. Misra, Z. Wang, R.J. McCabe, T.E. Mitchell, Dislocation motion in thin cu foils: a comparison between computer simulations and experiment, Acta Mater. 52 (6) (2004) 1535.
[11] J.P. Hansen, in: G. Gicotti, W.G. Hoover (Eds.), Molecular Dynamics Simulations of Statistical–Mechanical Systems, North-Holland, Amsterdam, 1986, pp. 89–128.
[12] H.Y. Wang, R. Lesar, O(N) algorithm for dislocation dynamics, Philos. Mag. A 71 (1) (1995) 149.
[13] J. Barnes, P. Hut, A hierarchical o($N\log N$) force-calculation algorithm, Nature 324 (1986) 446.
[14] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (1987) 325–348.
[15] J.P. Singh, C. Holt, T. Totsuka, A. Gupta, J. Hennessy, Load balancing and data locality in adaptive hierarchical *n*-body methods: barnes-hut, fast multipole, and radiosity, J. Parallel Distrib. Comput. 27 (1995) 118.
[16] A.Y. Grama, V. Kumar, A. Sameh, Scalable parallel formulations of the barnes-hut method for *n*-body simulations, in: Proceedings Supercomputings, vol. 94, 1994, p. 439.

[17] M. Amor, F. Arguello, J. Lopez, O. Plata, E.L. Zapata, A data parallel formulation of the barnes-hut method for *n*-body simulations, in: Proceedings for Applied Parallel Computing, PARA 2000, 2000, p. 342.
[18] A.Y. Grama, V. Kumar, A. Sameh, *n*-Body simulations using message passing parallel computers, in: Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing, 1995, p. 355.
[19] J.-L. Lu, D.I. Okunbor, Parallel implementation of 3d fma using mpi, in: Proceedings of Second MPI Developer's Conference, 1996, p. 119.
[20] W. Cai, J. Fier, M. Hiratani, G. Hommes, T. Pierce, M. Tang, M. Rhee, V. Bulatov, T. Arsenlis, K. Yates, Scalable line dynamics in paradis, in: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, 2004, p. 19.
[21] Z.Q. Wang, N.M. Ghoniem, R. LeSar, Multipole representation of the elastic field of dislocation ensembles, Phys. Rev. B 69 (17) (2004) 174102–174107.
[22] C.E. Pearson, Numerical Methods in Engineering and Science, Van Nostrand Reinhold Company, New York, 1986.
[23] W. Gropp, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, MIT Press, New York, 1994.
[24] J.M. Huang, N.M. Ghoniem, The dynamics of dislocation interaction with sessile self-interstitial atom (SIA) defect cluster atmospheres, J. Comput. Mater. Sci. 23 (2002) 225.
[25] E. Nes, Modelling of work hardening and stress saturation in fcc metals, Prog. Mater. Sci. 41 (1998) 129–193.