Modelling Simul. Mater. Sci. Eng. **15** (2007) 553–595 doi:10.1088/0965-0393/15/6/001

Enabling strain hardening simulations with dislocation dynamics

A Arsenlis[1,3], W Cai[2], M Tang[1], M Rhee[1], T Oppelstrup[1], G Hommes[1], T G Pierce[1] and V V Bulatov[1]

[1] Lawrence Livermore National Laboratory, University of California, Livermore, CA 94551, USA
[2] Department of Mechanical Engineering, Stanford University, Stanford, CA 94305-4040, USA

E-mail: arsenlis@llnl.gov

Received 20 December 2006, in final form 5 June 2007
Published 25 July 2007
Online at stacks.iop.org/MSMSE/15/553

Abstract

Numerical algorithms for discrete dislocation dynamics simulations are investigated for the purpose of enabling strain hardening simulations of single crystals on massively parallel computers. The algorithms investigated include the $\mathcal{O}(N)$ calculation of forces, the equations of motion, time integration, adaptive mesh refinement, the treatment of dislocation core reactions and the dynamic distribution of data and work on parallel computers. A simulation integrating all these algorithmic elements using the Parallel Dislocation Simulator (ParaDiS) code is performed to understand their behaviour in concert and to evaluate the overall numerical performance of dislocation dynamics simulations and their ability to accumulate percent of plastic strain.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

Dislocation dynamics (DD) simulation methods have the potential of directly connecting the physics of dislocations with the evolution of strength and strain hardening in crystalline materials. These methods simulate explicitly the motion, multiplication and interaction of discrete dislocation lines, the carriers of plasticity in crystalline materials, in response to an applied load. While the fundamentals of the DD method have been established for a few decades, the challenge in connecting the aggregate behaviour of dislocations to macroscopic material response, such as strain hardening, has been and remains one of computability. For a reasonable choice of simulation volume and initial dislocation density, most existing DD simulations are limited to plastic strains of $<0.5\%$ (Zbib *et al* 2000, Devincre *et al* 2001, Madec *et al* 2002), because the computational expense, along with the total number of

[3] Author to whom any correspondence should be addressed.

dislocation segments in the simulation, increases dramatically at yield. Therefore, the information that DD simulations have been able to provide on the nature of strain hardening has been limited. In this paper, we focus on the development of DD algorithms in the pursuit of extending both the spatial and the temporal scales of these simulations.

Several DD simulation methods have been presented over the past several decades (Kubin *et al* 1992, Devincre 1996, Rhee *et al* 1998, Zbib *et al* 1998, Schwarz 1999, 2003, Ghoniem and Sun 1999, Ghoniem *et al* 2000, Shenoy *et al* 2000, Weygand *et al* 2002). While distinctions can be made between methods, there are basic features that all the codes have in common. All the codes discretize the general curvilinear dislocation geometry into a finite set of degrees of freedom. The calculation of forces on these degrees of freedom consists of an N-body problem where the bodies in this case are dislocation line segments. As in other N-body problems, every segment interacts, in principle, with every other segment in the system, and approximations are introduced for interactions between distant segments to reduce the order of the calculation from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. The position of the degrees of freedom is updated through predefined equations of motion and a time integration algorithm. Since the dislocation lines tend to multiply dramatically (more than two orders of magnitude) during these simulations, all the methods include procedures for adaptively refining the discretization. The most complex features of dislocation dynamics codes involve the treatment of dislocation core reactions. Core reactions include full annihilation reactions, partial annihilation reactions leading to junctions, cross slip events and other such events that occur at defect intersections. The different simulation methods cited above differ most in their treatment of core reactions.

While the accuracy of various DD algorithms has been well studied, there is little published information on the computational performance of DD codes, except for the level of plastic strain they are able to attain. The DD class of simulation codes presents unique computational challenges. The number of active degrees of freedom increases dramatically during the course of the simulation, and the position of any one degree of freedom may cross the simulation volume multiple times when periodic boundary conditions are imposed. The interaction between the degrees of freedom is long range $(1/r)$, and there are potentially sharp gradients in the short range. The spatial distribution of the degrees of freedom can be extremely heterogeneous due to the tendency of dislocations to cluster and pattern. Finally, dislocation core reactions introduce discontinuous topological events that must be detected and handled efficiently.

There are three computational strategies that must be investigated to increase the computability of dislocation dynamics and enable simulations of strain hardening. The first strategy is to minimize the computational expense of a simulation time step iteration. This can be achieved by describing the curvilinear dislocation network with the fewest degrees of freedom and efficiently calculating the forces on those degrees of freedom with a desired level of accuracy. The second strategy is to maximize the size of the simulation time step. The time step size is most sensitive to the algorithmic details of the simulation code: equations of motion, time integration scheme, adaptive mesh refinement and the treatment of core reactions. The third strategy is to use parallel computing and to achieve strong-scalability. A parallel code can achieve good scaling characteristics by evenly distributing the computational load across all the available processors while simultaneously minimizing the amount of communication that is done between them.

Occasionally, the three strategies outlined above may be in conflict with each other. For example, a method that extends the size of a time step may increase the computational cost of each time step, or an algorithm that leads to higher efficiency on a single processor may deteriorate the strong scaling properties. In such cases, the ultimate decision as to which

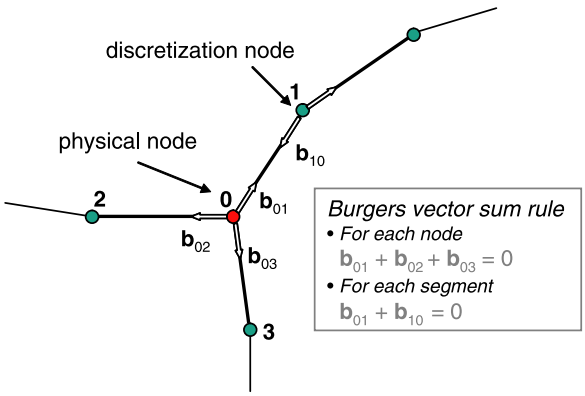

Figure 1. Schematic depicting the method of discretization of dislocation lines into inter-connected piecewise linear segments terminating at discretization nodes or physical nodes. Discretization nodes connect two dislocation segments while physical nodes connect an arbitrary number of segments. Burgers vector conservation is enforced at both types of nodes. The white arrows represent the line directions over which the Burgers vectors are defined, not the Burgers vectors themselves.

strategy to follow is determined by the overall goal of reaching large strains with large simulation boxes in the least amount of wall clock time.

In the following sections, we focus on the DD algorithm developments at Lawrence Livermore National Laboratory associated with the Parallel Dislocation Simulator (ParaDiS) project. The goal of the ParaDiS code project is to develop a simulation tool capable of performing large scale DD simulations of strain hardening. The sections are organized in a linear fashion with respect to the three computational strategies outlined above. Section 2 begins with the dislocation line discretization and calculation of forces followed by a discussion on equations of motion and time integration procedures in section 3. The discussion then transitions to handling of discrete topological events associated with adaptive remeshing and dislocation core reactions in sections 4 and 5. Finally, the implementation of DD on parallel computing architectures and the integrated performance of the ParaDiS code are discussed in sections 6 and 7.

2. Dislocation line discretization and force calculation

2.1. Line discretization

In our quest for reaching large strains, we focus on discrete representations that introduce the fewest number of active degrees of freedom while adequately capturing the general line topology. With this in mind, we choose to discretize the system into a series of inter-connected linear segments equivalent to the discretization employed by Rhee *et al* (1998), Kukta (1998) and Weygand *et al* (2002). As shown in figure 1, each linear dislocation segment is terminated by either a discretization node or a physical node. A discretization node connects two segments, while a physical node may connect more than two segments. Singly connected nodes are forbidden from the interior of a crystal and may only be employed as boundary nodes lying on surfaces. Furthermore, to avoid redundancy, no two nodes can be directly connected together by more than one segment, and every segment must have a non-zero Burgers vector. The active degrees of freedom, on which forces and equations of motion must be developed, are the positions of all the nodes in the system.

The conservation of Burgers vector must be enforced everywhere on the dislocation network during the entire course of the simulation. We use the indexing convention that $\boldsymbol{b}_{ij}$ represents the Burgers vector of a line segment $\boldsymbol{l}_{ij} = \boldsymbol{X}_j - \boldsymbol{X}_i$, which starts at node i positioned at $\boldsymbol{X}_i$ and terminates at node j positioned at $\boldsymbol{X}_j$. Following this convention, every segment in the simulation must satisfy $\boldsymbol{0} = \boldsymbol{b}_{ij} + \boldsymbol{b}_{ji}$, and every node i must satisfy the condition $\boldsymbol{0} = \sum_j \boldsymbol{b}_{ij}$. The position $\boldsymbol{x}_{ij}$ of a point on segment $\boldsymbol{l}_{ij}$ is defined as

$$\boldsymbol{x}_{ij}(l) = \mathcal{N}(-l)\boldsymbol{X}_i + \mathcal{N}(l)\boldsymbol{X}_j \tag{1}$$

$$\text{with } \mathcal{N}(l) = \tfrac{1}{2} + l \quad \text{and} \quad -\tfrac{1}{2} \leqslant l \leqslant \tfrac{1}{2} \tag{2}$$

such that $\boldsymbol{x}_{ij}(-1/2) = \boldsymbol{X}_i$ and $\boldsymbol{x}_{ij}(1/2) = \boldsymbol{X}_j$. Likewise, since our linear segments must remain linear during their motion, the velocity $\boldsymbol{v}_{ij}$ of a point on segment $\boldsymbol{l}_{ij}$ takes the form

$$\boldsymbol{v}_{ij}(l) = \mathcal{N}(-l)\boldsymbol{V}_i + \mathcal{N}(l)\boldsymbol{V}_j, \tag{3}$$

where $\boldsymbol{V}_i$ and $\boldsymbol{V}_j$ are the velocities of nodes i and j, respectively.

This discretization scheme of the dislocation network leads to a set of lines with positional continuity but with discontinuous tangents and undefined curvature at each node. Higher order discretization schemes have been considered by other researchers which satisfy the tangential and curvature continuity of the dislocation lines (Ghoniem and Sun 1999, Schwarz 1999). The desire to increase the level of continuity of the discretization stems partly from the difficulty in applying the classical singular continuum elastic theory of dislocations at locations with infinite curvature. However, these locations do appear in nature at points of dislocation intersection so the inability of the singular theory of dislocations to describe a configuration with tangential discontinuities cannot be avoided all together. Rather than raise the level of continuity of the discretization, we employ a recently developed non-singular continuum elastic theory of dislocations that requires positional continuity only and is capable of describing the forces acting on all points in the discrete network (Cai *et al* 2006).

2.2. *Calculation of nodal forces*

The force $\boldsymbol{F}_i$ on a node i is defined as the negative derivative of the stored energy $\mathcal{E}$ in the system with respect to the nodal position such that

$$\boldsymbol{F}_i \equiv -\frac{\partial\, \mathcal{E}(\{\boldsymbol{X}_j, \boldsymbol{b}_{jk}, \boldsymbol{T}^{\mathrm{s}}\})}{\partial\, \boldsymbol{X}_i}, \tag{4}$$

where the stored energy $\mathcal{E}$ is a function of the positions of all nodes in the system, the connections between them as defined by the Burgers vectors, and the externally applied surface traction $\boldsymbol{T}^{\mathrm{s}}$. It is convenient to partition the total stored energy of the system into two parts: one associated with the local atomic configuration of the dislocation cores $\mathcal{E}^{\mathrm{c}}$ and another associated with the long range elastic distortion $\mathcal{E}^{\mathrm{el}}$ that can be well described by continuum elasticity theory, such that $\mathcal{E} = \mathcal{E}^{\mathrm{c}} + \mathcal{E}^{\mathrm{el}}$. Likewise, the force on a node i may be similarly partitioned such that $\boldsymbol{F}_i^{\mathrm{c}}$ and $\boldsymbol{F}_i^{\mathrm{el}}$ derive from the spatial derivatives of the core and elastic energies, respectively, and $\boldsymbol{F}_i = \boldsymbol{F}_i^{\mathrm{c}} + \boldsymbol{F}_i^{\mathrm{el}}$.

The core energy $\mathcal{E}^{\mathrm{c}}$ is the portion of the stored energy associated with dislocations that cannot be accounted for in the linear elastic strain energy model that forms the basis of DD simulations. Typically, this portion of the energy can be attributed to the atoms in the region near the dislocation cores where the lattice deformation diverges significantly from that predicted by the linear elastic theory (Henager and Hoagland 2005). This energy contribution should be short ranged and can be well approximated by an energy per unit length ϵ^{c} as a function of the dislocation's Burgers vector $\boldsymbol{b}$ and line direction $\boldsymbol{t}$. This treatment of the core energy introduces an additional self-energy term for each segment but ignores any changes to the

segment–segment interaction energy that may occur in the short range when dislocation cores overlap. As will be shown in section 5, the core energy contributions to self-energy play an important role in the formation and breaking of dislocation junctions.

The total core energy of the discretized dislocation network can be expressed by the following sum:

$$\mathcal{E}^{\mathrm{c}} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \epsilon^{\mathrm{c}}(\boldsymbol{b}_{ij}, \boldsymbol{t}_{ij}) \|\boldsymbol{l}_{ij}\|, \tag{5}$$

where $\boldsymbol{t}_{ij} = \boldsymbol{l}_{ij}/\|\boldsymbol{l}_{ij}\|$ is the unit vector along segment $\boldsymbol{l}_{ij}$. The sum is arranged so that the contribution of every segment in the network is counted once. The function $\epsilon^{\mathrm{c}}(\boldsymbol{b}_{ij}, \boldsymbol{t}_{ij})$ describes the core energy variation with dislocation Burgers vector and line orientation and must be obtained by atomistic simulation. The corresponding dislocation core force $\boldsymbol{F}_i^{\mathrm{c}}$ on node i is

$$\boldsymbol{F}_i^{\mathrm{c}} = -\frac{\partial \mathcal{E}^{\mathrm{c}}}{\partial \boldsymbol{X}_i} = \sum_j \boldsymbol{f}_{ij}^{\mathrm{c}}, \tag{6}$$

$$\boldsymbol{f}_{ij}^{\mathrm{c}} = \epsilon^{\mathrm{c}}(\boldsymbol{b}_{ij}, \boldsymbol{t}_{ij})\boldsymbol{t}_{ij} + (\boldsymbol{I}_2 - \boldsymbol{t}_{ij} \otimes \boldsymbol{t}_{ij}) \cdot \frac{\partial \epsilon^{\mathrm{c}}(\boldsymbol{b}_{ij}, \boldsymbol{t}_{ij})}{\partial \boldsymbol{t}_{ij}}, \tag{7}$$

where $\boldsymbol{I}_2$ is the second order identity tensor. The force $\boldsymbol{f}_{ij}^{\mathrm{c}}$ can be considered as the part of the core force on node i attributable to its connection to node j. The first term in equation (7) is a line tension that acts to shrink a segment's length, while the second term is a moment that acts to rotate segments to lower core energy orientations. It can be easily shown that $\boldsymbol{f}_{ji}^{\mathrm{c}} = -\boldsymbol{f}_{ij}^{\mathrm{c}}$, and this equality is used to eliminate redundancy in calculating the core force of connected nodes.

To obtain an expression for the elastic force at a node, it is more convenient to use a virtual work argument (Kukta 1998, Weygand *et al* 2002) than to differentiate the elastic energy $\mathcal{E}^{\mathrm{el}}$ directly. It is similarly convenient to partition $\boldsymbol{F}_i^{\mathrm{el}}$ into contributions that can be attributed to each of the segments connected to node i as $\boldsymbol{F}_i^{\mathrm{c}}$ was partitioned in equation (6) such that

$$\boldsymbol{F}_i^{\mathrm{el}} = \sum_j \boldsymbol{f}_{ij}^{\mathrm{el}}, \tag{8}$$

where $\boldsymbol{f}_{ij}^{\mathrm{el}}$ is the contribution to the elastic force on node i due to its connection to node j. Applying the principle of virtual work to the motion of node i results in the following integral expression for $\boldsymbol{f}_{ij}^{\mathrm{el}}$:

$$\boldsymbol{f}_{ij}^{\mathrm{el}} \equiv \|\boldsymbol{l}_{ij}\| \int_{-1/2}^{1/2} \mathcal{N}(-l)\boldsymbol{f}^{\mathrm{pk}}(\boldsymbol{x}_{ij}(l))\,\mathrm{d}l, \tag{9}$$

$$\boldsymbol{f}^{\mathrm{pk}}(\boldsymbol{x}_{ij}) = [\boldsymbol{\sigma}(\boldsymbol{x}_{ij}) \cdot \boldsymbol{b}_{ij}] \times \boldsymbol{t}_{ij}, \tag{10}$$

where $\boldsymbol{f}^{\mathrm{pk}}$ is the Peach–Koehler force at point $\boldsymbol{x}_{ij}$ on the dislocation segment, as defined by the stress $\boldsymbol{\sigma}$, line direction $\boldsymbol{t}_{ij}$ and Burgers vector $\boldsymbol{b}_{ij}$ at that point. If we limit our treatment to linear elastic continua, the dislocation segment's own stress field, the stress field of all other dislocation segments in the continuum and the stress field from the imposed surface traction $\boldsymbol{T}^{\mathrm{s}}$ may be superposed to yield the local stress $\boldsymbol{\sigma}$. Likewise, the elastic force $\boldsymbol{f}_{ij}^{\mathrm{el}}$ may also be written as a superposition of three contributions,

$$\boldsymbol{f}_{ij}^{\mathrm{el}} = \boldsymbol{f}_{ij}^{\mathrm{ext}} + \boldsymbol{f}_{ij}^{\mathrm{s}} + \sum_{k=1}^{n-1} \sum_{l=k+1}^{n} \boldsymbol{f}_{ij}^{kl} \qquad \text{and} \qquad [k, l] \neq [i, j] \text{ or } [j, i], \tag{11}$$

where $\boldsymbol{f}_{ij}^{\text{ext}}$ is the contribution from the surface tractions, $\boldsymbol{f}_{ij}^{s}$ is the force on node i in response to the segment ij's own stress field and $\boldsymbol{f}_{ij}^{kl}$ is the force on node i due to the interaction between segments ij and kl.

If the applied surface tractions lead to a uniform stress field $\boldsymbol{\sigma}^{\text{ext}}$ in the crystal, as is commonly the case in DD simulations with periodic boundary conditions, then

$$\boldsymbol{f}_{ij}^{\text{ext}} = \tfrac{1}{2}\{[\boldsymbol{\sigma}^{\text{ext}} \cdot \boldsymbol{b}_{ij}] \times \boldsymbol{l}_{ij}\}, \tag{12}$$

and $\boldsymbol{f}_{ji}^{\text{ext}} = \boldsymbol{f}_{ij}^{\text{ext}}$. However, if a spatially varying stress field results from the externally applied tractions, the integral in equation (9) must be evaluated with $\boldsymbol{\sigma}$ replaced by $\boldsymbol{\sigma}^{\text{ext}}$. Discussion on the implementation of boundary conditions in finite elastic media may be found in the works of van der Giessen and Needleman (1995), Fivel *et al* (1996), Khraishi and Zbib (2002), Liu and Schwarz (2005) and Tang *et al* (2006).

To evaluate the force on a node due to the stress field of the segments it connects and all other segments in the system we use the newly developed non-singular expressions for the stress field of a dislocation segment developed by Cai *et al* (2006). In this new theory, the core singularity in the classical solutions of the Volterra dislocation is removed by introducing a spherically symmetric Burgers vector distribution at every point on a dislocation line. The distribution is parametrized by a single variable a, the spread width, and is special in that it leads to simple analytical expressions for the interaction and self-energies as well as the stress fields of linear segments. Furthermore, analytical expressions may also be obtained for $\boldsymbol{f}_{ij}^{s}$ and $\boldsymbol{f}_{ij}^{kl}$ in infinite isotropic elastic continua. These analytical expressions are given in appendix A. The calculation of forces for DD simulations in anisotropic elastic continua are discussed in detail within the works of Bacon *et al* (1979), Rhee *et al* (2001) and Han *et al* (2003) and are outside the scope of this work.

There are several advantages for computing the elastic force contributions based on the non-singular expressions. First, the force expressions are continuous, smooth and well defined everywhere in space; the same is true for their spatial derivatives. The smoothness of the elastic force field can potentially allow the simulation to take large time steps that would otherwise be inhibited if the forces were discontinuous. Second, it requires minimal continuity conditions for the discretized dislocation network, and no further approximation is required to handle physical nodes where the local curvature may be undefined. Lastly, the forces on dislocations at point of intersection are still well defined within the non-singular theory. In this sense, the non-singular expressions provide a complete elastic theory of dislocations that is attractive for use in numerical simulations.

One evaluation of $\boldsymbol{f}_{ij}^{kl}$ for a given pair of segments, using the expressions in appendix A, is approximately 3.5 times more expensive than one evaluation of the stress field of a segment at a point, using the expressions given by Cai *et al* (2006). Therefore, under certain conditions it may be more computationally efficient to numerically integrate $\boldsymbol{f}_{ij}^{kl}$ through a Gaussian quadrature than to use the analytical expressions if a tolerable error can be achieved. However, if $\boldsymbol{f}_{ij}^{kl}$, $\boldsymbol{f}_{ji}^{kl}$, $\boldsymbol{f}_{lk}^{ij}$ and $\boldsymbol{f}_{kl}^{ij}$ are all evaluated simultaneously, the evaluation of all four interaction forces between two segments is approximately 3.9 times more expensive than a single stress evaluation. The expense of finding all four interaction forces does not increase arithmetically because symmetries in the vectors and the integrals used in a single force evaluation can be exploited to find the other three with little additional effort. Thus, computing the nodal force by the Gaussian quadrature of stress field is more attractive only when an acceptable accuracy can be achieved with a single integration point per segment.

The computational expense of numerically integrating the nodal forces between two interacting dislocation segments is examined in figure 2 as a function of the normalized distance between the two segments. The computational expense of the calculation is directly

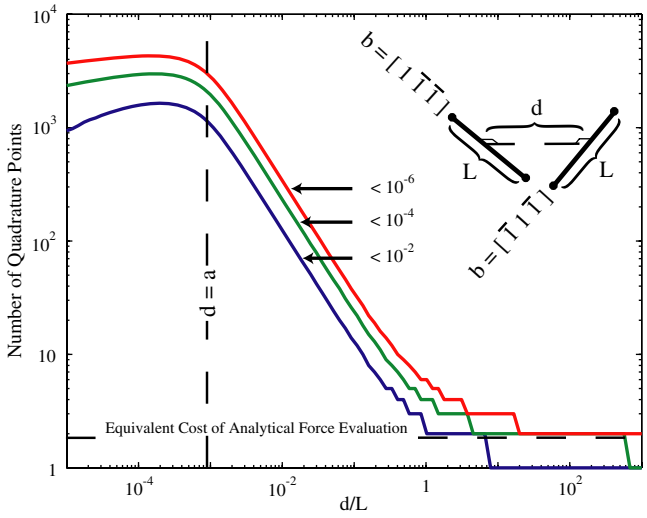

Figure 2. Plot of the number of Gauss quadrature points needed to calculate the force at the end node of a screw dislocation with Burgers vector $\boldsymbol{b} = [1\ \bar{1}\ \bar{1}]$ due to a screw dislocation with Burgers vector $\boldsymbol{b} = [\bar{1}\ 1\ \bar{1}]$ for a desired level of accuracy ϵ as a function of the distance between them. The computational cost for evaluating the analytical force expressions given in appendix A is shown relative to the computational cost of evaluating the forces through numerical integration if the forces on all four end nodes are calculated simultaneously.

proportional to the number of Gauss quadrature points that are needed to reach the desired level of accuracy with the reference force value obtained from the analytical expressions in appendix A. When the separation d between the segments is much larger than the segment length L, the number of quadrature points necessary to reach a specified accuracy level eventually drops below 2, at which point it becomes more efficient than the analytic expression. At these distances, other more efficient lumped source approximations using a fast multipole method (FMM) will be used to compute the interaction between segments. Therefore, when dislocation segments are close enough such that their interactions need to be accounted for individually, we find that the most efficient way to compute nodal forces is to use the analytic expressions given in appendix A.

No matter how efficient the nodal force computation from a pair of segments becomes, the total amount of calculation scales $\mathcal{O}(N^2)$ for a system of N segments, if the interaction between every segment pair is accounted for individually. This means that the force computation quickly becomes too expensive to be feasible in large scale simulations (Kukta 1998, Schwarz 1999, Weygand *et al* 2002). One approach to reduce the computational expense is to simply neglect the interactions between segments that are further than a predefined cut-off distance (Kubin *et al* 1992, Devincre 1996, Devincre and Kubin 1997). Since the stress field of a dislocation in an infinite elastic medium is long ranged and decays slowly $(1/r)$, the introduction of a cut-off distance can lead to numerical artefacts in the force field. In other physical systems that contain long range forces, such as Coulomb interactions in ionic crystals or gravitational fields in galaxies, the fast multipole method (FMM) (Greengard and Rokhlin 1997) is typically used to incorporate remote interactions in simulations without the need of a cut-off distance. Lumped source approximations enabled by FMM and its variants have been applied in other DD simulation codes to account for the elastic interaction between segments considered well

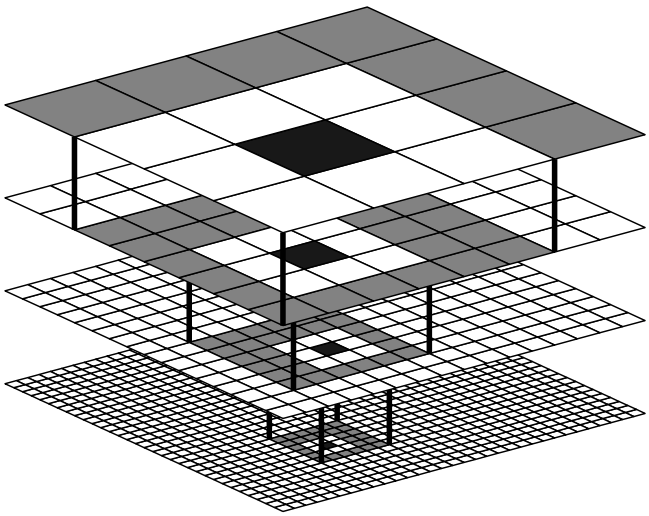

Figure 3. Schematic depicting hierarchical tiling used for force computations for segments contained in the computational cells in black. The segments in the black cell at the lowest level in the hierarchy interact with other segments in the same cell and neighbouring cells through the explicit n^2 calculation. The black cells and neighbouring white cells at other levels in the hierarchy are treated by finer levels. Interactions between the black cells and the cells shaded in grey interact through a lumped source approximation and a fast multipole algorithm at all levels of the hierarchy. The interaction of the black cells with unshaded cells outside the grey cells on the same level is treated using the fast multipole algorithm at coarser levels.

separated (Zbib *et al* 1998, LeSar and Rickman 2002, Wang *et al* 2004). Here we present a hierarchical FMM algorithm that extends the developments of LeSar and Rickman (2002) and Wang *et al* (2004).

Our FMM implementation relies on the commonly used hierarchical grid structure shown in figure 3 where the total number of computational subcells at the lowest level is restricted to 8^q in three dimensions, and the number of levels in the hierarchy is $q + 1$. Consider a cell C in the lowest level of the hierarchy and the union U of its nearest neighbours plus itself (27 cells in total). The nodal forces on a segment contained in C due to its local interaction with all the segments contained in U are accounted for explicitly using analytic expressions in appendix A. The nodal forces on a segment contained in C due to its remote interaction with segments outside U are accounted for by the FMM algorithm.

The multipole expansion of a cell in the lowest level of the hierarchy is calculated from the individual dislocation segments contained in that cell. The multipole expansions of cells at higher levels in the hierarchy are calculated by using an 'upward pass' translation algorithm in which the multipole moments of daughter cells one level lower are translated and combined to form the multipole moments of parent cells. In our implementation, every parent cell in the hierarchy has 8 daughter cells.

Once the moments of multipole expansion have been established for all the cells in the hierarchy, we compute the stress field in the cells at the lowest level of the hierarchy due to the multipole expansions of cells considered well separated. The resulting stress field in a cell is approximated using a Taylor series expansion about the centre of the cell. Assuming linear elasticity, we can superpose the contribution of multipole moments from cells at different levels of the hierarchy.

The Taylor series of the stress due to remote interactions for any cell within the hierarchy is built in two parts. The first requires calculating the stress field in every cell (black squares

in figure 3) due to the multipole moments in 189 cells that are outside its nearest neighbour distance but within the nearest neighbour distance of its parent cell (shaded squares in figure 3). The second part requires performing a 'downward pass' algorithm that moves the centre of the Taylor series expansion of the stress field in the parent cell to the centres of its daughter cells and adds it to the contribution of the first part. The first part can be done simultaneously for all the cells within the hierarchy. The second part must be done after the first and recursively through the fast multipole hierarchy starting at the top. Finally, the force contribution on the end nodes of a segment due to its remote interactions is calculated by performing Gaussian quadrature on the Taylor series of the stress field in the cell containing the segment.

To build the multipole moments of a cell from dislocation segments contained in it, and to obtain the Taylor series expansion of the stress field in one cell due to the multipole moments in another, we use the expressions developed by LeSar and Rickman (2002), Wang *et al* (2004). Modifications are made to further increase the computational efficiency. Since the spatial derivatives of $R \equiv \sqrt{x^2 + y^2 + z^2}$ are independent of the order in which they are taken (e.g. $R_{xy} = R_{yx}$), a factor of almost 6 can be saved by taking advantage of these inherent symmetries in the derivatives of R and not recomputing equivalent derivatives. Likewise, the terms in the multipole expansion that operate on these equivalent derivatives can be reorganized and regrouped to further reduce the number of operations required to calculate a Taylor series expansion of the stress field in remote cells. Details of our numerical implementation of the FMM for DD are contained in appendix B.

In application, the multipole moments (of the dislocation density) and Taylor series expansions (of the stress field) must be truncated at a certain order. In our implementation, the order of the multipole moments and Taylor series expansions are independent of the level of the hierarchy on which the expansions reside but are dependent on one another. Their interdependence stems from the competition between accuracy and computational expense. If the multipole moments are truncated at a low order, pushing the Taylor series expansions to high orders will significantly increase the computational cost without improving the accuracy. Conversely, the computational cost of keeping high orders of multipole moments is wasted if the Taylor series expansions are truncated at lower orders. As shown in figure 4, the optimal ratio between the order of multipole moments and Taylor series expansions is approximately 2, and figure 5 shows the relative expense of calculating the stress field in a computational subcell at the lowest level of the hierarchy for a given truncation limit in the multipole moments keeping the optimal ratio found in figure 4. As shown, the expense grows rapidly with increasing accuracy.

Incorporation of this fast multipole method algorithm leads to $\mathcal{O}(N)$ scaling of the remote segment interaction forces where N is the total number of segments in the simulation. However, since the interaction between dislocation segments in the same subcell or in neighbouring subcells is still calculated using the analytical expressions that scale $\mathcal{O}(n^2)$ locally, where n is the number of segments per cell, achieving $\mathcal{O}(N)$ scaling for the total force calculation is not trivial. It requires a judicious choice in the number of FMM cells, n_c, because certain parts of the FMM algorithm that scale $\mathcal{O}(n_c)$ may come to dominate the calculation if the ratio of n_c/N becomes too large. As illustrated in figure 6, there always exists an optimal choice for n_c for a given range of N. Consequently, during a strain hardening simulation, the number of fast multipole subcells must increase as the total number of degrees of freedom increases.

In strain hardening simulations, a computationally feasible simulation volume is much smaller than the volume of macroscopic specimens used in typical tensile straining experiments. To overcome this limitation, periodic boundary conditions (PBC) can be applied to reduce the influence of unwanted boundaries in a finite simulation box. The interaction of the dislocation segments in the simulation box with all of their periodic images is handled within

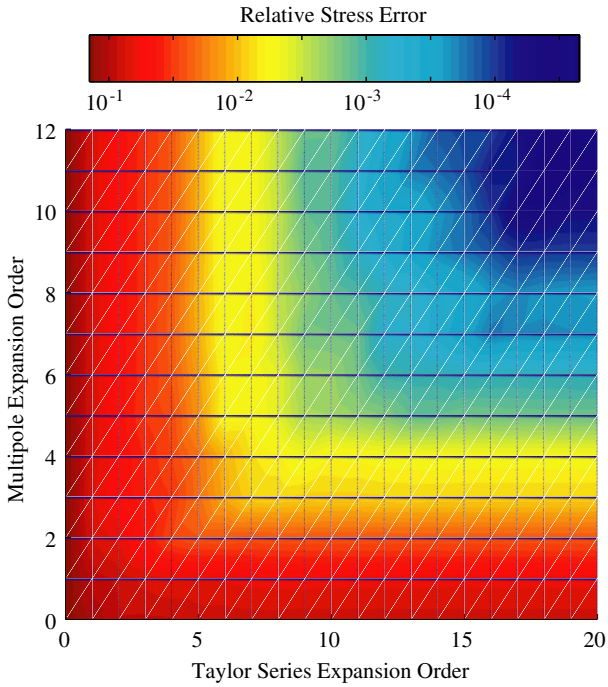

Figure 4. Contour depicting the relative error in the stress calculated using a fast multipole method as a function of the truncation in the multipole moment expansion order and the Taylor series expansion order.

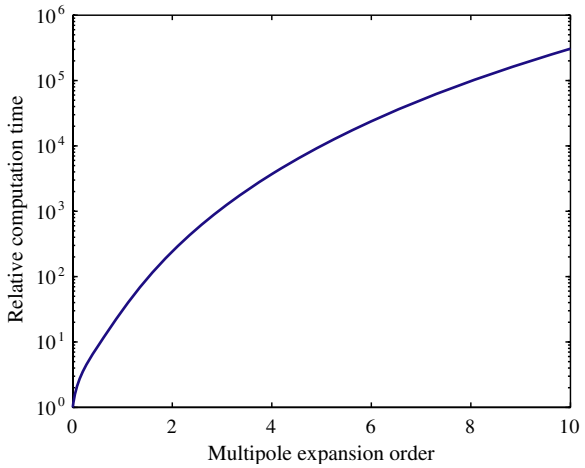

Figure 5. Computational expense as a function of the truncation multipole moment expansion relative to the expense of a zeroth order truncation. The truncation of Taylor series expansion is maintained at a constant factor of 2 of the truncation of the multipole moment expansion.

the FMM algorithm by calculating the Taylor series expansion of the stress field at the top-most level of the hierarchy due to all the images outside the nearest neighbour distance (Bulatov and Cai 2006). The images that are within the nearest neighbour distance of the parent cell are incorporated by operations at lower levels in the hierarchy. The conditional

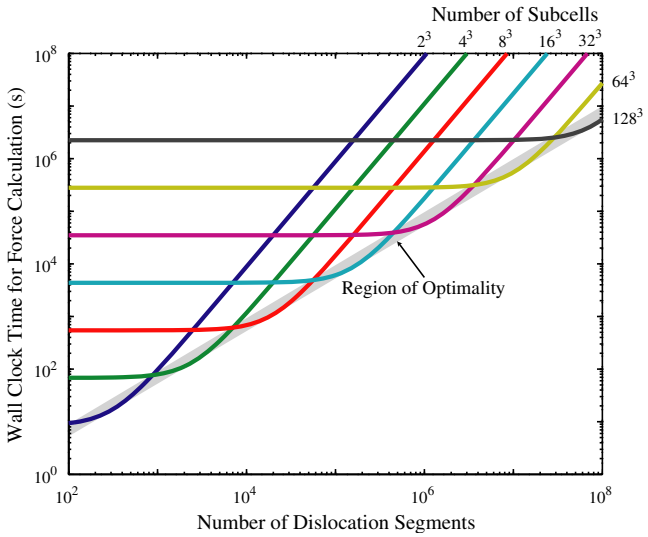

Figure 6. Plot depicting the optimal number of computational subcells as a function of the total number of dislocation segments in the system. The plot assumes that the spatial distribution of segments is homogeneous and that the force calculation will be partitioned between the local n^2 and non-local lumped sources as shown in figure 3.

convergence problem (Cai *et al* 2003) associated with long range interactions can be avoided at the top-most level of the hierarchy by numerically solving an infinite series expression with appropriate corrections (Challacombe *et al* 1997). The implementation of periodic boundary conditions may lead to spurious annihilations between dislocations and their negative periodic images. These spurious annihilations can be mitigated by enabling the dislocations to climb and cross-slip, by choosing the edges of the periodic cell to be different irrational lengths and/or by taking some care in specifying the glide planes relative to the edges of the simulation cell.

3. Equations of motion and time integration

3.1. Equations of motion

With the nodal forces determined, the nodal equations of motion are completed by specifying the response of the nodes to these forces through mobility functions. It is in these mobility functions that the material specificity of a DD model becomes most apparent. The material specific parameters that were required for the force calculation were a limited set of well-defined constants including the isotropic elastic constants, Burgers vectors and dislocation core energies and core radii. In comparison, the nodal mobility functions may require many more parameters to detail the kinetic response of a dislocation network and can contain complex non-linear expressions with functional dependences on the geometry of the dislocation, pressure, temperature and local forces. For clarity, we omit the explicit dependence of the mobility function on all factors other than the local forces for the expressions developed in this section.

Since we are mainly interested in simulating strain hardening to large extents of plastic strain, we will assume that the integration time step will be sufficiently large so that the inertia of the dislocation segments can be neglected. Mobility functions in this overdamped regime

typically take the form

$$\boldsymbol{v}(\boldsymbol{x}) = \mathcal{M}[\boldsymbol{f}(\boldsymbol{x})], \tag{13}$$

where $\boldsymbol{v}(\boldsymbol{x})$ is the velocity of point $\boldsymbol{x}$ on the dislocation line and $\boldsymbol{f}(\boldsymbol{x})$ is the force per unit length at the same point. $\mathcal{M}$ is a general mobility function whose arguments include not only the local force $\boldsymbol{f}$ but also other variables not included explicitly here.

Unfortunately, equation (13) is in a form that is not conducive to our discretization scheme of inter-connected linear segments. If we applied equation (13) as is to our discrete system, the velocity field in response to the local force per unit length would violate the constraint that our linear segments remain linear during their motion. Therefore, it is more convenient to invert the mobility function and define a local drag force per unit length $\boldsymbol{f}^{\text{drag}}$ along the dislocation line as a function of the velocity at that point, with the form

$$\boldsymbol{f}^{\text{drag}}(\boldsymbol{x}) = -\mathcal{M}^{-1}[\boldsymbol{v}(\boldsymbol{x})] = -\mathcal{B}[\boldsymbol{v}(\boldsymbol{x})], \tag{14}$$

where $\mathcal{B}$ is defined as the drag function that depends on the local velocity of the dislocation line and the other implicit variables. Again, the constraint that the velocity varies linearly along a segment means that the local equilibrium between the driving force $\boldsymbol{f}$ and the drag force $\boldsymbol{f}^{\text{drag}}$ cannot be enforced at every point $\boldsymbol{x}$ on the segment. Instead, equilibrium can be enforced in the weak sense at every node in the discretized dislocation network, i.e.

$$\boldsymbol{F}_i = -\boldsymbol{F}_i^{\text{drag}}, \tag{15}$$

where

$$-\boldsymbol{F}_i^{\text{drag}} \equiv \sum_j \|\boldsymbol{l}_{ij}\| \int_{-1/2}^{1/2} \mathcal{N}(-l)\mathcal{B}_{ij}[\boldsymbol{v}_{ij}(l)]\,\mathrm{d}l \tag{16}$$

is the integrated drag force at a node i and $\mathcal{B}_{ij}$ is the drag function of the segment connecting nodes i and j. Enforcing equilibrium in this weak sense leads to a sparse system of equations that relates the velocity of all nodes, $\{\boldsymbol{V}_i\}$, to forces on all nodes, $\{\boldsymbol{F}_i\}$. The bandwidth of the set of equations is narrow and limited to the order of the connectivity of the nodes in the system. Using the definition of the nodal drag force above, the power dissipation $\mathcal{P}^{\text{dis}}$ of the system takes a simple form

$$\mathcal{P}^{\text{dis}} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left\{ \int_{-1/2}^{1/2} \boldsymbol{v}_{ij}(l) \cdot \mathcal{B}_{ij}[\boldsymbol{v}_{ij}(l)]\,\mathrm{d}l \right\} = \sum_{i=1}^{n} \boldsymbol{F}_i \cdot \boldsymbol{V}_i. \tag{17}$$

The power dissipated by the system will play an important part in the treatment of dislocation core reactions discussed in section 5.

In the special case of linear mobility, i.e. $\mathcal{B}_{ij}[\boldsymbol{v}_{ij}] = \boldsymbol{B}_{ij} \cdot \boldsymbol{v}_{ij}$, the integral expression can be evaluated leading to the algebraic form

$$\boldsymbol{F}_i = \sum_j \frac{\|\boldsymbol{l}_{ij}\|}{6} \boldsymbol{B}_{ij} \cdot (2\boldsymbol{V}_i + \boldsymbol{V}_j) \qquad \text{for all } i, \tag{18}$$

where $\boldsymbol{B}_{ij}$ is the drag tensor for segment ij. The summation is over all nodes j that are connected to node i. An example of such a mobility law is given for BCC crystals in appendix C.

Instead of solving the global sparse system of equations to find the nodal velocities in response to nodal forces, approximations can be made to keep the calculation local. Both spatial and temporal approximations may be considered. If we assume that the dislocation network is well discretized, we can assume that the velocities of neighbouring, inter-connected

nodes are roughly the same, namely, $\boldsymbol{V}_i \approx \boldsymbol{V}_j$ and $\boldsymbol{v}_{ij}(l) \approx \boldsymbol{V}_i$. The approximation leads to a relationship between the nodal force and velocity of the form

$$\boldsymbol{F}_i \approx \frac{1}{2} \sum_j \|\boldsymbol{l}_{ij}\| \mathcal{B}_{ij}[\boldsymbol{V}_i]. \tag{19}$$

This approximation greatly simplifies the solution of the nodal mobilities by making the velocity at a particular node a function of the force and connectivity of the same node. The approximation breaks down around physical nodes where the mobility of neighbouring nodes is dramatically different. Alternatively, a temporal approximation can be made, in which the velocities of the nodes connected to node i are assumed not to vary significantly from one time step to another, namely, $\boldsymbol{V}_j^t \approx \boldsymbol{V}_j^{t-\Delta t}$. Here, the superscripts t and $t - \Delta t$ are used to denote the value at the present and previous time steps, respectively. This approximation again leads to a scenario where each nodal velocity can be obtained independently. While the temporal approximation may not break down around physical nodes like the spatial approximation does, it may fail immediately after discontinuous topological events (e.g. creation or deletion of nodes) or if the time step Δt becomes too large. Both the spatial and temporal approximations should be considered seriously because of their potential to significantly reduce the computational expense of a simulation time step.

3.2. Numerical time integration

With the nodal velocities defined, a suitable algorithm must be chosen to advance the nodal positions during a time step of the simulation. The simplest algorithm is the explicit Euler-forward method where

$$\boldsymbol{X}_i^{t+\Delta t} = \boldsymbol{X}_i^t + \boldsymbol{V}_i^t \, \Delta t. \tag{20}$$

The algorithm is computationally inexpensive in that it requires a single nodal force and velocity computation per time step. Unfortunately, the algorithm is subject to the Courant condition for numerical stability and is limited to relatively small time steps. The size of the time step is controlled by the ratio between the length of the shortest segment and the velocity of the fastest moving node. Because of a wide distribution of nodal velocities, the time step allowed by the Courant condition is usually too small for strain hardening simulations. A more sophisticated, e.g. implicit, time integrator can be more attractive if the size of the time step can be significantly increased.

An alternative to the explicit Euler-forward method is an implicit trapezoidal method in which the Euler-forward and Euler-backward methods are mixed to form

$$\boldsymbol{X}_i^{t+\Delta t} = \boldsymbol{X}_i^t + \tfrac{1}{2}(\boldsymbol{V}_i^t + \boldsymbol{V}_i^{t+\Delta t})\Delta t. \tag{21}$$

The implicit Euler-backward method is unconditionally stable, but solving the system of equation requires an iterative procedure that may involve multiple nodal force and velocity calculations. Although implicit update methods are more computationally expensive than explicit methods for a given time step, implicit methods can potentially enable significantly larger time steps compared with explicit methods. Implicit methods are favoured when the ratio between allowable time step sizes becomes greater than the ratio between their computational expense. In our experience with the ParaDiS code project, the computational expense of implicit time integration is justified in most cases because of the gains that are made in the time step size for most simulation conditions. However, care must be taken in crafting an efficient iterative solution technique.

Using a full Newton–Raphson method to solve equation (21) is prohibitively expensive because the bandwidth of the matrix that must be inverted is on the order of $81n_c$ and is

too large to be computationally useful. Jacobian based iterative methods or matrix-free iterative methods, such as Newton–Krylov methods (Kelley *et al* 2004) which require only force calculations, may be considered as an alternative to a direct solution method. Presently, there is little published on the use of implicit solvers within existing DD codes to *a priori* know which if any of these implicit methods is better than explicit time integration. To gauge if any implicit time integration may be better than the commonly used explicit scheme, we will employ a simple implicit scheme even though it may have poor convergence properties.

If we assume that the derivatives of all the nodal velocities with respect to all the nodal positions are small compared with $1/\Delta t$, they can be ignored, and an iterative update technique can be considered in which successive iterates take the form

$$\boldsymbol{X}_i^{t+\Delta t}(n+1) = \boldsymbol{X}_i^t + \tfrac{1}{2}(\boldsymbol{V}_i^{t+\Delta t}(n) + \boldsymbol{V}_i^t)\Delta t, \tag{22}$$

$$\boldsymbol{X}_i^{t+\Delta t}(1) = \boldsymbol{X}_i^t + \tfrac{1}{2}(\boldsymbol{V}_i^{t-\Delta t} + \boldsymbol{V}_i^t)\Delta t, \tag{23}$$

and the iteration ends when $\|\boldsymbol{X}_i^{t+\Delta t}(n+1) - \boldsymbol{X}_i^{t+\Delta t}(n)\| < r_{\text{tol}}$. Unfortunately, such an iteration scheme does not have good convergence properties in general, and it requires as many nodal force evaluations as the number of iterations before the solution converges within an acceptable accuracy. However, it is still relatively inexpensive for an implicit method, and time step sizes are large enough compared with time steps allowed under an explicit scheme to justify the added expense of multiple force evaluations per update.

Whereas the Courant stability condition gives rise to a natural time scale in explicit time integration schemes, implicit schemes do not have such a simple criterion to set the size of their time steps. To maximize simulation efficiency, we would like to use the largest time step that still allows the iteration in equation (22) to converge within a specified number of steps, but that time step size is not known *a priori*. However, we can choose to increase the time step by a constant factor whenever the iteration converges in the previous step, and to reduce it by another constant factor whenever the iteration fails to converge in the present step. Depending on the ratio of those two factors, the simulation will proceed with a distribution of time steps, and the code will perform sub-optimally, in that the maximum allowable time step is not always used. This approach can be improved by taking into account the error accepted in the previous time step, when deciding the step size to use in the present time step. A possible expression for such time control after a successful iteration is

$$\Delta t^{\text{new}} = \Delta t\; d \left[\frac{r_{\text{tol}}}{r_{\text{tol}} - (1-d^m)\max_i \|\boldsymbol{X}_i^{t+\Delta t}(n+1) - \boldsymbol{X}_i^{t+\Delta t}(n)\|} \right]^{1/m}, \tag{24}$$

where $d > 1$ and $m > 1$ are constants. The maximum increment of Δt from one step to the next is by a factor of d, when the maximum error in the previous step is zero. The expression can be extended to include a similar dependence on the number of iterations that were needed relative to the maximum number allowed. A similar expression can also be used to decrease Δt by a variable amount if the solution does not converge fast enough, but we find that a constant decrement is sufficient. Lastly, it is good practice to introduce an upper bound to the size of the time step to $\Delta t_{\max} = (l_c \rho |\boldsymbol{b}|)/\dot{\epsilon}$ where l_c is the linear dimension of an FMM subcell, ρ is the the dislocation density and $\dot{\epsilon}$ is the average strain rate of the simulation. The purpose of the upper bound is to keep dislocation segments from crossing multiple FMM subcells during a single time increment.

4. Topological operations

4.1. General topological operators

During the course of a simulation, it may be necessary to modify the topology of the discrete description of the dislocation network. Topological changes of the network are operations that add or remove degrees of freedom and change the connectivity of the system. In strain hardening simulations, the dislocation density, measured in line length per unit volume, may increase by two or more orders of magnitude. To maintain the same discretization accuracy at the end of the simulation as in the beginning, the number of discretization nodes must increase proportionally. Apart from discretization concerns, topological operations are needed to properly account for dislocation core reactions such as annihilation, junction formation, and cross slip events. Performing topological operations in such instances provides a more accurate description of the physical processes and potentially allows the use of larger time steps.

Changing the connectivity of the dislocation network through a topological operation is a discontinuous event, and it is not differentiable with respect to time. Therefore, during the update of the nodal positions, the degrees of freedom and their connectivity must remain constant; however, between updates, the number of degrees of freedom and their connectivity may change to improve the discrete description of the dislocation network. While changing the topology is a discontinuous operation, the changes should ideally be conducted on a fixed geometry. The position of the continuous dislocation network represented by the discretization should not be perturbed. Satisfying the fixed geometry constraint rigorously is difficult in practice, but any perturbations of the system should be small compared with the dislocation segment lengths and error tolerances within the simulation.

Apart from these properties, the implementation of topological operations should also be simple and robust. Simplicity of the operations enables efficient communication among processors in a parallel computing environment. Robustness ensures that, at the end of the operation, the Burgers vector is always conserved and that any two nodes share at most a single connection between them. We have developed two such general operations that we have termed *Mergenode* and *Splitnode* that satisfy these conditions for a node-based DD code. The two operations can be considered opposites of each other, and all complex topological procedures can be accomplished through different combinations of the two of them.

The *Mergenode* operation combines two nodes into one, as depicted in figure 7, thus reducing the number of degrees of freedom. During this operation, all the connections of one node are forwarded to the other, and the former node is deleted. All the connections on the remaining node are checked for (1) self-connections and (2) double connections to another node. The self-connections, if they exist, are deleted. Double connections are collapsed into a single connection by a Burgers vector sum. If the Burgers vector sum is zero, both connections (segments) to the common node are deleted. As a result of this deletion, if any node loses all its connections, it too is deleted. Finally, the position of the resultant node is updated, if necessary, to reflect its change in connectivity.

The *Splitnode* operation creates a new node and moves a set of selected connections from an existing node to the new node, as depicted in figure 8. If the sum of all Burgers vectors transferred to the new node is non-zero, a connection is made between the existing node and the new node to conserve the Burgers vector at both. The new node is often displaced away from the existing node immediately after *Splitnode*.

All topological procedures required for performing adaptive mesh refinement and dislocation core reactions can be constructed from combinations of these two elementary operations. For example, the reaction between two dislocation segments can be handled by

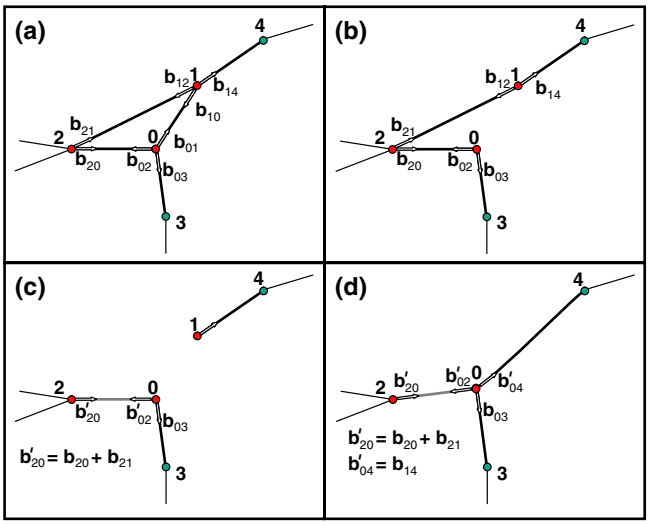

Figure 7. The sequence of topological operations that are conducted during a merging of two nodes 0 and 1 into one node 0. (*a*) Initial topology in the immediate vicinity of nodes 0 and 1. (*b*) Connections between node 0 and node 1 are deleted. (*c*) Connections between nodes 1 and 2 and other nodes in common are collapsed into single connections or no connection at all depending on the sum of Burgers vectors. (*d*) Remaining connections of the node to be removed, node 1, are transferred to node 0, and the position of node 0 is updated.

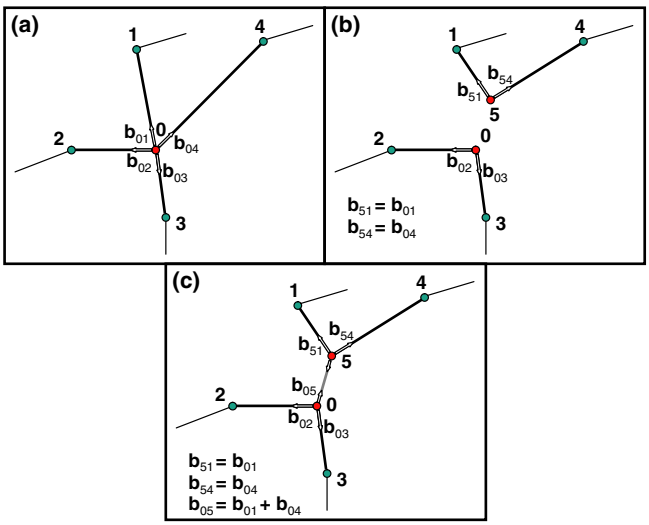

Figure 8. The sequence of topological operations that are conducted during the splitting of node 0 into two nodes 0 and 5. (*a*) Initial topology in the immediate vicinity of node 0. (*b*) The connections of node 0 to nodes 1 and 4 are removed and connected to a new node 5 with no change in Burgers vector. (*c*) A connection between nodes 0 and 5 is made to conserve Burgers vector at nodes 0 and 5, if necessary.

using a combination of *Mergenode* and *Splitnode* operations to create a physical node with four connections, and subsequent junction zipping or annihilation reactions can be handled by *Splitnode*. Logic as to when and how they should be used to perform these tasks must be specified.

4.2. Adaptive mesh refinement

Mesh adaption is a necessary component of DD simulations. This is especially true in a strain hardening simulation, in which the total length of the dislocation lines can increase by several orders of magnitude. The goal of mesh adaption is to optimize the numerical description of the continuous dislocation line geometry, so that a given level of accuracy is achieved with the fewest number of degrees of freedom. Operationally, a mesh refinement operation can be performed by using *Splitnode* on a discretization node to bisect one of its segments. A mesh coarsening operation can be performed by using *Mergenode* to combine a discretization node with one of its neighbouring nodes. It is very likely that the shape of the dislocation line will be slightly changed after a coarsening procedure. Hence, attention must be paid to ensure that the resulting shape change is within the error tolerances of the simulation.

Before a discussion into the logic associated with the addition and removal of discretization nodes in mesh adaption procedures can ensue, we must first consider the possibility of improving the discretization by simply rearranging the positions of existing nodes. The nodal forces acting on the discretization nodes within a DD simulation act to redistribute those nodes along the lines towards an optimal representation that minimizes the energy of the system.

An illustrative example of the nodal redistribution along a Frank–Read source held at a stressed equilibrium below its activation threshold is shown in figure 9. The equilibrium configuration of the continuous dislocation under this condition takes the form of an ellipse due to the variation in the line energy with the direction of the dislocation line. Initially, the line is discretized with two populations of segments lengths in a suboptimal fashion as shown in figure 9(*c*); however, as the simulation proceeds, the nodes redistribute along the line to form a more homogeneous distribution of segments lengths without changing the overall shape of the underlying ellipse. The enduring variation in the segment lengths is associated with the overall elliptical shape of the configuration as shown in figure 9(*d*). As a result of the redistribution, the energy of the discrete configuration decreases as shown in figure 9(*b*) and reaches a minimum when the nodes become optimally distributed.

Even though nodal forces will work to improve the discretization during a simulation, procedures for adaptively refining and coarsening the discretization through the insertion and removal of nodal degrees of freedom will be needed in strain hardening simulations where the length and connectivity of the dislocation network change dramatically. The initial placement of new discretization nodes during mesh refinement procedures and old nodes after mesh coarsening procedures does not have to be optimal because of the subsequent redistribution, but the logic as to where and when to add or remove nodes from the system must be developed simultaneously so that they do not conflict.

The simplest mesh adaption scheme is to adopt a minimum and maximum segment length, $L_{\min}$ and $L_{\max}$, and to bound every segment connected to a discretization node to be between those limits. The selection of $L_{\min}$ and $L_{\max}$ must be constrained by the inequality $2L_{\min} < L_{\max}$. Otherwise, a discretization node inserted to bisect a segment larger than $L_{\max}$ will lead to two segments smaller than $L_{\min}$ and meet the criterion for mesh coarsening. Even with this constraint, conflicts may still occur when the removal of a segment smaller than $L_{\min}$ results in a segment larger than $L_{\max}$. Another limitation of this simple scheme is that it does not include any consideration of the local curvature of the line.

A careful examination of figure 9(*b*) shows that in an optimal distribution, the nodes are more closely spaced in regions of higher curvature than in regions of lower curvature. The simple scheme outlined above will tend to enforce a more or less uniform segment length, regardless of the local curvature, counteracting the influence of configurational forces. As an improvement, the mesh adaption scheme can take into account the areas of discretization

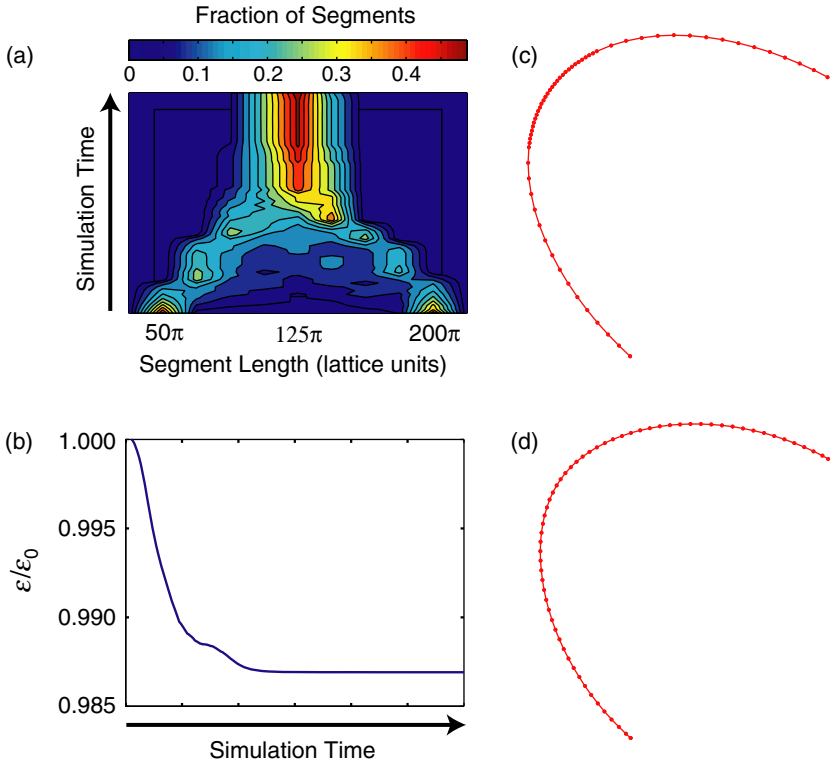

Figure 9. (*a*) Contour plot depicting the redistribution of segment lengths on a curved dislocation segment under a stressed equilibrium. The segments were initially placed in two equal populations with lengths of 50π and 200π measured in lattice units and ended in a more uniform distribution in response to configurational forces. (*b*) Change in the energy of the discrete system (normalized to the initial energy) during the course of the redistribution of the discretization nodes. (*c*) Initial node configuration. (*d*) Final node configuration.

A_i, where A_i is defined as the triangular area with vertices at discretization node i and its two neighbours. Along with the segment length bounds, we can define area bounds, $A_{\min}$ and $A_{\max}$, corresponding to the minimum and maximum areas for each discretization node. When the area is associated with a discretization node $A_i < A_{\min}$, the node i will be removed by a coarsening procedure. When $A_i > A_{\max}$, the discretization will be locally refined by bisecting both segments connected to node i. With this area criterion, high curvature regions will be discretized with short segments whereas low curvature areas will predominantly consist of long segments.

Similarly to the constraint $2L_{\min} < L_{\max}$, there are also constraints on $A_{\min}$ and $A_{\max}$,

$$0 \leqslant 4A_{\min} < A_{\max} \leqslant \frac{\sqrt{3}}{4} L_{\max}^2. \tag{25}$$

The upper bound on $A_{\max}$ leads to a condition where the area criterion is no longer used for mesh refinement and a segment is only bisected when its length exceeds $L_{\max}$. The lower bound on $A_{\min}$ leads to a criterion in which nodes are only removed when they connect co-linear segments such that their removal does not change the dislocation geometry. This is a strong criterion that is difficult to satisfy because of inevitable errors in numerical integrators. Since the position of any node is only accurate to within r_{tol}, A_i can potentially have an error as large

as $2\,r_{\text{tol}}L_{\max}$; therefore, a minimum area criterion that is consistent with the error tolerances of the simulation is $A_{\min} = 2\,r_{\text{tol}}L_{\max}$.

The inequality between $A_{\min}$ and $A_{\max}$ in equation (25) is needed because we choose to refine the mesh through segment bisection. The value of $A_{\min}$ proposed above logically leads to another constraint $\sqrt{3}L_{\max} > 32\,r_{\text{tol}}$. Since the new nodes inserted through the mesh refinement procedure are placed at the midpoint of existing segments, their area of discretization immediately after insertion is zero. An additional criterion for mesh coarsening, dependent on the time rate of change in A_i, must be introduced to keep these newly added nodes from being immediately coarsened. Node i should be removed through a mesh coarsening procedure only if $A_i < A_{\min} \bigcap \dot{A}_i < 0$.

A combined mesh refinement criterion for performing a segment bisection procedure, that accounts for both the segment length and the area associated with a discretization node, becomes

$$\text{Refine}(\boldsymbol{l}_{ij}) \equiv \Big(\max(A_i, A_j) > A_{\max} \bigcup \|\boldsymbol{l}_{ij}\| > L_{\max}\Big) \bigcap \|\boldsymbol{l}_{ij}\| > 2L_{\min}, \tag{26}$$

where $A_i \equiv 0$ for physical nodes, leaving the segment length criterion alone for the bisection of segments that connect two physical nodes. Likewise, a combined mesh coarsening criterion for performing a *Mergenode* operation on node i and a connected neighbour node becomes

$$\text{Coarsen}(i) \equiv \Big[\Big(A_i < A_{\min} \bigcap \dot{A}_i < 0\Big) \bigcup \min(\|\boldsymbol{l}_{ij}\|, \|\boldsymbol{l}_{ik}\|) < L_{\min}\Big] \bigcap \|\boldsymbol{X}_j - \boldsymbol{X}_k\| < L_{\max}, \tag{27}$$

where nodes j and k are connected to node i. Segments that connect two physical nodes can be bisected using the refinement criterion, but they cannot be deleted with a coarsening criterion because it is only applied to discretization nodes. Therefore, these segments are guaranteed to be smaller than $L_{\max}$ but are not necessarily larger than $L_{\min}$. The lower bound on these segment lengths will be defined by the annihilation distance r_{ann} introduced in the next section.

5. Dislocation core reactions

The topological treatment of dislocation core reactions is an important part of a DD simulation. Not only is it needed to correctly describe the physics of the dislocation core, but it also has a strong effect on the size of the simulation time step. In this section, we discuss the treatment of the most basic dislocation reactions, such as annihilation, junction zipping and unzipping. The list of dislocation core reactions can also include cross slip, dissociation of perfect and partial dislocations and dislocation reactions with other defects (such as free surfaces, grain boundaries and inclusions), which are beyond the scope of this paper. In the DD literature, there is no standard procedure for treating these reactions. Some existing DD codes (Kubin *et al* 1992, Schwarz 1999, Ghoniem *et al* 2000) have topological operations to account for annihilation but not for junction formation, while other existing DD codes (Rhee *et al* 1998, Weygand *et al* 2002) treat intersecting dislocations that either lead to junction forming reactions or annihilations through the same topological operations. We will investigate and evaluate these different strategies in terms of their physical accuracy and their impact on time step size.

Since the dislocation annihilation reaction is a special case of the more general junction formation reaction, we will simply focus our discussion on the general case of junction formation reactions. Annihilation reactions result from junction reactions that yield a net zero Burgers vector for the junction dislocation. Since the active degrees of freedom in the ParaDiS code are nodes connecting dislocation segments, the topological procedures developed

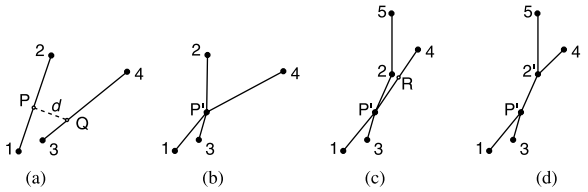

Figure 10. (*a*) The minimum distance $d_{\min}$ between two unconnected segments 1–2 and 3–4 is reached at points P and Q. Two segments are considered to be in contact if $d_{\min} < r_{\text{ann}}$. New nodes are introduced at points P and Q. (*b*) Nodes P and Q are merged into a single node P'. (*c*) Segment 2–5 comes into contact with segment P'-4, on which a new node R is added. (*d*) Merging nodes 2 and R into a new node $2'$ leads to the formation of junction segment P'-$2'$, if the Burgers vectors of segments P'-2 and P'-R do not cancel each other. If the Burgers vectors do cancel, segment P'-$2'$ is deleted leading to a dislocation annihilation reaction.

to treat core reactions will employ combinations of the *Mergenode* and *Splitnode* operations to modify the topology of the dislocation network.

5.1. Collision procedures

A DD algorithm must be able to detect dislocation network configurations that may be undergoing a core reaction such as the intersection of two unconnected line segments. The shortest distance, $d_{\min}$, between two unconnected segments $\boldsymbol{l}_{ij}$ and $\boldsymbol{l}_{kl}$ is found by minimizing the expression

$$d_{\min}(\boldsymbol{X}_i, \boldsymbol{X}_j; \boldsymbol{X}_k, \boldsymbol{X}_l) \equiv \min_{\alpha,\,\beta} \|\boldsymbol{x}_{ij}(\alpha) - \boldsymbol{x}_{kl}(\beta)\|$$

$$\text{with } i \neq j \neq k \neq l,\ -\tfrac{1}{2} \leqslant \alpha \leqslant \tfrac{1}{2} \text{ and } -\tfrac{1}{2} \leqslant \beta \leqslant \tfrac{1}{2}, \tag{28}$$

and the two segments can be considered intersecting if $d_{\min} \leqslant r_{\text{ann}}$, where r_{ann} is a predefined annihilation distance. For consistency with other lengths already introduced in the DD code, the r_{ann} should be comparable to $a/2$ and to $2\,r_{\text{tol}}$. At a distance of $a/2$, there is significant overlap in the cores of the two intersecting dislocations, and at a distance of $2\,r_{tol}$, the separation between the segments is within the acceptable error of the implicit time integrator introduced in section 3.2.

An efficient $\mathcal{O}(N)$ algorithm, based on the cell-list construction widely used in atomistic simulations (Allen and Tildesley 1989), should be used to detect dislocation segment intersections . In the algorithm, the cell partitions used for the FMM calculations, as shown in figure 3, can be reused. The minimum distance between unconnected segments must only be calculated for segments in the same or neighbouring cells. The shortest distance between segment pairs in non-neighbouring cells is guaranteed to be larger than r_{ann}. If these cells are too large, they can be subdivided into smaller cells to further reduce the number of $d_{\min}$ computations.

In ParaDiS, a collision procedure is performed on every unconnected segment pair that satisfies the condition $d_{\min} < r_{\text{ann}}$ using a combination of *Splitnode* and *Mergenode* operations. Upon solving $d_{\min}$, we automatically obtain the points of closest contact on these two segments: $\boldsymbol{x}_{ij}(\alpha_{\min})$ and $\boldsymbol{x}_{kl}(\beta_{\min})$, respectively. New nodes, P and Q, are then added on both segments at these points using the *Splitnode* operation on one of end nodes of each segment, as shown in figure 10(*a*). An exception is made if the point of closest contact overlaps with an end node of the segment. In this case, a new node is not added, and node P or Q corresponds to

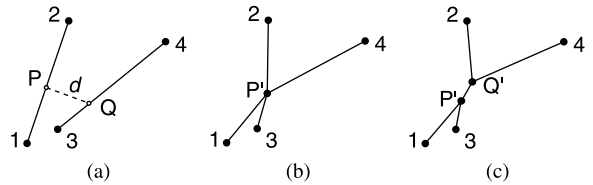

Figure 11. (*a*) and (*b*) are the same as those in figure 10. (*c*) Dissociation of node P' into two nodes, P' and Q', leading to a topology different from that in (*a*).

that existing end node. Nodes P and Q are then combined into a single node P' using the *Mergenode* operation, as shown in figure 10(*b*), completing the collision procedure.

Care must be taken in specifying the final position of the node P' at the end of the collision procedure. If intersecting segments $\boldsymbol{l}_{ij}$ and $\boldsymbol{l}_{kl}$ are confined to different glide planes, then the new node P' must be placed on the line contained in both glide planes, so that the new segments will satisfy the original glide constraints. Otherwise, the mobility of new segments could be adversely affected. An acceptable position $\boldsymbol{X}^{\text{coll}}$ to place the new node P' can be obtained by solving a constrained minimization problem that takes into account the mobility constraints of the intersecting segments. The problem takes the form

$$\min(\|\boldsymbol{X}^{\text{coll}} - \boldsymbol{X}_P\|^2 + \|\boldsymbol{X}^{\text{coll}} - \boldsymbol{X}_Q\|^2), \qquad \text{subject to } (\boldsymbol{X}^{\text{coll}} - \boldsymbol{X}_p) \cdot \boldsymbol{n}_{Pi} = 0,$$
$$(\boldsymbol{X}^{\text{coll}} - \boldsymbol{X}_Q) \cdot \boldsymbol{n}_{Ql} = 0.$$

where $\boldsymbol{n}_{Pi}$ are the glide plane normals of all segments connected to node P before the merge and $\boldsymbol{n}_{Ql}$ are the glide plane normals of all segments connected to node Q before the *Mergenode* operation.

Dislocation annihilation and junction forming reactions can be captured by performing a series of collision procedures on a pair of intersecting dislocation lines. Figure 10 shows a sequence of collisions leading to the formation of a dislocation junction. The first collision procedure, figures 10(*a*) and (*b*), involves two unconnected segments and results in a physical node P' connecting four segments. The second collision procedure, figures 10(*c*) and (*d*), involves two segments connected to a common segment through P' and results in two nodes P' and $2'$ that become the physical nodes at the ends of a junction dislocation. If segment intersections are detected and collision procedures are performed only for segments sharing the same Burgers vector, the DD simulation will perform annihilation reactions but not junction forming reactions.

A series of collision procedures can also capture junction unzipping. In this sequence, the two physical nodes at the ends of the junction dislocation come within r_{ann} leading to a collision procedure and the reformation of a node with four connections, as in figure 10(*b*). The second collision procedure is the same as the one depicted in figures 10(*c*) and (*d*) for the case when the Burgers vectors of the two intersecting segments sum up to zero. A potential alternative to this sequence of two collision procedures to capture junction formation and destruction is a sequence in which the second collision procedure is replaced by a dissociation procedure.

5.2. Dissociation procedures

For a physical n-node connecting $n > 3$ segments, it may be advantageous to reduce its connectivity by performing a dissociation procedure. In the dissociation procedure, the *Splitnode* operation is invoked to move at least two connections, but no more $n-2$ connections, from the n-node to a newly created node. Figure 11 depicts an alternative sequence of

topological procedures to treat a dislocation junction forming core reaction using a dissociation procedure. As before a collision procedure is used to create a physical 4-node P' at the point of intersection of two unconnected segments. Node P' then undergoes a dissociation procedure leading to the formation of a junction dislocation with two physical 3-nodes P' and Q' at its ends. A physical 4-node has three unique topological configurations that are possible end states of a dissociation procedure. A 5-node has ten possible end states. The dissociation procedure must decide which, if any, of these possible end states is preferred and must also decide where to position the resulting nodes after a dissociation procedure is performed.

In ParaDiS, the decision as to whether and how a physical n-node should be dissociated is made by a maximum power dissipation criterion. If the rate of energy dissipation is greatest with the n-node left intact, no dissociation procedure is performed. If, however, a dissociated configuration is found to dissipate the most power, the dissociation procedure is performed on the n-node to make the local topology conform to that configuration. The contribution of any one node i to the power dissipation is

$$\mathcal{P}_i^{\text{dis}} = \boldsymbol{F}_i \cdot \boldsymbol{V}_i. \tag{29}$$

Assuming that forces and velocities of other nodes in the simulation do not change if node i is dissociated into nodes P and Q, the change in the rate of energy dissipation due to the dissociation of the node is

$$\Delta\mathcal{P}_{i\to P,Q}^{\text{dis}} = \boldsymbol{F}_P \cdot \boldsymbol{V}_P + \boldsymbol{F}_Q \cdot \boldsymbol{V}_Q - \boldsymbol{F}_i \cdot \boldsymbol{V}_i. \tag{30}$$

In ParaDiS, every physical node connecting four or more segments is considered for a potential dissociation reaction, and for every node that satisfies the condition $\Delta\mathcal{P}_{i\to P,Q}^{\text{dis}} > 0$, a dissociation procedure is performed. If more than one dissociated configuration of a node satisfies the condition, the dissociated topology which dissipates the greatest power is selected.

Initial placement of the two new nodes P and Q resulting from the dissociation of node i requires some care. The simplest solution would be to leave them overlapping at the original position of the parent node. Unfortunately, they would satisfy the intersection criterion and recombine in the next time step. Also, the new nodes P and Q may be connected by a new segment to conserve the Burgers vector of the network. Allowing nodes P and Q to share the same position would lead to a segment with zero length and present numerical difficulties for subsequent nodal force and velocity calculations. Therefore, in ParaDiS, nodes P and Q are separated by a small distance r_{dis} to complete the dissociation procedure. The distance $r_{\text{dis}} = r_{\text{ann}} + \delta$, with $\delta > 0$, is chosen such that segments connected to the two nodes will not satisfy the intersection criterion. Specifically, we place the faster of the two nodes, P and Q, a distance of r_{dis} away from the location of the parent node in the direction of its motion and leave the other node at the original location, i.e.

$$\boldsymbol{X}_P = r_{\text{dis}} \frac{(1+\phi)\boldsymbol{V}_P}{2\|\boldsymbol{V}_P\|} + \boldsymbol{X}_i, \tag{31}$$

$$\boldsymbol{X}_Q = r_{\text{dis}} \frac{(1-\phi)\boldsymbol{V}_Q}{2\|\boldsymbol{V}_Q\|} + \boldsymbol{X}_i, \tag{32}$$

$$\phi = \text{sign}(\|\boldsymbol{V}_P\| - \|\boldsymbol{V}_Q\|). \tag{33}$$

In equations (31) and (32), the positions of the new nodes are computed based on their velocity when they are located at the original position of the parent node. However, after one of the nodes is displaced by r_{dis}, their forces and velocities may reverse leading to a configuration that satisfies the intersection criterion in the next time step. The dissociation would be undone by a collision procedure, and the original node would be reformed. To prevent this from occurring, a *confidence factor* is used to augment the dissociation criterion. The *confidence*

factor, c, is computed based on the velocities of both nodes P and Q at their displaced positions though the expression

$$c = \frac{(\boldsymbol{V}'_P - \boldsymbol{V}'_Q) \cdot (\boldsymbol{X}_P - \boldsymbol{X}_Q)}{\|\boldsymbol{V}'_P - \boldsymbol{V}'_Q\| \|\boldsymbol{X}_P - \boldsymbol{X}_Q\|}, \tag{34}$$

where $\boldsymbol{V}'_P$ and $\boldsymbol{V}'_Q$ are the velocities of the new nodes P and Q at $\boldsymbol{X}_P$ and $\boldsymbol{X}_Q$, respectively, as defined by equations (31) and (32). The *confidence factor* lies between the positive and the negative one. It returns a positive value if the two nodes continue to move apart and returns a negative value if the two nodes move back towards one another. Including this *confidence factor*, a more robust node dissociation criterion becomes

$$\text{Dissociate}(i \to \{P, Q\}) \equiv \max[c(\boldsymbol{F}_P \cdot \boldsymbol{V}_P + \boldsymbol{F}_Q \cdot \boldsymbol{V}_Q) - \boldsymbol{F}_i \cdot \boldsymbol{V}_i] > 0. \tag{35}$$

Dissociation procedures can be used to complete dislocation annihilation and junction formation and dissolution reactions. Once an intermediate crossed dislocation state is created by a collision procedure, the resultant physical 4-node connecting four segments can be treated using the dissociation procedure to change the topology. An annihilation reaction may result after the dissociation if all four segments share the same Burgers vector. If the four segments have two distinct Burgers vectors, a junction may be created if the dissociation procedure introduces a new segment or may be destroyed if the dissociation procedure does not introduce a new segment. Furthermore, the dissociation procedure is general enough to treat the dissolution of multi-nodes: physical nodes that connect four or more segments with more than two distinct Burgers vectors (Bulatov *et al* 2006).

5.3. Comparison of different topological treatments

Since there are multiple topology handling strategies employed by different DD algorithms to treat dislocation junction reactions, each option should be investigated for its ability to capture the physics of junction formation and dissolution and for its ability to enable large time steps to be taken by the time integration scheme. Option I is to perform no topological operations at all. Option II is to perform collision procedures only. Option III is to perform both collision and dissociation procedures to treat dislocation junction reactions.

Option I is the simplest. When two dislocations come together due to attractive elastic interactions, the dislocations are left to overlap with each other. Thus the junction dislocation is represented by two overlapping dislocation lines. Similarly, when the applied stress is large enough to unzip the junction, the two lines move apart, requiring no topological operations either. Options II and III are increasingly more complex. Both require detecting the intersections of unconnected dislocation segments, and Option III requires checking the stability of nodes with four or more connections. Furthermore, the properties of the junction dislocation formed by Options II and III may have different physical properties (Burgers vector, core energy, mobility) than the two overlapping parent dislocations that represent the junction dislocation in Option I. Option III is more computationally intensive than Option II which is more computationally intensive than Option I per time step; however, it may be more attractive if it enables larger timesteps to be taken, and if it enables a better description of junction physics.

The competing benefits of each of the three options can be evaluated by investigating how each handles the formation and dissolution of a simple binary junction between two $\langle 1\,1\,1 \rangle$-type dislocations in BCC crystals. A series of simulations focused on the behaviour of a 'short' junction in which the maximum stress required to break the junction was reached when the two parent dislocation lines had unzipped the junction and were in a crossed configuration. A

second series focused on a 'long' junction in which the maximum stress required to break the junction was reached while configuration still contained a dislocation junction of finite length. The configurations were loaded with a constant strain rate in a direction that equally loaded both parent dislocations, and the maximum stress reached to break the junction was recorded as a function of mesh refinement.

Since the $\langle 1\,0\,0\rangle$-type junction dislocation formed in the simulations using Options II and III has a different Burgers vector than the two $\langle 1\,1\,1\rangle$-type parent dislocations, its mobility and core energy must be specified separately. To be consistent with the physical limits of Option I, the segments with a $\langle 1\,0\,0\rangle$-type Burgers vector are constrained not to move in directions perpendicular to their line direction, and the core energy of the $\langle 1\,0\,0\rangle$-type is set to be $\epsilon^{\mathrm{c}}_{\langle 1\,0\,0\rangle} = 2\,\epsilon^{\mathrm{c}}_{\langle 1\,1\,1\rangle}$. This core energy of the $\langle 1\,0\,0\rangle$-type dislocation is the effective core energy of the junction under Option I because the junction is represented by overlapping $\langle 1\,1\,1\rangle$-type segments. However, realizing that the dislocation core energy is a material parameter in DD simulations that must ultimately be constructed from atomistic input, a second set of simulations are conducted for Options II and III with $\epsilon^{\mathrm{c}}_{\langle 1\,0\,0\rangle} = \frac{4}{3}\,\epsilon^{\mathrm{c}}_{\langle 1\,1\,1\rangle}$. Without atomistic input, the latter ratio of core energies may be more natural. It assumes that the core energy of a segment is proportional to the square of its Burgers vector. With these two different ratios, the influence of the core energy on the junction strength can be evaluated.

Figure 12 shows the critical stress to break the junction as a function of the fineness of the discretization $L/L_{\max}$ for approaches I, II and III. The length L is the initial length of the two parent dislocations which remains fixed for all the simulations, and $L_{\max}$ is the maximum segment length allowed by the discretization. As the discretization becomes finer, the results of all three options converge, but not necessarily to the same value. Option I appears to converge most rapidly, while Option II appears to be the last to converge. Option I (no topological operations, blue dashed line) predicts the lowest junction strengths, and Option II (collision procedures only, green lines) predicts the highest junction strengths. The response of Option III (both collision and dissociation procedures, red lines) is bound by the response of Options I and II. For the cases when the core energy ratio was set as $\epsilon^{\mathrm{c}}_{\langle 1\,0\,0\rangle} = 2\,\epsilon^{\mathrm{c}}_{\langle 1\,1\,1\rangle}$, Option II yields a nonphysical trend in which the 'short' junction requires a larger breaking stress than the 'long' junction. The trend that the 'long' junctions require a larger breaking stress than 'short' junctions is followed for all other cases. This nonphysical result yielded by Option II occurs because the crossed configuration endures until the two segments on a parent dislocation bow around enough to satisfy the intersection criterion, and a collision procedure is performed. The procedure results in an annihilation reaction involving the physical 4-node, and the dislocation lines are unbound from the crossed state. Therefore, only Options I and III should be considered as candidate strategies for treating dislocation junction reactions.

Figure 12 also shows that the ratio between the core energies of the parent dislocations and the junction dislocation may have a strong influence on the junction strength. Although the junction strength will also depend on the absolute magnitude of the core energies, the strength's dependence on this ratio serves as an illuminating exercise. For a relative reduction of 30% in the core energy of the junction dislocation, the strength of the 'long' junction increases by 57%, and the strength of the 'short' junction increases by 118% when applying the topological treatments of Option III. This suggests that dislocation core energies can have a significant effect on the strain hardening characteristics in a large scale DD simulation. Therefore, on a purely physical basis, dislocation core reactions leading to junctions should be treated using the full topological handling (collision and dissociation) procedures developed in the previous subsections because it allows the user to specify the physical properties (core energy and mobility) of junction dislocations.

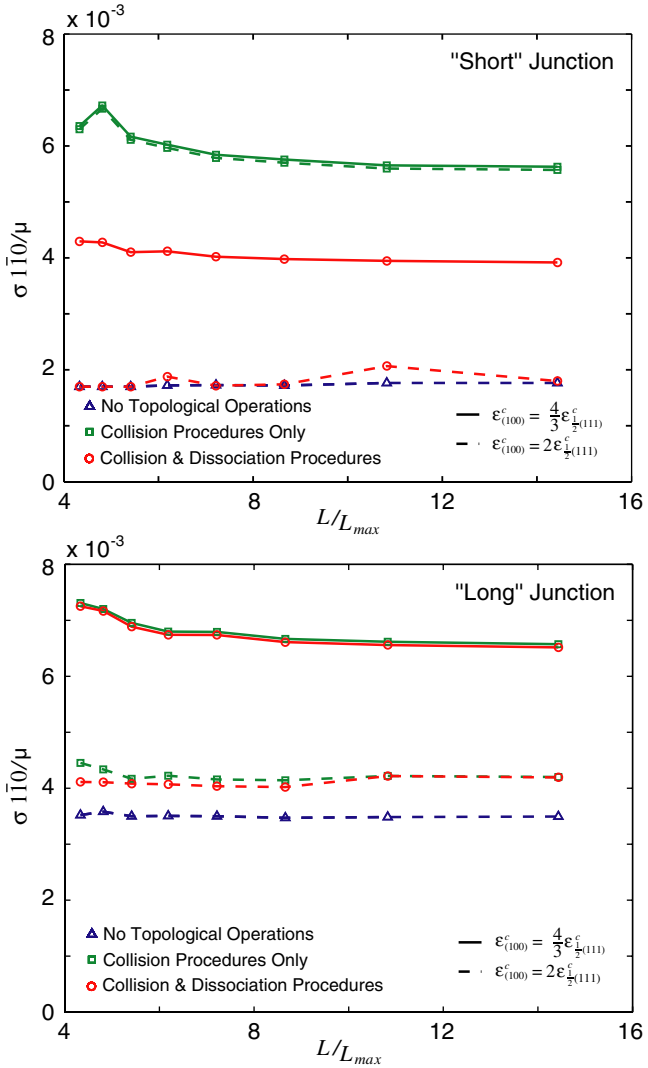

Figure 12. Simulated strength of binary junction as a function of discretization ratio $L/L_{\max}$, topology handling strategies and core energy. Three topology handling options were explored: no topological operations, collision procedures only and both collision and dissociation procedures enabled. Two core energies were explored for the junction dislocation. The 'short' junction geometry consisted of two dislocations of length $L = 1000$ lattice spacings with $\boldsymbol{b}_1 = \frac{1}{2}[\bar{1}\,1\,1]$, $\boldsymbol{t}_1 = \frac{1}{\sqrt{33}}[\bar{1}\,4\,4]$ and $\boldsymbol{b}_2 = \frac{1}{2}[1\,\bar{1}\,1]$, $\boldsymbol{t}_2 = \frac{1}{\sqrt{33}}[4\,\bar{1}\,4]$ intersecting at their midpoints. The 'long' junction geometry consisted of two dislocations with $\boldsymbol{b}_1 = \frac{1}{2}[\bar{1}\,1\,1]$, $\boldsymbol{t}_1 = \frac{1}{\sqrt{33}}[1\,4\,4]$ and $\boldsymbol{b}_2 = \frac{1}{2}[1\,\bar{1}\,1]$, $\boldsymbol{t}_2 = \frac{1}{\sqrt{33}}[4\,1\,4]$ intersecting at their midpoints.

The different strategies to handle dislocation core reactions will also influence the performance of the time integration algorithm. In choosing between Options I and III, it is also desirable to select the strategy that enables the largest time steps to be taken. In section 3, we described a time integration algorithm used in ParaDiS, in which the size of the time step is automatically adjusted at each cycle to satisfy a given numerical accuracy. Figure 13 shows the time step history enabled by this time integration scheme while simulating the 'short'

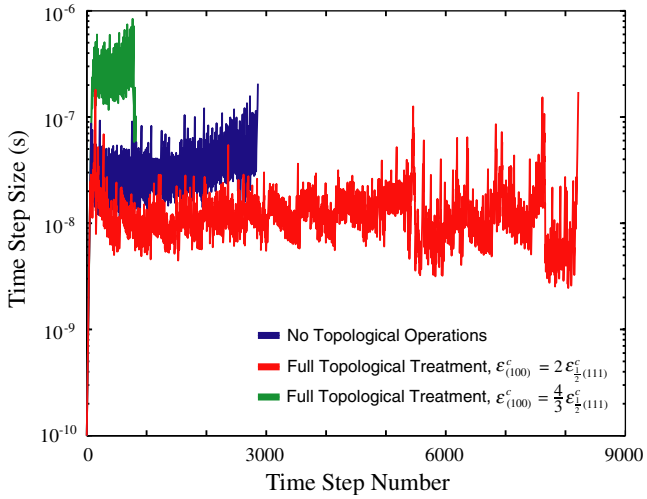

Figure 13. Time step history for the junction formation and annihilation simulations described shown in figure 12 for a 'short' junction with $L/L_{\max} = 6$ under a constant applied strain rate. The case in which collision procedures are performed only is omitted, and a case in which full topological procedures are performed is added with the core energy of the $\langle 1\,0\,0\rangle$ junction dislocation equal to 4/3 of the core energy of the two $\frac{1}{2}\langle 1\,1\,1\rangle$ dislocations.

junction from initial creation to final dissolution using Option I and Option III with the two different core energy ratios. Interestingly, enabling topological operations in Option III (with $\epsilon^{\mathrm{c}}_{\langle 1\,0\,0\rangle} = 2\,\epsilon^{\mathrm{c}}_{\langle 1\,1\,1\rangle}$) reduces the average time step by 1/3 compared with Option I. Reducing the junction core energy to $\epsilon^{\mathrm{c}}_{\langle 1\,0\,0\rangle} = \frac{4}{3}\,\epsilon^{\mathrm{c}}_{\langle 1\,1\,1\rangle}$ increases the average time step by an order of magnitude. Detecting dislocation intersections and probing potential modes of dissociation for highly connected nodes do not add an order of magnitude to the computational expense of the DD algorithm. Option III may offer numerical advantages in addition to a more accurate description of underlying physics.

6. Parallel algorithms

The computational expense of three-dimensional DD simulations is so great that parallel computers are required to simulate all but the smallest dislocation networks. For example, according to figure 6, a DD simulation with 10^5 segments would require a parallel computer with approximately 400 central processor units (CPUs) to keep the nodal force calculation per time step below 10 s. In fact, any simulation containing more than 200 segments can benefit from parallel computation.

The difficulty in implementing a DD simulation on parallel computer architectures is that the nodal degrees of freedom in the network have a tendency to cluster in space. The clustering leads to a condition where the computational expense of calculating forces on different nodal degrees of freedom can significantly differ due to the neighbourhood dependence on the number of local segment–segment force interactions that must be evaluated. Under such conditions, a simple partitioning algorithm, that divides the simulation domain into subdomains of equal volume and assigns the calculation of forces for the active degrees of freedom (DOF) within different subdomains to different CPUs, fails spectacularly. We have found that the fraction of time spent performing computations, η, is typically under 10% with this simple spatial

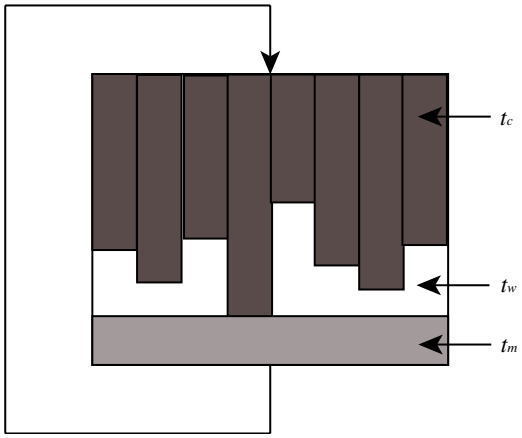

Figure 14. Schematic depicting the performance of a simple parallel program with a single synchronization point where t_c is the time spent computing, t_w is the time spent waiting and t_m is the time spent communicating.

partitioning scheme. Similarly poor results are obtained from a partitioning algorithm that attempts to distribute an equal number of DOF (and the responsibility of calculating their forces) to each CPU. More powerful dynamic load balancing schemes must be considered to fully exploit the processing power of parallel computer architectures.

In developing a good load balancing scheme, we will attempt to optimize $\eta \equiv t_c/(t_c + t_w + t_m)$ where t_c, t_w and t_m is the cumulative time that the parallel computer spends in performing computations, in waiting for CPUs to reach synchronization points and in inter-processor communications (messaging), respectively. Figure 14 illustrates the relationship between these three times in a simple parallel calculation with a single synchronization point. Optimization of η is best achieved through minimization of t_w and t_m. The time spent waiting is minimized by evenly partitioning the total computation across all the available CPUs, and t_m is minimized by using inter-processor communication patterns that rely on local, point-to-point, messaging exclusively. Since the nodal force calculation is usually the most computationally intensive part of each time step, we will focus on algorithms that optimally distribute this part of the computation and neglect other parts of a DD cycle, such as the detection of dislocation collisions and core reactions.

The $\mathcal{O}(n)$ implementation of the nodal force calculation using FMM creates a spatial locality in the data required to calculate the nodal forces. Dislocation segments in non-neighbouring FMM subcells do not need to know each others' positions and connectivity. Under such circumstances, an advanced domain decomposition partitioning algorithm is anticipated to perform well. In this more advanced spatial partitioning algorithm, the responsibility of maintaining accurate information and calculating the forces for the DOF within different subdomains of *unequal* volume is assigned to different CPUs, and the boundaries of subdomains are allowed to shift in time. The additional computation required to dynamically adjust the position of the subdomain boundaries is small compared with the wall clock time saved in evenly distributing the computational load, and each processor needs to communicate with only a few 'neighbouring' CPUs and not the whole agglomeration.

Currently in the ParaDiS code, the hierarchical decomposition scheme shown in figure 15 is used. The entire simulation volume is first divided into N_x slabs along the x direction. Each slab is then divided into N_y columns along the y direction. Each column is then divided into N_z

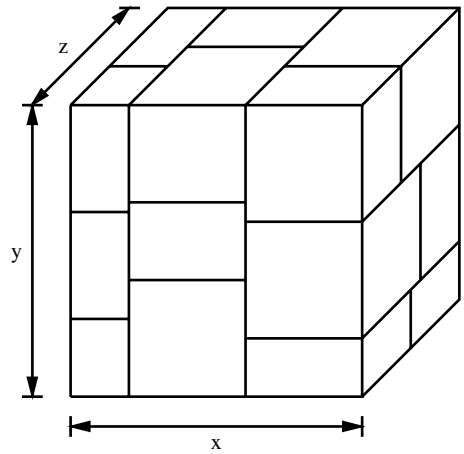

Figure 15. Schematic depicts the dynamic domain decomposition algorithm used to partition the computation on parallel computer architectures. The volume is recursively sectioned by first cutting the volume with planes parallel to the x-axis, then with planes parallel to the y-axis, followed by planes parallel to the z-axis.

subdomains along the z direction. The total number of subdomains is chosen to be equal to the number of available CPUs, and the positions of the slab, column and subdomain boundaries are periodically adjusted to balance the computation based on the relative wall clock time each CPU spends computing versus waiting. Since the nodal force calculation dominates the computational expense of a time step iteration, this part will be aggressively balanced at the expense of creating load imbalances in other parts of time step iteration.

The data and operation ownership rules are more complicated in DD simulations than in molecular dynamics simulations because the elements of the DD force calculation consist of one-dimensional objects (segments), as opposed to point objects, that may cross the FMM and subdomain boundaries. Although the ownership rules are somewhat arbitrary, the specified set of rules must consistently ensure that every element of the computation is uniquely distributed across the CPUs with common precedents. All dislocation segments must be owned by an FMM subcell for the purpose of building the multipole moments of FMM hierarchy at its lowest level, and no two FMM subcells should claim ownership of the same segment. In the ParaDiS code, a dislocation segment is owned by the FMM subcell with the highest index that contains one of its end nodes, unless the segment crosses the periodic boundary, in which case the segment is owned by the subcell with the lowest index that contains one of its end nodes. Although this specified rule is by no means the only manner to set ownership, it does satisfy the data and operation exclusivity criteria. Similar rules must be established to distribute the simulation data and computational operations onto the CPUs that will perform the force calculation.

In the ParaDiS code, the CPU owns the nodal degrees of freedom whose positions are contained within its subdomain. Node ownership entails that the CPU will maintain and update the force, position and velocity for its nodes and send that data to other CPUs required to perform some computations involving its nodes. The segment ownership rules are a bit more complicated. In the ParaDiS code, a segment is owned by the CPU with the lowest index whose domain both intersects the FMM subcell that owns that segment and contains one of the segment's end nodes. Segment ownership entails that the CPU will maintain and update the segment's Burgers vector and the IDs of its end nodes. The CPU is also

responsible for calculating the forces on the segment's end nodes due to their dislocation core and elastic self-energies and their interaction with the far field stresses (contained in the Taylor series expansions of their FMM subcells). Lastly, the CPU is responsible for calculating the contribution of its segments to the multipole moments of the FMM subcells that own them.

The most intensive part of the force computation that must be evenly distributed across parallel computers is the calculation of the interaction forces between pairs of segments in the same or neighbouring FMM subcells and the calculation of the Taylor series expansion of the stress field in a subcell due to multipole moments of dislocation segments in non-neighbouring subcells. The analytical expressions in appendix A are used to simultaneously compute the interaction forces on all 4 ends nodes of two interaction segments within neighbouring FMM cells. In the ParaDiS code, this computation is owned by the CPU with the lowest index in the FMM cell with the lowest index that owns one of the interacting segments.

The most time-consuming part of the fast multipole algorithm is computing the Taylor series expansion of the stress field in one FMM cell due to the multipole expansion of another cell. Since the cost and structure of this calculation is fixed for a given hierarchy of FMM cells, the computational load can be statically distributed at the beginning of the problem. The distribution remains unchanged at every DD time step until the FMM hierarchy or the number of available CPUs changes. Let the total number of FMM cells (of the entire hierarchy) be N_{m}, and let the total number of CPUs be P. By assigning the elements of an ordered list of all multipole cells in the hierarchy to an ordered list of CPUs, a load distribution is achieved in which each CPU calculates the Taylor series expansion of stress field inside $\sim(N_{\mathrm{m}}/P)$ number of cells. Fast multipole cell ownership also includes the computations involved in the upward and downward pass translations, but the parallel distribution of these parts of the computation is less critical because of their relative low cost.

For consistency, all operations in a DD algorithm should respect the data ownership specified. This includes not only force calculations but also integrating nodal equations of motion, adaptive mesh refinement and the treatment of core reactions. Only the CPU that owns a node may update its position or delete it altogether. Only the CPU that owns a segment may change its Burgers vector, the node IDs of its ends nodes, or delete it altogether. Therefore, a CPU may perform a topological operation only if it owns all the data that will be affected by this change. As a result, all topological operations that would have been performed in a serial simulation may not be performed in a parallel simulation if doing so would violate the specified ownership rules. To prevent bad dislocation configurations from enduring in parallel computations, the ownership rules for topological operations are inverted every other time step.

6.1. Force calculation flow chart

The distributed ownership of data and computational work across different CPUs determines the pattern and frequency of communications between them. The total amount of communication per time step should be kept small for an efficient parallel algorithm. The ownership rules outlined above require only local, point-to-point, communications for the calculation of forces on all the nodes. In this instance, the combined inter-processor communications and intra-processor operations required to compute nodal forces fit the following logical pattern.

1. Each CPU communicates the position and connectivity of the nodes that it owns to CPUs whose domains intersect same and/or neighbouring fast multipole subcells.
2. *Construct multipole moments*. Each CPU calculates the contribution of its dislocation segments to the multipole moments of the FMM subcell to which these segments belong

and communicates these contributions to the CPU that owns the FMM subcell. Each CPU that owns a FMM subcell adds all the contributions together.

3. *Upward pass*. Starting from the bottom of the fast multipole hierarchy, each CPU collects and sums the contribution to the multipole moments of the FMM subcells it owns from its eight daughter cells and then calculates the upward pass translation of its multipole moments and communicates the result to the CPU that owns the subcell's parent, until the top of the hierarchy is reached.
4. *Transverse translation*. The multipole moments from 189 cells that are outside the nearest neighbour distance of the target cell but inside the nearest neighbour distance of its parent are collected by the CPU that owns the target cell and their contribution to the Taylor series expansion of the stress field in target is calculated.
5. *PBC correction*. The CPU that owns the FMM cell at the highest level of the hierarchy calculates the Taylor series expansion of the stress state due to the periodic images of the system and adds that to the externally applied stress.
6. *Downward pass*. Starting at the highest level of the FMM hierarchy, each CPU that owns a subcell sums the contribution from its parent to its Taylor series expansion of the stress from Step 4 and then calculates the downward pass translation of the stress for each one of its daughter cells and sends the result to the CPUs that own the daughter cells until the bottom of the hierarchy is reached.
7. Each CPU that owns a lowest level subcell in the fast multipole hierarchy communicates the Taylor series expansions of the stress field to the CPUs whose domains intersect the subcell.
8. $O(n)$ *force computation*. Each CPU calculates and combines the contribution to the nodal force from the Taylor series expansion of the stress field and from the elastic and core self-energy contributions for the segments that it owns.
9. $O(n^2)$ *force computation*. Each CPU calculates and combines the contribution to the nodal forces from local segment–segment interactions for the interaction pairs that it owns.
10. Each CPU communicates the nodal force information for nodes it does not own to the appropriate CPUs and adds the force information for its nodes that it receives from other CPUs. Force calculation ends.

The most computationally intensive parts of the above algorithm are in Steps 4 and 9. The work load of Step 4 is distributed at the beginning of the simulation, and it remains unchanged as long as the FMM hierarchy and number of CPUs remain constant. However, the work load of explicit segment–segment calculations in Step 9 must be distributed dynamically by adjusting the boundaries of the domains. In ParaDiS, this is done by measuring the time it takes for each CPU (domain) to perform computation in Steps 8 and 9 of every cycle. The boundary between two domains is then shifted towards the domain that has a relatively higher computational time. The amount of shifting is computed based on the estimated computational time per unit volume in each domain. Because of the hierarchical decomposition scheme, this procedure is applied to adjust the boundary between neighbouring domains in the same column, as well as the boundary between neighbouring columns in the same slab and the boundary between neighbouring slabs. The performance of this load balancing algorithm will be assessed in the next section.

7. Evaluation of ParaDiS code performance

A DD simulation using the nodal force calculation described in section 2, the time integration procedure described in section 3, the adaptive mesh refinement strategy described in

Table 1. (*a*) Material parameters used to capture the properties of bcc molybdenum at elevated temperatures. (*b*) ParaDiS runtime parameters for the demonstration simulation.

(a) Material parameters
$\mu = 130\,\text{GPa}$
$\nu = 0.309$
$\|\boldsymbol{b}\|_{\langle 1\,1\,1\rangle} = 2.725 \times 10^{-10}\,\text{m}$
$a = 40\|\boldsymbol{b}\|_{\langle 1\,1\,1\rangle}$
$\epsilon_c = \dfrac{\mu}{4(1-\nu)\pi} \ln[4\,0\,0](\|\boldsymbol{b}\|^2 - \nu\|\boldsymbol{b}\cdot\boldsymbol{t}\|^2)$
$B_{\text{eg}}^{\langle 1\,1\,1\rangle} = B_{\text{s}}^{\langle 1\,1\,1\rangle} = 1\,\text{Pa}\,\text{s}^{-1}$
$B_{\text{ec}}^{\langle 1\,1\,1\rangle} = B_{\text{ec}}^{\langle 1\,0\,0\rangle} = 1000\,\text{Pa}\,\text{s}^{-1}$
$B_{\text{eg}}^{\langle 1\,0\,0\rangle} = B_{\text{s}}^{\langle 1\,0\,0\rangle} = 1000\,\text{Pa}\,\text{s}^{-1}$
$B_{l}^{\langle 1\,1\,1\rangle} = B_{l}^{\langle 1\,0\,0\rangle} = 0.001\,\text{Pa}\,\text{s}^{-1}$
(b) ParaDiS runtime parameters
$r_{\text{tol}} = 10\|\boldsymbol{b}\|_{\langle 1\,1\,1\rangle}$
$r_{\text{ann}} = 2r_{\text{tol}}$
$r_{\text{dis}} = 2r_{\text{ann}}$
$L_{\text{min}} = 50\|\boldsymbol{b}\|_{\langle 1\,1\,1\rangle}$
$L_{\text{max}} = 200\|\boldsymbol{b}\|_{\langle 1\,1\,1\rangle}$
$A_{\text{min}} = 2r_{\text{tol}}L_{\text{max}}$
$A_{\text{max}} = 2A_{\text{min}} + \dfrac{\sqrt{3}}{8}L_{\text{max}}^2$
$\Delta t_{\text{max}} = 10^{-7}\,\text{s}$
$d = 1.2$
$m = 1$

section 4, the core reaction procedures described in section 5 and the dynamic load balancing algorithm described in section 6 were performed to evaluate the efficacy of these algorithms in enabling the direct simulation of strain hardening. Therefore, a thorough analysis of the numerical performance and not the physical results of the simulation will be conducted. The material properties and boundary conditions of the simulation are given for reference and to aid in comparing the performance of ParaDiS with other DD simulation codes.

The material properties were chosen to capture the behaviour of bcc molybdenum single crystals at elevated temperatures. The mobility of the dislocations is described in full detail in appendix C, and a list of the material constants is given in table 1. The spatial approximation discussed in section 3 was used to simplify the solution of the nodal velocities from the nodal forces. The simulation cell was $5\,\mu$m on a side with periodic boundary conditions applied. An initial dislocation density of $1.8 \times 10^{12}\,\text{m}^{-2}$ was seeded by randomly placing screw dislocations that pierced the periodic boundaries of the cell, and the deformation was driven by a constant tensile strain rate of $10\,\text{s}^{-1}$ applied along the [1 0 0] crystallographic axis. All the other discretization and runtime parameters used in the simulation that have been described in the previous sections are also listed in table 1.

The simulation was performed on 128 nodes of the Thunder computer at the Lawrence Livermore National Laboratory. A node of the Thunder machine consists of four Intel Itanium II 1.4 GHz CPUs with 8 GB of shared RAM per node. As shown in the stress–strain response of figure 16, a total tensile strain of approximately 1.7% was achieved in approximately 2×10^5 CPU-hours. The tensile behaviour of the material exhibits an initial upper yield point due to the low initial dislocation density and the simple linear mobility law. If the initial dislocation density is chosen such that it corresponds to the density ($10^{13}\,\text{m}^{-2}$) when the minimum in the yield stress is reached, an upper yield point is not observed. During the simulation, there was

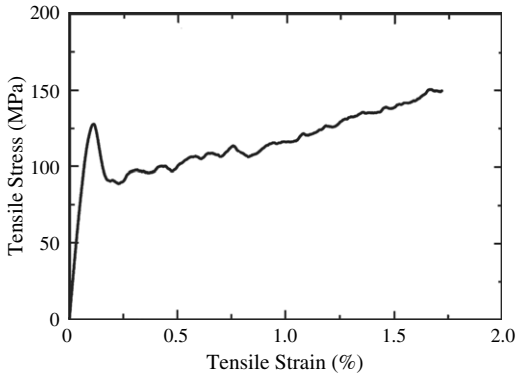

Figure 16. A representative tensile stress–strain response for a simulation of a molybdenum single crystal with the loading direction along the [1 0 0] crystallographic axis using the analytical forces and mobility law described in the appendix.

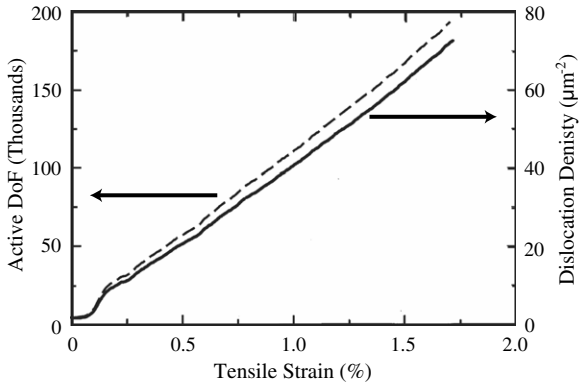

Figure 17. Growth of the dislocation density and the active degrees of freedom in a strain hardening simulation as a function of the accumulated deformation.

approximately a 40× increase in the length of dislocation lines within the simulation cell as shown in figure 17. The growth in the dislocation density led to a proportional growth in the active DOF from approximately 5×10^3 to 1.9×10^5. Therefore, the number of CPUs was increased proportionally as well.

The number of processors was doubled every time the average number DOF per CPU reached 400, so that the average number of active DOF per CPU always ranged between 200 and 400 during the course of the simulation. The simulation initially started on 16 CPUs and grew until it fully loaded 512 CPUs requiring an increase to 1024 CPUs at which point the simulation was terminated. The average size of a time step in the simulation was 3.35×10^{-9} s, and a little over half a million time step iterations were required to reach the final strain of 1.7%. The simulation required 17 wall clock days to execute, with each time step iteration taking approximately 2.9 wall clock seconds.

The strong scaling characteristics of the ParaDiS code are shown for two different numbers of active degrees of freedom in figure 18. In this plot, perfect scaling would yield a straight line from the bottom left corner to the top right corner. While the force calculation continues to show good strong scaling characteristics for fewer than 200 active DOF per CPU, the force

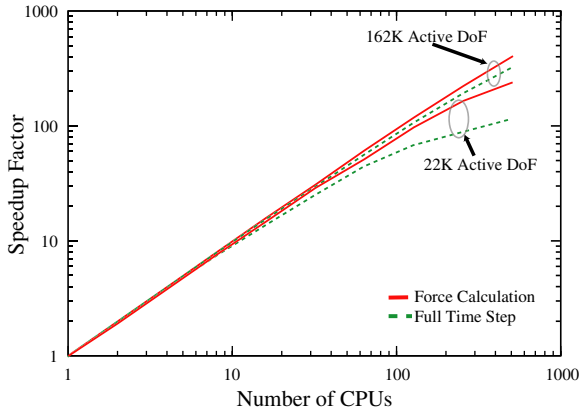

Figure 18. Strong scaling characteristics of the ParaDiS code on the LLNL/thunder machine for two snapshots of a strain hardening simulation containing 22 and 162 K active degrees of freedom.

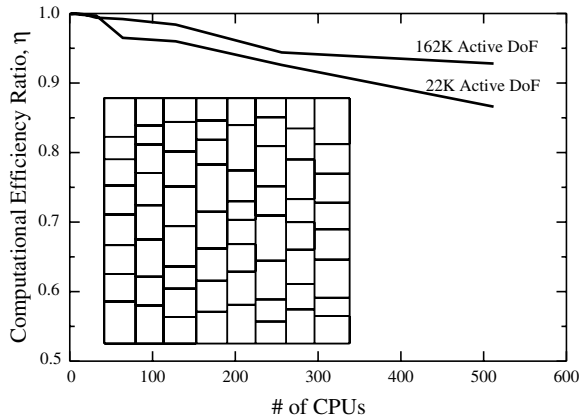

Figure 19. The ratio, η, of CPU time spent performing calculations over the total CPU time spent in computing nodal forces for two snapshots of a strain hardening simulation containing 22 and 162 K active degrees of freedom. Inset: position of domain boundaries obtained by the dynamic load balancing algorithm with 512 CPUs for a slice of the simulation cell taken along the y-axis.

calculation no longer dominates the time spent in the time step iteration, and the other parts of the time step iteration that have not been optimally parallelized begin to strongly affect the scaling of the code.

The reduction in perfect scaling of the force calculation is directly attributable to the residual load imbalance of the dynamic load balancing algorithm. Figure 19 shows the fraction of time spent performing computations as a function of the number of CPUs for two different timesteps in the simulation. The parallel efficiency degrades with increasing number of processors for both cases and degrades more rapidly for the configuration with fewer degrees of freedom. However, ParaDiS is still able to achieve parallel efficiencies better than 85% in the force calculation even when the average number DOF per CPU reaches 40. In the 200–400 range that is maintained during the simulation, the parallel efficiency in the force calculation is always better than 90%. Figure 19 also shows the position of the domain boundaries for a slice of the simulation box which are used for the simulation step with 512

CPUs handling 162 000 active DOF. We find that this particular domain decomposition routine behaves well for simulations up to several thousand CPUs but begins to degrade as the number of CPUs reaches on the order of 10 000. Other hierarchical domain decomposition schemes such as binary space partition algorithms may be considered in the future to better distribute the calculation across computers with tens of thousands of CPUs.

Significant gains in the average size of a simulation time step may also be possible by constructing better implicit algorithms for the time integration methods. In its current form, the implicit time integration scheme in the ParaDiS code does not use any information about the stiffness matrix that may be easily accessible to update the positions of the active DOF. Such information may be able to significantly increase the average time step size in these simulations and make it possible to perform strain hardening simulations with DD at quasi-static strain rates ($10^{-3}\,\mathrm{s}^{-1}$). As implemented, it is difficult to accumulate any appreciable deformation at strain rates below $1\,\mathrm{s}^{-1}$ in a reasonable amount of wall clock time.

Acknowledgments

This work was performed under the auspices of the US Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No W-7405-Eng-48.

Appendix A. Calculation of force on the end node of a dislocation segment

The following subsections detail the force on an end node of a dislocation segment due to the stress field of another dislocation segment. The stress field of a dislocation segment is derived from the non-singular treatment developed by Cai *et al* (2006) for the special case of isotropic elastic solids. The first subsection gives the force on the end node of a dislocation segment due to another non-parallel dislocation segment. The second subsection gives the force on the end node of a dislocation segment due to another parallel segment, and the third subsection gives the force on the end node of a dislocation segment due to its own stress field. The double integral equation for the force $\boldsymbol{f}_{43}^{12}$ at a node located at $\boldsymbol{x}_4$ that terminates a dislocation segment that begins at $\boldsymbol{x}_3$ with Burgers vector $\boldsymbol{b}$ due to the stress field $\boldsymbol{\sigma}^{12}$ of a dislocation segment with Burgers vector $\boldsymbol{b}'$ that begins at $\boldsymbol{x}_1$ and terminates at $\boldsymbol{x}_2$ is given by

$$\boldsymbol{f}_{43}^{12} = \int_0^L \frac{l}{L}\left\{\left[\boldsymbol{\sigma}^{12}\left(\left(1-\frac{l}{L}\right)\boldsymbol{x}_3 + \frac{l}{L}\boldsymbol{x}_4\right)\cdot\boldsymbol{b}\right]\times\boldsymbol{t}\right\}\,\mathrm{d}l,$$

$$\begin{aligned}\boldsymbol{\sigma}^{12}(\boldsymbol{x}) = &-\frac{\mu}{8\pi}\int_{\boldsymbol{x}_1}^{\boldsymbol{x}_2}\left(\frac{2}{R_a^3}+\frac{3a^2}{R_a^5}\right)[(\boldsymbol{R}\times\boldsymbol{b}')\otimes\,\mathrm{d}\boldsymbol{x}' + \mathrm{d}\boldsymbol{x}'\otimes(\boldsymbol{R}\times\boldsymbol{b}')]\\ &+\frac{\mu}{4\pi(1-\nu)}\int_{\boldsymbol{x}_1}^{\boldsymbol{x}_2}\left(\frac{1}{R_a^3}+\frac{3a^2}{R_a^5}\right)[(\boldsymbol{R}\times\boldsymbol{b}')\cdot\,\mathrm{d}\boldsymbol{x}']\boldsymbol{I}_2\\ &-\frac{\mu}{4\pi(1-\nu)}\int_{\boldsymbol{x}_1}^{\boldsymbol{x}_2}\frac{1}{R_a^3}((\boldsymbol{b}'\times\mathrm{d}\boldsymbol{x}')\otimes\boldsymbol{R}+\boldsymbol{R}\otimes(\boldsymbol{b}'\times\mathrm{d}\boldsymbol{x}'))\\ &+\frac{\mu}{4\pi(1-\nu)}\int_{\boldsymbol{x}_1}^{\boldsymbol{x}_2}\frac{3}{R_a^5}[(\boldsymbol{R}\times\boldsymbol{b}')\cdot\mathrm{d}\boldsymbol{x}'\boldsymbol{R}\otimes\boldsymbol{R}],\end{aligned}$$

where

$$R = \boldsymbol{x}-\boldsymbol{x}' \qquad R_a = \sqrt{\boldsymbol{R}\cdot\boldsymbol{R}+a^2},$$
$$L = \|\boldsymbol{x}_4-\boldsymbol{x}_3\| \qquad \boldsymbol{t} = \frac{\boldsymbol{x}_4-\boldsymbol{x}_3}{L}.$$

The variable $\boldsymbol{x}$ spans the segment $\boldsymbol{x}_3 \rightarrow \boldsymbol{x}_4$ and the variable $\boldsymbol{x}'$ spans the segment $\boldsymbol{x}_1 \rightarrow \boldsymbol{x}_2$. The integral equations holds for isotropic elastic solids whose elastic properties are described by their shear modulus μ and the Poisson ratio ν. This non-singular form also incorporates a third material constant a whose length describes the width of the dislocation core.

Appendix A.1. Force on the end node of a segment due to another non-parallel segment

The solution integral equation for the force $\boldsymbol{f}_{43}^{12}$ is described by the following expression:

$$\boldsymbol{f}_{43}^{12} = \frac{\mu}{4\pi(1-\nu)L}[\tilde{\boldsymbol{f}}(\boldsymbol{x}_4-\boldsymbol{x}_2) - \tilde{\boldsymbol{f}}(\boldsymbol{x}_3-\boldsymbol{x}_2) - \tilde{\boldsymbol{f}}(\boldsymbol{x}_4-\boldsymbol{x}_1) + \tilde{\boldsymbol{f}}(\boldsymbol{x}_3-\boldsymbol{x}_1)],$$

$$\begin{aligned}
\tilde{\boldsymbol{f}}(\boldsymbol{R}) = d\{&(1-\nu)[\boldsymbol{u}\cdot(\boldsymbol{b}'\times\boldsymbol{b})]\boldsymbol{u} - (1-\nu)(\boldsymbol{t}\cdot\boldsymbol{b}')(\boldsymbol{t}'\cdot\boldsymbol{b})\boldsymbol{u} + (\boldsymbol{v}'\cdot\boldsymbol{b}')(\boldsymbol{t}\times\boldsymbol{b}) \\
&+[\boldsymbol{t}\times(\boldsymbol{t}'\times\boldsymbol{b}')](\boldsymbol{u}\cdot\boldsymbol{b}) - [\boldsymbol{t}'\cdot(\boldsymbol{b}'\times\boldsymbol{b})]\boldsymbol{v}\}(I_{103} - y_3 I_{003}) \\
&+\{(1-\nu)[\boldsymbol{t}\cdot(\boldsymbol{b}'\times\boldsymbol{b})]\boldsymbol{u} - (1-\nu)[\boldsymbol{t}\times(\boldsymbol{t}\times\boldsymbol{b}')](\boldsymbol{t}'\cdot\boldsymbol{b}) \\
&+[\boldsymbol{t}\times(\boldsymbol{t}'\times\boldsymbol{b}')](\boldsymbol{t}\cdot\boldsymbol{b}) - (\boldsymbol{u}\cdot\boldsymbol{b}')(\boldsymbol{t}\times\boldsymbol{b})\}(I_{203} - y_3 I_{103}) \\
&+\{(2-\nu)[\boldsymbol{t}'\cdot(\boldsymbol{b}'\times\boldsymbol{b})]\boldsymbol{u} + \nu[\boldsymbol{t}\times(\boldsymbol{t}'\times\boldsymbol{b}')](\boldsymbol{t}'\cdot\boldsymbol{b})\}(I_{113} - y_3 I_{013}) \\
&+3d\frac{a^2(1-\nu)}{2}\{[\boldsymbol{u}\cdot(\boldsymbol{b}'\times\boldsymbol{b})] - (\boldsymbol{t}\cdot\boldsymbol{b}')(\boldsymbol{t}'\cdot\boldsymbol{b})\}\boldsymbol{u}(I_{105} - y_3 I_{005}) \\
&+3d\{a^2(\boldsymbol{v}'\cdot\boldsymbol{b}')(\boldsymbol{t}\times\boldsymbol{b}) - d^2(\boldsymbol{v}'\cdot\boldsymbol{b}')(\boldsymbol{u}\cdot\boldsymbol{b})\boldsymbol{v}\}(I_{105} - y_3 I_{005}) \\
&+\frac{3a^2(1-\nu)}{2}\{[\boldsymbol{t}\cdot(\boldsymbol{b}'\times\boldsymbol{b})]\boldsymbol{u} - [\boldsymbol{t}\times(\boldsymbol{t}\times\boldsymbol{b}')](\boldsymbol{t}'\cdot\boldsymbol{b})\}(I_{205} - y_3 I_{105}) \\
&-3a^2(\boldsymbol{u}\cdot\boldsymbol{b}')(\boldsymbol{t}\times\boldsymbol{b})(I_{205} - y_3 I_{105}) \\
&-3d^2\{(\boldsymbol{v}'\cdot\boldsymbol{b}')(\boldsymbol{t}\cdot\boldsymbol{b})\boldsymbol{v} - (\boldsymbol{u}\cdot\boldsymbol{b}')(\boldsymbol{u}\cdot\boldsymbol{b})\boldsymbol{v}\}(I_{205} - y_3 I_{105}) \\
&+\frac{3a^2(1-\nu)}{2}\{[\boldsymbol{t}'\cdot(\boldsymbol{b}'\times\boldsymbol{b})]\boldsymbol{u} - [\boldsymbol{t}\times(\boldsymbol{t}'\times\boldsymbol{b}')](\boldsymbol{t}'\cdot\boldsymbol{b})\}(I_{115} - y_3 I_{015}) \\
&-3d^2\{(\boldsymbol{v}'\cdot\boldsymbol{b}')(\boldsymbol{t}'\cdot\boldsymbol{b})\boldsymbol{v} - (\boldsymbol{v}'\cdot\boldsymbol{b}')(\boldsymbol{u}\cdot\boldsymbol{b})\boldsymbol{u}\}(I_{115} - y_3 I_{015}) \\
&+3d(\boldsymbol{u}\cdot\boldsymbol{b}')(\boldsymbol{t}\cdot\boldsymbol{b})\boldsymbol{v}(I_{305} - y_3 I_{205}) + 3d(\boldsymbol{v}'\cdot\boldsymbol{b}')(\boldsymbol{t}'\cdot\boldsymbol{b})\boldsymbol{u}(I_{125} - y_3 I_{025}) \\
&-3d\{(\boldsymbol{u}\cdot\boldsymbol{b}')(\boldsymbol{u}\cdot\boldsymbol{b})\boldsymbol{u} - (\boldsymbol{u}\cdot\boldsymbol{b}')(\boldsymbol{t}'\cdot\boldsymbol{b})\boldsymbol{v} - (\boldsymbol{v}'\cdot\boldsymbol{b}')(\boldsymbol{t}\cdot\boldsymbol{b})\boldsymbol{u}\}(I_{215} - y_3 I_{115}) \\
&-3(\boldsymbol{u}\cdot\boldsymbol{b}')(\boldsymbol{t}\cdot\boldsymbol{b})\boldsymbol{u}(I_{315} - y_3 I_{215}) - 3(\boldsymbol{u}\cdot\boldsymbol{b}')(\boldsymbol{t}'\cdot\boldsymbol{b})\boldsymbol{u}(I_{225} - y_3 I_{125}),
\end{aligned}$$

where

$$\boldsymbol{t}' = \frac{\boldsymbol{x}_2 - \boldsymbol{x}_1}{\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|} \qquad \boldsymbol{u} = \boldsymbol{t}\times\boldsymbol{t}' \qquad \boldsymbol{v} = \boldsymbol{u}\times\boldsymbol{t} \qquad \boldsymbol{v}' = \boldsymbol{t}'\times\boldsymbol{u},$$

$$d = \frac{(\boldsymbol{x}_3 - \boldsymbol{x}_1)\cdot\boldsymbol{u}}{\boldsymbol{u}\cdot\boldsymbol{u}} \qquad y_3 = \frac{(\boldsymbol{x}_3 - \boldsymbol{x}_1)\cdot\boldsymbol{v}'}{\boldsymbol{u}\cdot\boldsymbol{u}},$$

I_{ijk} represents the following class of integral equations:

$$I_{ijk} = \iint \frac{y^i z^j}{(R_a)^k}\,\mathrm{d}y\,\mathrm{d}z \qquad \text{where } y = \frac{\boldsymbol{R}\cdot\boldsymbol{v}'}{\boldsymbol{u}\cdot\boldsymbol{u}} \text{ and } z = \frac{\boldsymbol{R}\cdot\boldsymbol{v}}{\boldsymbol{u}\cdot\boldsymbol{u}}.$$

With this convention

$$I_{001} = yJ^z_{01} + zJ^y_{01} - \left[a^2 + \frac{(\boldsymbol{R}\cdot\boldsymbol{u})^2}{\boldsymbol{u}\cdot\boldsymbol{u}}\right] I_{003},$$

$$I_{003} = -\frac{2}{\sqrt{\boldsymbol{u}\cdot\boldsymbol{u}a^2 + (\boldsymbol{R}\cdot\boldsymbol{u})^2}} \tan^{-1}\left[\frac{(1+\boldsymbol{t}\cdot\boldsymbol{t}')R_a + \boldsymbol{R}\cdot(\boldsymbol{t}+\boldsymbol{t}')}{\sqrt{\boldsymbol{u}\cdot\boldsymbol{u}a^2 + (\boldsymbol{R}\cdot\boldsymbol{u})^2}}\right],$$

$$I_{103} = \frac{1}{\boldsymbol{u}\cdot\boldsymbol{u}}(\boldsymbol{t}\cdot\boldsymbol{t}'J^y_{01} - J^z_{01}),$$

$$I_{013} = \frac{1}{\boldsymbol{u}\cdot\boldsymbol{u}}(\boldsymbol{t}\cdot\boldsymbol{t}'J^z_{01} - J^y_{01}),$$

$$I_{113} = \frac{1}{\boldsymbol{u}\cdot\boldsymbol{u}}[\boldsymbol{t}\cdot\boldsymbol{t}'(yJ^z_{01} - I_{001}) - J^y_{11}],$$

$$I_{203} = \frac{1}{\boldsymbol{u}\cdot\boldsymbol{u}}[(I_{001} + \boldsymbol{t}\cdot\boldsymbol{t}'J^y_{11}) - yJ^z_{01}],$$

$$I_{005} = \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{3[\boldsymbol{u}\cdot\boldsymbol{u}a^2 + (\boldsymbol{R}\cdot\boldsymbol{u})^2]}(I_{003} + yJ^z_{03} + zJ^y_{03}),$$

$$I_{105} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}(\boldsymbol{t}\cdot\boldsymbol{t}'J^y_{03} - J^z_{03}),$$

$$I_{015} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}(\boldsymbol{t}\cdot\boldsymbol{t}'J^z_{03} - J^y_{03}),$$

$$I_{115} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}[\boldsymbol{t}\cdot\boldsymbol{t}'(yJ^z_{03} - I_{003}) - J^y_{13}],$$

$$I_{205} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}(I_{003} + \boldsymbol{t}\cdot\boldsymbol{t}'J^y_{13} - yJ^z_{03}),$$

$$I_{025} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}(I_{003} + \boldsymbol{t}\cdot\boldsymbol{t}'J^z_{13} - zJ^y_{03}),$$

$$I_{215} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}[I_{013} - yJ^z_{13} + \boldsymbol{t}\cdot\boldsymbol{t}'(zJ^y_{13} - I_{103})],$$

$$I_{125} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}[I_{103} - zJ^y_{13} + \boldsymbol{t}\cdot\boldsymbol{t}'(yJ^z_{13} - I_{013})],$$

$$I_{225} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}[(I_{203} - 2\boldsymbol{t}\cdot\boldsymbol{t}'I_{113}) - zJ^y_{23} + (\boldsymbol{t}\cdot\boldsymbol{t}')y^2J^z_{13}],$$

$$I_{305} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}(2I_{103} + \boldsymbol{t}\cdot\boldsymbol{t}'J^y_{23} - y^2J^z_{03}),$$

$$I_{315} = \frac{1}{3\boldsymbol{u}\cdot\boldsymbol{u}}[2I_{113} - I_{203} + (\boldsymbol{t}\cdot\boldsymbol{t}')zJ^y_{23} - y^2J^z_{13}],$$

where J^y_{ij} and J^z_{ij} represent the following integrals:

$$J^y_{ij} = \int \frac{y^i}{(R_a)^j}\,\mathrm{d}y \qquad J^z_{ij} = \int \frac{z^i}{(R_a)^j}\,\mathrm{d}z,$$

$$J^y_{01} = \ln[R_a + \boldsymbol{R}\cdot\boldsymbol{t}] \qquad J^z_{01} = \ln[R_a + \boldsymbol{R}\cdot\boldsymbol{t}'],$$

$$J^y_{11} = R_a \qquad J^y_{13} = J^z_{13} = -\frac{1}{R_a},$$

$$J^y_{03} = -\frac{1}{(R_a + \boldsymbol{R}\cdot\boldsymbol{t})R_a} \qquad J^z_{03} = -\frac{1}{(R_a + \boldsymbol{R}\cdot\boldsymbol{t}')R_a},$$
$$J^y_{23} = 2J^y_{01} - (R_a^2 - y^2)J^y_{03}.$$

Although the expression for the force $\boldsymbol{f}^{12}_{43}$ is expensive to compute, all the integrals and vectors that make up the solution can be reused to calculate $\boldsymbol{f}^{12}_{34}$. Furthermore, with the calculation of four more integrals and a few vectors $\boldsymbol{f}^{34}_{12}$ and $\boldsymbol{f}^{34}_{21}$ can also be obtained.

Appendix A.2. Force on the end node of a segment due to another parallel segment

The above solution breaks down however if the two lines are parallel $\boldsymbol{t} = \boldsymbol{t}'$ or $\boldsymbol{u} = \boldsymbol{0}$. A special solution $\boldsymbol{f}'^{12}_{43}$ must be obtained for the case of parallel lines:

$$\boldsymbol{f}'^{12}_{43} = \frac{\mu}{8\pi(1-\nu)L}[\tilde{\boldsymbol{f}}'(\boldsymbol{x}_4 - \boldsymbol{x}_2, \boldsymbol{x}_4 - \boldsymbol{x}_3) - \tilde{\boldsymbol{f}}'(\boldsymbol{x}_3 - \boldsymbol{x}_2, 0)$$
$$-\tilde{\boldsymbol{f}}'(\boldsymbol{x}_4 - \boldsymbol{x}_1, \boldsymbol{x}_4 - \boldsymbol{x}_3) + \tilde{\boldsymbol{f}}'(\boldsymbol{x}_3 - \boldsymbol{x}_1, 0)],$$

$$\tilde{\boldsymbol{f}}'(\boldsymbol{R}, \boldsymbol{L}) = \{[(\boldsymbol{t}\times\boldsymbol{b}')\times\boldsymbol{t}](\boldsymbol{R}\cdot\boldsymbol{b}) + (\boldsymbol{R}\times\boldsymbol{t})[(\boldsymbol{t}\times\boldsymbol{b}')\cdot\boldsymbol{b}] + [(\boldsymbol{R}\times\boldsymbol{b}')\cdot\boldsymbol{t}](\boldsymbol{b}\times\boldsymbol{t})$$
$$-(1-\nu)[(\boldsymbol{R}\times\boldsymbol{b}')\times\boldsymbol{t}](\boldsymbol{t}\cdot\boldsymbol{b})\}\left(\ln[R_a + \boldsymbol{R}\cdot\boldsymbol{t}] + \frac{(\boldsymbol{R}\cdot\boldsymbol{t} - 2\boldsymbol{L}\cdot\boldsymbol{t})R_a}{R_a^2 - (\boldsymbol{R}\cdot\boldsymbol{t})^2}\right)$$
$$-\nu\{[(\boldsymbol{t}\times\boldsymbol{b}')\times\boldsymbol{t}](\boldsymbol{t}\cdot\boldsymbol{b})\}\left[(3\boldsymbol{R}\cdot\boldsymbol{t} - 2\boldsymbol{L}\cdot\boldsymbol{t})\ln[R_a + \boldsymbol{R}\cdot\boldsymbol{t}]\right.$$
$$\left. + \frac{(\boldsymbol{R}\cdot\boldsymbol{t})(\boldsymbol{R}\cdot\boldsymbol{t} - 2\boldsymbol{L}\cdot\boldsymbol{t})R_a}{R_a^2 - (\boldsymbol{R}\cdot\boldsymbol{t})^2} - 2R_a\right]$$
$$+\{2[(\boldsymbol{R}\times\boldsymbol{b}')\cdot\boldsymbol{t}](\boldsymbol{R}\times\boldsymbol{t})(\boldsymbol{R}\cdot\boldsymbol{b}) - a^2(1-\nu)[(\boldsymbol{R}\times\boldsymbol{b}')\times\boldsymbol{t}](\boldsymbol{t}\cdot\boldsymbol{b})$$
$$+2a^2[(\boldsymbol{R}\times\boldsymbol{b}')\cdot\boldsymbol{t}](\boldsymbol{b}\times\boldsymbol{t})\}\left(\frac{\boldsymbol{L}\cdot\boldsymbol{t}}{(R_a^2 - (\boldsymbol{R}\cdot\boldsymbol{t})^2)R_a} + \frac{(\boldsymbol{R}\cdot\boldsymbol{t} - 2\boldsymbol{L}\cdot\boldsymbol{t})R_a}{(R_a^2 - (\boldsymbol{R}\cdot\boldsymbol{t})^2)^2}\right)$$
$$-\{2[(\boldsymbol{R}\times\boldsymbol{b}')\cdot\boldsymbol{t}](\boldsymbol{R}\times\boldsymbol{t})(\boldsymbol{t}\cdot\boldsymbol{b}) - a^2(1-\nu)[(\boldsymbol{t}\times\boldsymbol{b}')\times\boldsymbol{t}](\boldsymbol{t}\cdot\boldsymbol{b})\},$$
$$\times\left[\frac{1}{R_a} + \frac{(\boldsymbol{R}\cdot\boldsymbol{t})}{(R_a^2 - (\boldsymbol{R}\cdot\boldsymbol{t})^2)}\left(\frac{(\boldsymbol{R}\cdot\boldsymbol{t})}{R_a} + \frac{(\boldsymbol{R}\cdot\boldsymbol{t} - 2\boldsymbol{L}\cdot\boldsymbol{t})R_a}{R_a^2 - (\boldsymbol{R}\cdot\boldsymbol{t})^2}\right)\right].$$

Just as the integrals and vectors that composed the solution of $\boldsymbol{f}^{12}_{43}$ could be reused to compute the forces on the other end points of the two segments, likewise the vectors and integrals in this special case can be reused to calculate forces at the other end points of the two segments for this special case as well. Care must be taken to ensure a smooth transition between non-parallel and parallel expressions for the interaction force between segments as they come close to parallel so as not to introduce a discontinuity in the force field.

Appendix A.3. Force on the end node of a segment due to itself

The self-force $\boldsymbol{f}^{s}_{43}$ at the end node positioned at $\boldsymbol{x}_4$ is determined by using the expression for the force on the end node of a segment due to a parallel segment. However, since in this case $\boldsymbol{x}_3 = \boldsymbol{x}_1$, $\boldsymbol{x}_4 = \boldsymbol{x}_2$ and $\boldsymbol{b} = \boldsymbol{b}'$, the expression can be significantly simplified. The result of all

of the simplification is given in a relatively compact form below:

$$f_{43}^{s} = \frac{\mu}{16\pi(1-\nu)L}[\tilde{f}'(\mathbf{0}, \boldsymbol{x}_4 - \boldsymbol{x}_3) - \tilde{f}'(\boldsymbol{x}_3 - \boldsymbol{x}_4, \mathbf{0}) - \tilde{f}'(\boldsymbol{x}_4 - \boldsymbol{x}_3, \boldsymbol{x}_4 - \boldsymbol{x}_3) + \tilde{f}'(\mathbf{0}, \mathbf{0})],$$

$$f_{43}^{s} = -\frac{\mu}{4\pi}[\boldsymbol{t} \times (\boldsymbol{t} \times \boldsymbol{b})](\boldsymbol{t} \cdot \boldsymbol{b}) \left\{ \frac{\nu}{1-\nu} \left(\ln \left[\frac{L_a + L}{a} \right] - 2\frac{L_a - a}{L} \right) - \frac{(L_a - a)^2}{2L_a L} \right\},$$

$$L_a = \sqrt{L^2 + a^2}.$$

From the expression, it is clear that the self-force $\boldsymbol{f}_{34}^{s}$ at the end node positioned at $\boldsymbol{x}_3$ is related to $\boldsymbol{f}_{43}^{s}$ through the equality $\boldsymbol{f}_{34}^{s} = -\boldsymbol{f}_{43}^{s}$.

Appendix B. Fast multipole algorithms for far field interactions

The mathematical framework of the fast multipole method is based on the formulae for generating and evaluating multipole expansions and a few translation theorems. The formulae for multipole operations have appeared before in the literature (LeSar and Rickman 2002, Wang *et al* 2004). The following sections contain a summary of how the FMM formulae are implemented in the ParaDiS code to enhance their computability. In general, the Einstein summation convention will be used for lower indices. Vectors and tensors will be denoted by bold face, e.g. the stress tensor $\boldsymbol{\sigma}$, whereas scalars and elements in vectors and tensors will appear in regular face, as in σ_{ij}.

The multipole expansion is based on the derivatives of the stress field, and a central component of its implementation is a good representation of the nth derivative of the distance function $R(\boldsymbol{x}) = \|\boldsymbol{x}\|$. The elements of the first derivative of $R(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ may be written in the following equivalent form:

$$R_i = \partial_{x_i} R = \frac{x_i}{R}. \tag{36}$$

We use the notation $\partial_{x_i}^{n} = (\partial^n / \partial x_i^n)$ to denote the elements of repeated differentiation of a function. With this notation, an element of the nth derivative of $R(\boldsymbol{x})$ is represented by the left-hand side of the following expression and calculated using the sum on the right-hand side,

$$\partial_{x_1}^{n_1} \partial_{x_2}^{n_2} \partial_{x_3}^{n_3} R(\boldsymbol{x}) = \sum_{\alpha=0}^{\lfloor n_1/2 \rfloor} \sum_{\beta=0}^{\lfloor n_2/2 \rfloor} \sum_{\gamma=0}^{\lfloor n_3/2 \rfloor} (-1)^{\zeta} (2n - 3 - 2\zeta)!! C_1^{\alpha} C_2^{\beta} C_3^{\gamma}, \tag{37}$$

where $n = n_1 + n_2 + n_3$, $\zeta = \alpha + \beta + \gamma$ and

$$C_j^{\iota} = \left(\frac{x_j}{R} \right)^{n_j - 2\iota} \binom{n_j}{2\iota} \frac{(2\iota)!}{\iota!\, 2^{\iota}}. \tag{38}$$

$\lfloor x \rfloor$ is the floor operator which rounds down to the nearest integer $\leqslant x$. $n!!$ is the double factorial:

$$n!! = \begin{cases} 1 \cdot 3 \cdot 5 \cdots (n-2) \cdot n, & n \text{ odd}, \\ 2 \cdot 4 \cdot 6 \cdots (n-2) \cdot n, & n \text{ even}, \end{cases}$$

and we define $(-1)!! = 1$.

Equation (37) is derived from equation (A4) in LeSar and Rickman (2002) by collecting similar terms. The full nth derivative of R is a symmetric n-dimensional tensor where the order

in which the derivatives are taken does not matter (e.g. $(\partial^2 R/\partial x_1 \partial x_2) = (\partial^2 R/\partial x_2 \partial x_1)$). An element in the nth derivative of R written as $\partial_{x_1}^{n_1}\partial_{x_2}^{n_2}\partial_{x_3}^{n_3} R$ occurs

$$M^{n_1 n_2 n_3} = \frac{(n_1 + n_2 + n_3)!}{n_1! n_2! n_3!}$$

times in the n-dimensional tensor but needs to be computed only once.

The collection of similar terms and the computation of each unique element in the nth derivative of R only once reduce the number of terms that need to be computed and stored in the multipole expansion of the dislocation density. The computational cost of creating and evaluating multipole and Taylor series expansions is reduced by a factor of ≈6, and the computational cost of translating the centres of multipole and Taylor series expansions as well as the cost of converting multipole expansions to Taylor series expansions is reduced by a factor of ≈36.

Appendix B.1. The multipole expansion

The stress $\boldsymbol{\sigma}$ at a point $\boldsymbol{x}'$ induced by a dislocation segment is given by the line integral

$$\sigma_{ij}(\boldsymbol{x}') = \frac{\mu b_n}{8\pi} \int R_{mpp}(\boldsymbol{x}' - \boldsymbol{x})[\epsilon_{jmn}\mathrm{d}x_i + \epsilon_{imn}\,\mathrm{d}x_j] + \frac{\mu b_n}{4(1-\nu)\pi} \int \epsilon_{kmn}[R_{ijm}(\boldsymbol{x}' - \boldsymbol{x}) - \delta_{ij} R_{ppm}(\boldsymbol{x}' - \boldsymbol{x})]\,\mathrm{d}x_k, \tag{39}$$

where μ is the bulk modulus, ν is the Poisson ratio, $\boldsymbol{b}$ is the Burgers vector, ϵ is the permutation tensor and $\boldsymbol{x}$ is a position of a point on the dislocation segment. The integral is taken along the dislocation segment. The position of a point $\boldsymbol{x}$ on a dislocation line segment starting at $\boldsymbol{x}^0$ and terminating at $\boldsymbol{x}^1$ may be described by the expression $x_i(s) = x_i^0 + s\xi_i$, with $0 \leqslant s \leqslant 1$ and $\boldsymbol{\xi} = \boldsymbol{x}^1 - \boldsymbol{x}^0$.

It may be more efficient to approximate the integral by truncating an infinite series solution to the integral that converges quickly if the length of the segment is much smaller than the distance between the segment and the field point at which the stress is evaluated. The infinite series solution requires calculating the multipole moment expansion $\eta_{ij}^{\alpha\beta\gamma}$ of the dislocation segment about a point $\boldsymbol{x}''$ that is much closer to the dislocation segment than the point $\boldsymbol{x}'$ where the stress is evaluated. The multipole moment expansion of the segment is defined by

$$\eta_{ij}^{\alpha\beta\gamma}(\boldsymbol{x}'') \equiv b_i \xi_j \int_0^1 (x_1 - x_1'')^\alpha (x_2 - x_2'')^\beta (x_3 - x_3'')^\gamma \,\mathrm{d}s. \tag{40}$$

The integrand is a polynomial of order $\zeta = \alpha + \beta + \gamma$ that can be efficiently integrated with the Gaussian quadrature using $\lfloor(\zeta + 3)/2\rfloor$ quadrature points. Equation (40) is used to build the multipole moment expansion in an FMM cell at the lowest level of the hierarchy by summing up the contributions of the dislocation segments contained within that cell and using the multipole moment expansion of the sum to compute the stress in non-neighbouring FMM cells.

The integral expression of the stress may now be rewritten as an infinite sum of the multipole moments of the dislocation segment. These multipole moments form the coefficients in a Taylor expansion of the stress which converges rapidly far away from the dislocation line segment. The infinite series of the stress takes the form

$$\sigma_{ij}(\boldsymbol{x}') = \frac{\mu}{8\pi}\left[\epsilon_{jmn}\mathrm{G}_{nimpp} + \epsilon_{imn}\mathrm{G}_{njmpp} + \frac{2\epsilon_{kmn}}{1-\nu}(\mathrm{G}_{nkijm} - \delta_{ij}\mathrm{G}_{nkmpp})\right], \tag{41}$$

where

$$\mathrm{G}_{ijklm} = \sum_{\zeta=0}^{\infty} \sum_{\alpha=0}^{\zeta} \sum_{\beta=0}^{\zeta-\alpha} M^{\alpha\beta\gamma} \eta_{ij}^{\alpha\beta\gamma}(\boldsymbol{x}'') \partial_{x_1}^{\alpha} \partial_{x_2}^{\beta} \partial_{x_3}^{\gamma} R_{klm}(\boldsymbol{x}' - \boldsymbol{x}''), \tag{42}$$

with $\gamma = \zeta - \alpha - \beta$.

In implementation, one can use the invariance of G_{ijklm} w.r.t. permutations of k, l, m, so that $\mathrm{G}_{ijklm} = \mathrm{G}_{ijkml} = \mathrm{G}_{ijlkm} = \mathrm{G}_{ijlmk} = \mathrm{G}_{ijmkl} = \mathrm{G}_{ijmlk}$. Also, when evaluating a multipole expansion, only certain elements of G_{ijklm} and G_{ijkll} are needed. None of the elements of G_{iiklm}, G_{ijkli}, G_{ijklj} and G_{ijill} are ever used in the stress expression. The above symmetries and unused elements save a factor of 6 in the number of elements of $\boldsymbol{G}$ that need to be computed.

Appendix B.2. Translation operators

The upward pass in the FMM algorithm requires shifting the point $\boldsymbol{x}''$ about which a multipole moment expansion is defined. We have a set of multipole coefficients $\boldsymbol{\eta}$ calculated for the expansion about a point $\boldsymbol{x}''$ and wish to obtain the coefficients for the expansion about different point $\boldsymbol{y}''$. The expansion $\boldsymbol{\eta}(\boldsymbol{y}'')$ may be written as a function of the expansion $\boldsymbol{\eta}(\boldsymbol{x}'')$ through the expression

$$\eta_{ij}^{\alpha\beta\gamma}(\boldsymbol{y}'') = \sum_{a=0}^{\alpha} \sum_{b=0}^{\beta} \sum_{c=0}^{\gamma} \binom{\alpha}{a} \binom{\beta}{b} \binom{\gamma}{c} (x_1'' - \mathrm{y}_1'')^a (x_2'' - \mathrm{y}_2'')^b (x_3'' - y_3'')^c \eta_{ij}^{abc}(\boldsymbol{x}''). \tag{43}$$

The $\boldsymbol{\eta}$ of a parent cell in the FMM hierarchy is formed by combining the translation of the multipole expansions of the 8 daughter cells from the centres of their cells to the centre of their parent's cell. The process is performed layer by layer, from the next finest grid to the top level, so that multipole expansions of all cells in the FMM hierarchy are formed.

A multipole expansion constructed from a set of segments accurately represents the stress due to those segments at points far away from the segments used in its construction. Rather than using equation (41) to compute the stress at points within a remote cell, a local Taylor series expansion of the stress field is constructed about the centre $\boldsymbol{x}'''$ of the target cell. The local Taylor series expansion is then used to calculate the stress at points $\boldsymbol{x}'$ on dislocation segments owned by the cell. The Taylor series expansion of the stress takes the form

$$\sigma_{ij}(\boldsymbol{x}') = \sum_{\zeta=0}^{\infty} \frac{(-1)^{\zeta}}{\zeta!} \sum_{\alpha=0}^{\zeta} \sum_{\beta=0}^{\zeta-\alpha} (x_1' - \mathrm{x}_1''')^{\alpha} (x_1' - x_1''')^{\beta} (x_1' - x_1''')^{\gamma} \mathrm{T}_{ij}^{\alpha\beta\gamma}(\boldsymbol{x}'''), \tag{44}$$

where

$$\mathrm{T}_{ij}^{\alpha\beta\gamma}(\boldsymbol{x}''') = \frac{\mu}{8\pi} \left[\epsilon_{jmn} \mathrm{G}_{nimpp}^{\alpha\beta\gamma} + \epsilon_{imn} \mathrm{G}_{njmpp}^{\alpha\beta\gamma} + \frac{2\epsilon_{kmn}}{1-\nu} \left(\mathrm{G}_{nkijm}^{\alpha\beta\gamma} - \delta_{ij} \mathrm{G}_{nkmpp}^{\alpha\beta\gamma} \right) \right], \tag{45}$$

$$\mathrm{G}_{ijklm}^{\alpha\beta\gamma} = \sum_{t=0}^{\infty} \sum_{a=0}^{t} \sum_{b=0}^{t-a} M^{abc} \eta_{ij}^{abc}(\boldsymbol{x}'') \partial_{x_1}^{\alpha+a} \partial_{x_2}^{\beta+b} \partial_{x_3}^{\gamma+c} R_{klm}(\boldsymbol{x}''' - \boldsymbol{x}''), \tag{46}$$

with $\gamma = \zeta - \alpha - \beta$ and $c = t - a - b$.

In our FMM scheme, there are 189 sets of multipole expansions that are combined to form the Taylor series expansion of the stress in another cell. The calculation is conducted for every cell in the FMM hierarchy and is the most computationally intensive element of the FMM implementation. The translation of $\eta_{ij}^{abc}(\boldsymbol{x}'')$ to $\mathrm{T}_{ij}^{abc}(\boldsymbol{x}''')$ is a linear operation on the elements of $\eta_{ij}^{abc}(\boldsymbol{x}'')$, and the linear operator is a function of a set of quantized vectors $\boldsymbol{y} = \boldsymbol{x}''' - \boldsymbol{x}''$

between FMM cell centres. By scaling, and using rotation and reflection symmetries of the linear operator on $\boldsymbol{y}$, the number of unique linear operators can be reduced from 189 to 13. The 13 unique linear operators at each level of the hierarchy can be computed once in the beginning of the simulation and stored in the memory (if available).

Lastly, the downward pass algorithm in the FMM implementation requires shifting the point about which a Taylor series expansion of the stress is defined from $\boldsymbol{x}'''$ to $\boldsymbol{y}'''$. This shift can be performed in a manner similar to the shift for multipole moments in equation (43). Assume we have a Taylor expansion $\boldsymbol{T}(\boldsymbol{x}''')$, transforming that into an expansion $\boldsymbol{T}(\boldsymbol{y}''')$ is done by

$$\mathrm{T}_{ij}^{\alpha\beta\gamma} = \sum_{a=\alpha}^{\zeta} \sum_{b=\beta}^{\zeta-a} \sum_{c=\gamma}^{\zeta-a-b} \binom{a}{\alpha}\binom{b}{\beta}\binom{c}{\gamma} (y_1''' - x_1''')^{a-\alpha} (y_2''' - x_2''')^{b-\beta} (y_3''' - x_3''')^{c-\gamma} \mathrm{T}_{ij}^{abc}, \tag{47}$$

where $\zeta = \alpha + \beta + \gamma$.

The translation in equation (47) is used to shift the complete Taylor series expansion of the stress field in a parent FMM cell to its eight daughter cells. The daughter cells are required to add this contribution from the parent to the contribution from the cells on its level of the hierarchy that are outside its nearest neighbour distance but inside the nearest neighbour distance of its parent cell using equation (44). The process is recursively repeated until the lowest level in the FMM hierarchy is reached.

Appendix C. Description of a linear BCC mobility law

In BCC metals, screw dislocations do not dissociate into partial dislocations the same way as they do in FCC metals. Therefore, for simulations of BCC crystals, it may be better not to assign glide plane normals $\boldsymbol{n}$ to screw segments. Instead, screw dislocations should be given the same mobility in all directions perpendicular to the line, i.e.

$$\mathcal{B}(\boldsymbol{t}) = B_{\mathrm{s}}(\boldsymbol{I}_2 - \boldsymbol{t} \otimes \boldsymbol{t}), \qquad \text{when } \boldsymbol{t} \parallel \boldsymbol{b}, \tag{110}$$

where B_{s} is a drag coefficient for the motion of screw dislocations. This isotropic mobility for screws mimics the so-called 'pencil-glide' behaviour of dislocations observed in BCC metals at elevated temperatures.

At the same time, the drag coefficient tensor for the non-screw segments should remain anisotropic with respect to glide and climb. Let us define the drag coefficient tensor for a pure edge dislocation as

$$\mathcal{B}(\boldsymbol{t}) = B_{\mathrm{eg}}(\boldsymbol{m} \otimes \boldsymbol{m}) + B_{\mathrm{ec}}(\boldsymbol{n} \otimes \boldsymbol{n}), \qquad \text{when } \boldsymbol{t} \perp \boldsymbol{b}, \tag{111}$$

where B_{eg} and B_{ec} are the drag coefficients for motion in glide and climb directions, respectively, and the unit vectors are defined as $\boldsymbol{n} \equiv \boldsymbol{b} \times \boldsymbol{t}/\|\boldsymbol{b} \times \boldsymbol{t}\|$ and $\boldsymbol{m} \equiv \boldsymbol{n} \times \boldsymbol{t}$. To completely specify the drag coefficient tensor for all segment orientations, we need a function that smoothly interpolates between the two limits: pure screw and pure edge orientations. A possible form for such an interpolation function is

$$\mathcal{B}(\boldsymbol{t}) = B_{\mathrm{g}}(\boldsymbol{m} \otimes \boldsymbol{m}) + B_{\mathrm{c}}(\boldsymbol{n} \otimes \boldsymbol{n}), \qquad \text{when } \boldsymbol{t} \times \boldsymbol{b} \neq 0, \tag{112}$$

$$B_{\mathrm{g}} = \frac{1}{\|\boldsymbol{b}\|}[B_{\mathrm{eg}}^{-2}\|\boldsymbol{b} \times \boldsymbol{t}\|^2 + B_{\mathrm{s}}^{-2}(\boldsymbol{b} \cdot \boldsymbol{t})^2]^{-1/2},$$

$$B_{\mathrm{c}} = \frac{1}{\|\boldsymbol{b}\|}[B_{\mathrm{ec}}^{2}\|\boldsymbol{b} \times \boldsymbol{t}\|^2 + B_{s}^{2}(\boldsymbol{b} \cdot \boldsymbol{t})^2]^{1/2}.$$

In the limit $\boldsymbol{t} \times \boldsymbol{b} \to 0$, the above function becomes the same as equation (110).

As currently defined, $\mathcal{B}$ is singular and cannot be inverted. In most cases, this does not cause problems because it is only part of the nodal drag equation, and when combined with the contributions of other segments to the total nodal drag, the total drag on a node becomes non-singular. This is always the case for physical nodes in the system, because no three segments connected to the same node can all be parallel. However, as the two line segments connected to a discretization node become co-linear, the nodal drag tensor becomes increasingly less well conditioned. The problem lies in the fact that there is no drag explicitly specified for forces along the tangent line direction of a segment. This can be remedied by adding a contribution to $\mathcal{B}$ which will set the mobility of a node in the line tangent direction in the form

$$\mathcal{B}(\boldsymbol{t}) = B_{\mathrm{g}}(\boldsymbol{m} \otimes \boldsymbol{m}) + B_{\mathrm{c}}(\boldsymbol{n} \otimes \boldsymbol{n}) + B_{\mathrm{l}}(\boldsymbol{t} \otimes \boldsymbol{t}), \tag{113}$$

where B_{l} is the drag associated with moving a node along its line direction. In implementation, B_{l} should be much less than B_{eg}, B_{ec} and B_{s} so that it does not adversely affect the mechanical behaviour of the system. This value of the drag most directly affects the rate of redistribution discretization node towards an optimal configuration.

References

Allen M and Tildesley D 1989 *Computer Simulation of Liquids* (Oxford: Oxford University Press)

Bacon D, Barnett D and Scattergood R 1979 Anisotropic elastic field of a dislocation segment in 3 dimensions *Phil. Mag.* A **39** 231–5

Bulatov V V *et al* 2006 Dislocation multi-junctions and strain hardening *Nature* **54** 561–87

Bulatov V and Cai W 2006 *Computer simulations of dislocations* (Oxford: Oxford University Press) chapter 10

Cai W, Arsenlis A, Weinberger C and Bulatov V 2006 A non-singular continuum theory of dislocations *J. Mech. Phys. Solids* **54** 561–87

Cai W, Bulatov V, Chang J, Li J and Yip S 2003 Periodic image effects in dislocation modeling *Phil. Mag.* **83** 539–67

Challacombe M, White C and Head-Gordon M 1997 Periodic boundary conditions and the fast multipole method *J. Chem. Phys.* **107** 10131–40

Devincre B 1996 Meso-scale simulation of the dislocation dynamics *Computer Simulation in Materials Science* (Dordrecht: Kluwer) pp 309–23

Devincre B and Kubin L 1997 Mesoscopic simulations of dislocations and plasticity *Mater. Sci. Eng.* A **A234–236** 561–87

Devincre B, Kubin L, Lemarchand C and Madec R 2001 Mesoscopic simulations of plastic deformation *Mater. Sci. Eng.* A **A309–310** 211–19

Fivel M, Gosling T and Canova G R 1996 Implementing image stresses in a 3d dislocation simulation *Modelling Simul. Mater. Sci. Eng.* **4** 581–96

Ghoniem N and Sun L 1999 Fast-sum method for the elastic field of three-dimensional ensembles *Phys. Rev.* B **60** 128–40

Ghoniem N, Tong S and Sun L 2000 Parametric dislocation dynamics: a thermodynamics-based approach to investigation of mesoscopic plastic deformation *Phys. Rev.* B **61** 913–27

Greengard L and Rokhlin V 1997 A new version of the fast multipole method for the Laplace equation in three dimensions *Acta Numer.* **6** 229–69

Han L, Ghoniem N and Wang Z 2003 Parametric dislocation dynamics of anisotropic crystals *Phil. Mag.* **83** 3705–21

Henager C and Hoagland R 2005 Dislocation and stacking fault core fields in fcc metals *Phil. Mag.* **85** 4477–508

Kelley C T, Kevrekidis I G and Qiao L 2004 Newton-Krylov solvers for time-steppers http://arxiv.org/PS_cache/math/pdf/0404/0404374.pdf

Khraishi T and Zbib H 2002 Free-surface effects in 3d dislocation dynamics: formulation and modeling *J. Eng. Mater. Technol.* **124** 342–51

Kubin L, Canova G, Condat M, Devincre B, Pontikis V and Bréchet Y 1992 Dislocation microstructures and plastic flow: a 3d simulation *Solid State Phenomena* **23–24** 455–72

Kukta R V 1998 Observations of the kinetics of relaxation in epitaxial films grown on conventional and compliant substrates: a continuum simlation of dislocation glide near an interface *PhD Thesis* Brown University

LeSar R and Rickman J K 2002 Multipole expansion of dislocation interactions: application to discrete systems *Phys. Rev.* B **65** 144110

Liu X and Schwarz K 2005 Modelling of dislocations intersecting a free surface *Modelling Simul. Mater. Sci. Eng.* **13** 1233–47

Madec R, Devincre B and Kubin L 2002 From dislocation junctions to forest hardening *Phys. Rev. Lett.* **89** 255508

Rhee M, Stolken J, Bulatov V, de la Rubia T, Zbib H and Hirth J 2001 Dislocation stress fields for dynamic codes using anisotropic elasticity: methodology and analysis *Mater. Sci. Eng.* A **309** 288–93

Rhee M, Zbib H, Hirth J, Huang H and de la Rubia T 1998 Models for long-/short-range interactions and cross slip in 3d dislocation simulation of bcc single crystals *Modelling Simul. Mater. Sci. Eng.* **6** 467–92

Schwarz K 1999 Simulaiton of dislocations on the mesoscopic scale: I. Methods and examples *J. Appl. Phys.* **85** 108–19

Schwarz K 2003 Local rules for approximating strong dislocation interactions in discrete dislocation dynamics *Modelling Simul. Mater. Sci. Eng.* **11** 609–25

Shenoy V B, Kukta R V and Phillips R 2000 Mesoscopic analysis of structure and strength of dislocation junctions in fcc metals *Phys. Rev. Lett.* **84** 1491–4

Tang M, Cai W, Xu G and Bulatov V V 2006 A hybrid method for computing forces on curved dislocations intersecting free surfaces in three dimensional dislocation dynamics *Modelling Simul. Mater. Sci. Eng.* **14** 1139–51

van der Giessen E and Needleman A 1995 Discrete dislocation plasticity: a simple planar model *Modelling Simul. Mater. Sci. Eng.* **3** 689–735

Wang Z, Ghoniem N and LeSar R 2004 Multipole representation of the elastic field of dislocation ensembles *Phys. Rev.* B **69** 174102

Weygand D, Friedman L, Van der Giessen E and Needleman A 2002 Aspects of boundary value problem solutions with three-dimensional dislocation dynamics *Modelling Simul. Mater. Sci. Eng.* **10** 437–68

Zbib H, Diaz de la Rubia T, Rhee M and Hirth J 2000 3d dislocation dynamics; stress–strain behavior and hardening mechanisms in fcc and bcc metals *J. Nucl. Mater.* **276** 154–65

Zbib H, Rhee M and Hirth J 1998 On plastic deformation and the dynamics of 3d dislocations *Int. J. Mech. Sci.* **40** 113–27