Efficient time integration in dislocation dynamics

2014 Modelling Simul. Mater. Sci. Eng. 22 025003

(http://iopscience.iop.org/0965-0393/22/2/025003)

IOP Publishing Modelling and Simulation in Materials Science and Engineering

Modelling Simul. Mater. Sci. Eng. **22** (2014) 025003 (26pp) doi:10.1088/0965-0393/22/2/025003

Efficient time integration in dislocation dynamics

Ryan B Sills[1,2] **and Wei Cai**[1]

[1] Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA
[2] Sandia National Laboratories, Livermore, CA 94551, USA

E-mail: rbsills@stanford.edu

Received 14 June 2013, revised 4 November 2013
Accepted for publication 4 December 2013
Published 10 January 2014

Abstract

The efficiencies of one implicit and three explicit time integrators have been compared in line dislocation dynamics simulations using two test cases: a collapsing loop and a Frank–Read (FR) source with a jog. The time-step size and computational efficiency of the explicit integrators is shown to become severely limited due to the presence of so-called stiff modes, which include the oscillatory zig-zag motion of discretization nodes and orientation fluctuations of the jog. In the stability-limited regime dictated by these stiff modes, the implicit integrator shows superior efficiency when using a Jacobian that only accounts for short-range interactions due to elasticity and line tension. However, when a stable dislocation dipole forms during a jogged FR source simulation, even the implicit integrator suffers a substantial drop in the time-step size. To restore computational efficiency, a time-step subcycling algorithm is tested, in which the nodes involved in the dipole are integrated over multiple smaller, local time steps, while the remaining nodes take a single larger, global time step. The time-step subcycling method leads to substantial efficiency gain when combined with either an implicit or an explicit integrator.

Keywords: dislocation dynamics, time integrator, implicit method, subcycling

(Some figures may appear in colour only in the online journal)

1. Introduction

Modeling crystal plasticity by dislocation dynamics (DD) has been a computationally challenging endeavor for the last 25 years [1–9]. In DD, individual dislocations are discretized and modeled in a material simulation cell. By accounting for the various driving forces on each

segment arising from elastic interactions between segments, applied loads, and other features such as free surfaces and obstacles, and then applying suitable mobility laws, the plastic deformation of the material arises naturally from motion of the dislocations. Such models are computationally challenging for two main reasons: (1) force calculations are expensive due to the long-range stress fields of dislocations and (2) efficient time integration of dislocation motion is difficult. While much research has focused on dealing with the former issue [9–11], the latter issue has gone largely uninvestigated.

Most DD models assume dislocation motion is over-damped, where drag forces intrinsic to the crystal lattice are dominant so that inertia can be neglected. This leads to a system of first-order ordinary differential equations (ODEs) of motion of the form [12]

$$\boldsymbol{v} \equiv \frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{g}(\{\boldsymbol{r}\}), \tag{1}$$

where $\boldsymbol{r}$ is the $3N$-dimensional vector of nodal positions of N nodes, and $\boldsymbol{g}(\cdot)$ is a mobility function that subsumes the force calculation [13] and the material-specific mobility law [14]. The problem of DD is thus to numerically solve this system of ODEs efficiently and accurately. Because DD seeks a solution to metal plasticity in the time domain, efficient time integration is necessary, in addition to efficient force calculation, at each time step.

The potential of DD to provide insight into to the physics of work hardening in metals can be fully realized only if DD simulations can accumulate plastic strains comparable to experimental values, usually on the order of 10% or higher. To date, DD simulations of material volumes large enough to represent bulk metal behavior ($\sim$1000 μm^3) are limited to about 3% strain or less [9, 15, 17]. Much of the work to overcome this disparity has focused on developing efficient, parallelized algorithms for force calculations [9–11, 16]. Despite the implementation of a variety of such approaches, simulation strains remain small. On the other hand, a survey of the major DD codes reveals that most use very simple explicit time integrators (e.g. forward Euler [3, 4, 18, 19, 36]). Very rarely are these simple approaches the most efficient. Some researchers have considered more advanced time integration algorithms in DD [20]. This work aims to extend these efforts by evaluating the performance of a variety of different time integrators and the time-step-limiting aspects of DD simulations.

In order to thoroughly investigate time integration in DD, we break up the problem into two separate studies. The first study, which is presented here, will examine the roles of accuracy and stability on the simulation time-step size, but will exclude the effects of topological changes (i.e. dislocation annihilation and junction formation, but remeshing is allowed); these effects will be avoided here for the sake of scope and clarity and will be the subject of an upcoming companion paper. It will be demonstrated that even in the absence of topological changes, achieving efficient time integration in DD is a complex and challenging problem. There exist multiple mechanisms that can dramatically reduce the time-step size of a simple integrator, and an efficient integration scheme needs to address all of them. This paper will present two case studies: a prismatic loop and a Frank–Read (FR) source with a jog. Though these case studies are small, they provide insight into the problematic modes of dislocation motion that can severely hinder the overall efficiency. Specifically, we will show that short dislocation segments and segments in close proximity, both of which are commonly found in large-scale simulations, introduce such modes. The overarching goal of these studies is to determine an optimized solution algorithm for DD simulations of bulk work hardening. The first step in doing so is optimizing the algorithm for smaller problems that isolate problematic modes, and this is the goal of this work.

The remainder of this paper will be organized as follows. Section 2 introduces the explicit and implicit integrators examined in this work. Section 3 compares the performance of these integrators in the two test cases. The analysis will focus on the modes of dislocation motion that

cause significant drops in the time step with certain integrators, and the algorithmic features needed to efficiently increase the time step. Section 4 presents a discussion on how the findings here can be applied to large-scale DD simulations of bulk work hardening, followed by a summary in section 5.

2. Time integrators

2.1. Overview

Time integrators can be broken into two broad categories: explicit and implicit. Explicit integrators provide an estimate of the solution at the next time-step based only on current and past nodal positions and velocities, allowing the solution to be determined explicitly in one calculation. They are relatively simple to implement and a variety of different types exist with various orders of accuracy. In contrast, implicit integrators require knowledge of the nodal velocities at the next time step also. Since these velocities are functions of the nodal positions at the next time step, which are still unknown, implicit integrators yield implicit expressions that must be solved iteratively (if the system is nonlinear), usually by the Newton–Raphson method. Solving the problem in this way is computationally much more demanding, but the added benefit is stability. When the system of ODEs contains phenomena occurring at both very small and very large time scales, it is called *stiff* and an implicit integrator is usually necessary to integrate it efficiently.

Therefore, an important question to ask when selecting a time integrator is whether the problem is stiff. Any system has a number of modes associated with its degrees of freedom and each has a characteristic time constant or frequency. Some modes may be of little practical interest in terms of the overall behavior we are trying to capture, but may introduce tremendous error to the simulation if the time step is not sufficiently small. For instance, the oscillation of a dislocation segment in a stable configuration during a work hardening simulation will bear no weight on the overall behavior. If such modes are present and have much smaller time constants than the main modes of interest (i.e. a FR source bowing out), then the system is said to be stiff. In this case, all explicit integrators will be stability limited and forced to take very small time steps; a large time step can cause dislocation motion to become unstable leading to unphysical multiplication. When this happens, an implicit integrator is likely to provide much higher efficiency [21, 22].

If, on the other hand, the system is not stiff and an explicit integrator can be competitive, there is a broad library of such integrators available for use. Which integrator will be most effective is dictated by the characteristics of the specific problem [25], so the selection process requires trial and error. Performance can vary significantly across different classes of integrators making it worthwhile to consider a broad range. For DD simulations, we find that both explicit and implicit integrators can be competitive, depending on the problem of interest.

2.2. Error control

For all simulations conducted in this work, the time-step size will be selected adaptively based on the error estimate provided by each method and the error tolerance allowed by the user. The error estimate for each integration scheme is presented below or in appendix A. To specify the error tolerance, we employ both relative and absolute error control. To control absolute error at each time step, the positioning error of each node, e_i, must be less than the absolute tolerance, r_{tol}. Relative error is limited by requiring that the ratio of the positioning error to the total distance traveled for each node, $e_i/\|\boldsymbol{r}_i(t+\Delta t) - \boldsymbol{r}_i(t)\|$ where $\boldsymbol{r}_i$ is the

position vector of node i and $\|\cdot\|$ denotes the L_2 norm, must be less than the relative error tolerance, f_{tol}. When dislocation segments are moving very slowly, the relative error criterion can lead to unnecessarily small error requirements. We therefore also enforce an error threshold, r_{th}, such that any node with absolute error below r_{th} is considered to have passed both requirements ($r_{\text{th}} \ll r_{\text{tol}}$). Unless otherwise stated, the tolerance parameters were given the values $r_{\text{tol}}/b = 10$, $f_{\text{tol}} = 0.01$, and $r_{\text{th}}/b = 0.01$, where b is the Burgers vector magnitude. These values were found to provide physically reasonable solutions (e.g. stable with smooth line structures) for all simulations presented here.

2.3. Explicit integrators

Three explicit integrators are evaluated in this paper: the Heun method, the Runge–Kutta–Fehlberg (RKF) method, and a Bulirsch–Stoer (BS) method. They are briefly summarized below. More implementation details are given in appendix A.

Both the Heun and RKF methods belong to the Runge–Kutta family of methods. These are referred to as single-step methods in that they use (mobility) function evaluations only within the range of integration (i.e. between the current and future time steps). The Heun method [23] is also called the Euler-trapezoid method and is used in the ParaDiS program [9]. It uses a forward Euler predictor and a trapezoidal method corrector:

$$\boldsymbol{r}_{k+1}^{0} = \boldsymbol{r}_k + \Delta t\, \boldsymbol{g}(\boldsymbol{r}_k) \tag{2}$$

$$\boldsymbol{r}_{k+1}^{j+1} = \boldsymbol{r}_k + \Delta t\, \frac{\boldsymbol{g}(\boldsymbol{r}_k) + \boldsymbol{g}(\boldsymbol{r}_{k+1}^{j})}{2}, \tag{3}$$

where we have denoted the nodal position vector at time step k as $\boldsymbol{r}_k$, the iterate number as a superscript, and the time-step size as Δt. The predictor and corrector are globally first and second order accurate, respectively. The error for each node i is defined as

$$e_i = \|(\boldsymbol{r}_{k+1}^{j+1})_i - (\boldsymbol{r}_{k+1}^{j})_i\|. \tag{4}$$

If the error is too large after using equations (2) and (3), then a fixed point iteration is applied with equation (3). After a pre-specified maximum number of iterations, if the iteration fails to converge the time step Δt is reduced and the process is attempted again (see appendix A for details).

The RKF method provides a fifth-order accurate solution and an error estimate in only six function evaluations. The RKF algorithm is outlined in appendix A and further information can be found in [24].

BS methods are a family of approaches that apply an explicit method over the same time range with multiple step sizes and then use Richardson extrapolation to produce a higher order estimate of the solution. This overall technique is called Romberg integration, and we utilize an implementation that uses the modified midpoint method [24]. With this method, each iteration increases the order of accuracy by two, so with the iteration constraints imposed here (see appendix A) the order of accuracy varies between 2, 4 and 6.

We end this section with some comments on multi-step methods, which are commonly used in numerical simulations, but are not evaluated in this paper. Multi-step methods use function evaluations from previous time steps to augment the order of accuracy. These methods are referred to as non-self-starting because there are no previous time steps to use at the beginning of the simulation. Thus a different method must be used until a sufficient nodal history is developed. In addition, for these methods, stability requires that the solution be smooth and continuous [25]. These aspects are particularly troublesome for DD simulations because new nodes are added and discontinuous topological changes occur frequently, effectively resetting

the time-step count to zero for these nodes. Furthermore, the fact that we need to maintain nodal history at intervals equal to the current time-step size imposes strict requirements on how the time-step size can be changed. For these reasons, multi-step methods will not be discussed further in this paper.

2.4. Implicit integrators

The most common implicit methods applied to stiff problems are the backward differentiation formulas (BDFs) developed by Gear [21]. These formulas are variable-order, implicit, multi-step relationships for the solution at the next time step. As discussed above, multi-step methods, implicit or explicit, are difficult to use in DD simulations where remeshing and topological changes occur. Therefore, in this work, instead of BDFs, we choose the single-step trapezoidal method:

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k + \Delta t \, \frac{\boldsymbol{g}(\boldsymbol{r}_k) + \boldsymbol{g}(\boldsymbol{r}_{k+1})}{2}. \tag{5}$$

The solution of equation (5) requires a nonlinear equation solver, for which we choose the Newton–Raphson method. First we rewrite equation (5) as the search for the root of function $\mathcal{F}(\cdot)$, where

$$\mathcal{F}(\boldsymbol{r}_{k+1}) = \boldsymbol{r}_{k+1} - \boldsymbol{r}_k - \Delta t \, \frac{\boldsymbol{g}(\boldsymbol{r}_k) + \boldsymbol{g}(\boldsymbol{r}_{k+1})}{2}. \tag{6}$$

The Newton–Raphson method can then be expressed as

$$\boldsymbol{r}_{k+1}^{j+1} = \boldsymbol{r}_{k+1}^{j} + \Delta \boldsymbol{r}_{k+1}^{j}, \tag{7}$$

where superscripts again represent iteration number, and $\Delta \boldsymbol{r}_{k+1}^{j}$ is the solution of the linear system

$$\boldsymbol{J}(\boldsymbol{r}_{k+1}^{j}) \, \Delta \boldsymbol{r}_{k+1}^{j} = -\mathcal{F}(\boldsymbol{r}_{k+1}^{j}). \tag{8}$$

$\boldsymbol{J}$ is the Jacobian matrix, whose components can be expressed as

$$J_{mn}(\boldsymbol{r}) = \frac{\partial \mathcal{F}_m}{\partial r_n} = \delta_{mn} - \frac{\Delta t}{2} \frac{\partial g_m(\boldsymbol{r})}{\partial r_n}. \tag{9}$$

Equation (9) shows that the Jacobian, $\boldsymbol{J}$, is the identity matrix (Kronecker delta) minus $\Delta t/2$ times the Jacobian $\boldsymbol{J}^{\mathrm{mob}}$ of the mobility function, $\boldsymbol{g}(\cdot)$, in equation (1),

$$J_{mn}^{\mathrm{mob}}(\boldsymbol{r}) \equiv \frac{\partial g_m(\boldsymbol{r})}{\partial r_n}. \tag{10}$$

The fact that the Jacobian needs to be calculated is the main source of difficulty for implicit methods. In this work, the Jacobians were calculated using a combination of analytical and numerical methods (see appendix B). To solve equation (8), the mldivide function (i.e. the '\' operator) in MATLAB was used. The cost of these operations was confirmed to be negligible in comparison to the dislocation force calculations in the test cases considered here. In general, the cost of solving these linear systems using direct methods can become large, however, making iterative solution methods desirable. A more detailed study of this cost is the subject of future work. The solution error at each iteration is taken to be the norm of the residual vector $\mathcal{F}$ for each node. For the first iteration of the method, the solution from the previous time step is used, since an initial guess made with an explicit method could be unstable and inaccurate. Finally, we emphasize that even though the Heun scheme also uses the trapezoidal approximation, it is a conditionally stable method; the trapezoidal method, equation (6), must be solved directly using a technique like the Newton–Raphson method to achieve unconditional stability [26].

2.5. Subcycling

In order to maintain efficiency with jogged FR source simulations, a time integration scheme known as subcycling will need to be implemented. Subcycling is an approach for handling problems with spatially disparate time-step sizes developed by the structural dynamics community [33–35]. The basic idea is to allow different degrees of freedom in the system to progress with different time-step sizes. Each element or node (depending on the partitioning scheme) is allowed to progress with its own time-step size, usually selected based on elemental stability conditions. In DD, however, these maximum stable (or sufficiently accurate) time steps are not known *a priori*. Instead, in DD simulations we usually allow the integrator to automatically adjust the time step to satisfy the specified error tolerance [9] (see section 2.2). Algorithms of this sort have been implemented in DD before (termed 'sub-stepping' [36] and 'under-integration' [37]) motivated by the observation that generally a small fraction of the nodes in a simulation, often those experiencing close-ranged interactions and/or in stable configurations, require very small time steps. Here we will present a new subcycling algorithm and perform a detailed evaluation of its performance.

Our implementation begins with the classification of the nodes of a simulation into two groups. We denote the nodes to be subcycled as group 1 and the remaining nodes as group 0. The 'global' time step corresponds to that of group 0. Group membership will be determined at the beginning of every global time step according to the proximity of dislocation segments. If two disconnected segments are within some radius, r_g, of each other, then their nodes will be added to group 1. In principle, group assignments can be made using a number of other criteria, such as the local error, but we here choose this approach for simplicity. We believe this grouping algorithm can be justified using stability arguments, but we save this discussion for a future study.

Because dislocation segment interactions are long-ranged and every degree-of-freedom interacts with every other degree-of-freedom, we need to manipulate the mobility function operator, $\boldsymbol{g}(\cdot)$, to enable subcycling. We employ the approach of *operator-splitting* [38], whereby the differential (mobility law) operator is split into two (or more) separate operators which are then solved separately with the solutions coupled in some way. We split the governing ODE, equation (1), into the form

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{g}_0(\{\boldsymbol{r}\}) + \boldsymbol{g}_1(\{\boldsymbol{r}\}), \tag{11}$$

where $\boldsymbol{g}_0(\cdot)$ only contains velocities for group 0 nodes (the velocities for group 1 nodes are zero), and $\boldsymbol{g}_1(\cdot)$ only contains velocities for group 1 nodes. We then solve this system as follows:

$$\frac{\mathrm{d}\widehat{\boldsymbol{r}}}{\mathrm{d}t} = \boldsymbol{g}_0(\{\widehat{\boldsymbol{r}}\}),\ t \in [t_n, t_{n+1}] \qquad \text{with } \widehat{\boldsymbol{r}}(t_n) = \boldsymbol{r}_n \tag{12}$$

$$\frac{\mathrm{d}\overline{\boldsymbol{r}}}{\mathrm{d}t} = \boldsymbol{g}_1(\{\overline{\boldsymbol{r}}\}),\ t \in [t_n, t_{n+1}] \qquad \text{with } \overline{\boldsymbol{r}}(t_n) = \widehat{\boldsymbol{r}}_{n+1}. \tag{13}$$

This means that we first move group 0 nodes with group 1 nodes fixed-in-place, and then move group 1 nodes with group 0 fixed in their new positions. The solution $\overline{\boldsymbol{r}}_{n+1}$ is then accepted as the final solution, $\boldsymbol{r}_{n+1}$. This is called the *sequential operator-splitting method* [38]. Obviously, some error is introduced by solving the problem in this way, referred to as *splitting error*, but we will demonstrate that for our test case here this error is small.

Now, to apply subcycling, we first advance the nodes in group 0 using the chosen integrator by solving equation (12) with the global time step Δt. Then the nodes in group 1 are advanced with equation (13) using the same integrator but at a smaller time step for multiple steps (i.e. subcycles) until they catch up with the global clock set by group 0. The subcycle step

size changes dynamically based on the selected integration algorithm. To save computational expense, the forces exerted by group 0 on group 1 are calculated only at the beginning of a subcycle phase (one global time step). During the subcycles, only the forces due to interactions within group 1 are updated, leading to significant computational savings. This means that we are essentially using the forward Euler integrator for group 0 effects on group 1, making our subcycling approach a mixed method [34, 39], i.e. mixing different time integration schemes. We believe this approach is sound because, if our grouping algorithm is effective, the behavior of group 1 nodes will be dominated by their interactions with other group 1 nodes and no destabilizing modes will exist between group 0 and group 1.

For additional computational savings, we employ a 'forward progress check' on all group 1 nodes during subcycling. The idea is that often times the nodes in group 1 are in an equilibrium state (i.e. a stable dipole, as in the case of the jogged FR source) and oscillating back and forth due to numerical noise. Hence capturing the details of this behavior is neither physically meaningful nor numerically significant. We cannot, however, completely ignore these nodes (e.g. hold them fixed-in-place) in case the local stress is large enough to overcome this equilibrium. The goal of the forward progress test is to determine whether the nodes are merely oscillating around an equilibrium state or attempting to leave group 1. If a node reverses its direction of motion during subcycling (within a global time step), this node will be fixed-in-place during the remainder of the subcycles. The subcycling stops if all nodes in group 1 are fixed, even if the local clock has not caught up with the global clock.

3. Results

We now compare the performance of the four integrators introduced above in the two case studies, and discuss the algorithmic features that lead to higher efficiency. All simulations were conducted using the MATLAB code DDLab [28], which is a serial variant of the parallel C program ParaDiS [9, 12, 29].

3.1. Collapsing prismatic loop

3.1.1. Simulation setup. Consider a circular prismatic loop collapsing by climb due to its own elastic energy and core energy (line tension). A prismatic loop is chosen here for simplicity because, unlike a glide loop, all dislocation segments in a prismatic loop have edge character and the loop remains circular during the simulation. All dislocation segments in the loop have the same core energy per unit length, which is designated by $E_c/(1-\nu)$, that results in a line tension force experienced locally everywhere along the loop. Additionally, when elastic interactions are enabled, all segments experience Peach–Koehler forces due to the stress field of every segment, including themselves. The non-singular continuum theory of dislocations [13] is used to evaluate the resulting forces on the nodes. A linear mobility law is assumed such that the nodal velocity is given by [12]

$$\boldsymbol{v}_i = \frac{\boldsymbol{f}_i}{B \sum_j L_{ij}/2}, \tag{14}$$

where $\boldsymbol{f}_i$ is the net force vector on node i and the summation is over the (two) neighboring nodes connected to node i. A constant drag coefficient B is used (all segments have edge character and are moving by climb).

DD simulations can be performed for various ratios of core and elastic energies, which can be measured by the dimensionless parameter $\alpha = E_c/(\mu b^2)$. When the core radius is chosen to be $r_c/b = 1$, α is in the range of 0.5 to 1 for most metals [32]. For benchmarking

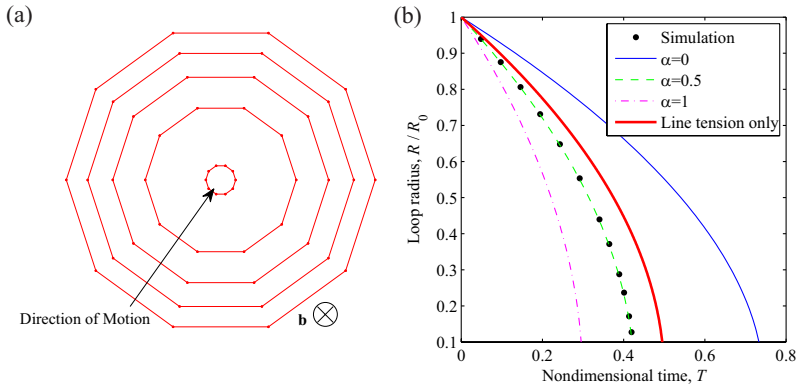

Figure 1. (*a*) Snapshots of a DD simulation of a circular prismatic loop collapsing by climb shown at intervals of about $\Delta t\, \mu b^2/[BR_0^2(1-\nu)] = 0.106$ with $\alpha = 0.5$. The circle is represented with 10 uniform segments. (*b*) Radius of a prismatic loop collapsing by climb as a function of nondimensional time, $T = t\, A/[BR_0^2(1-\nu)]$ where with elasticity enabled $A = \mu b^2$ and with line tension only $A = E_c$. Analytical solution shown for $\alpha = 0$ (no line tension), $\alpha = 0.5$, and $\alpha = 1$ with $r_c/b = 1$ and line tension only (thick line). Dots are the DD simulation results corresponding to (*a*). The initial radius for all is $R_0/b = 10^3$.

purposes, we have performed simulations with elastic energy only ($\alpha = 0$), line tension only ($\mu = 0$), and with both elastic energy and line tension present (e.g. $\alpha = 0.5$). Figure 1(*a*) plots snapshots of a DD simulation with $\alpha = 0.5$. Figure 1(*b*) plots the analytical solutions (given in appendix C) and a comparison with the DD prediction corresponding to figure 1(*a*).

3.1.2. Accuracy versus stability. When the collapsing prismatic loop is simulated by DD using an explicit integrator, the allowable time-step size can depend on both the error tolerance and the dislocation segment length. Figure 2(*a*) presents the number of solution time steps required for the prismatic loop of initial radius $R_0 = 10^3 b$ to shrink to 1/10 of its original radius using the Heun method as a function of the absolute integration tolerance, r_{tol}, when the loop is uniformly discretized into different numbers (N) of segments. For these calculations, the relative error tolerance, f_{tol}, was set to an arbitrarily large value and the error threshold, r_{th}, was set to zero. Because all segments have the same length (l), in this example there is a simple relationship between N and l, i.e. $N \approx 2\pi R/l$.

First, we note that the number of solution steps increases significantly as the number of segments increases (element size decreases). Secondly, with longer segments ($N = 10$) the number of solution steps decreases as r_{tol} is increased. This is an indication that the simulation accuracy is limiting the time step. As the segment size decreases, however, the curve slope lessens until the number of solution steps becomes insensitive to the tolerance ($N \geqslant 40$). Even when the integration tolerance is a significant fraction of the initial loop radius, the number of time steps still does not decrease. This signals the transition to a regime where the time step is limited by stability.

The maximum stable time step for an explicit integrator can be expressed as (when all eigenvalues are real)

$$\Delta t^{\text{s}}_{\text{max}} = \frac{c_{\text{int}}}{|\lambda|_{\text{max}}}, \tag{15}$$

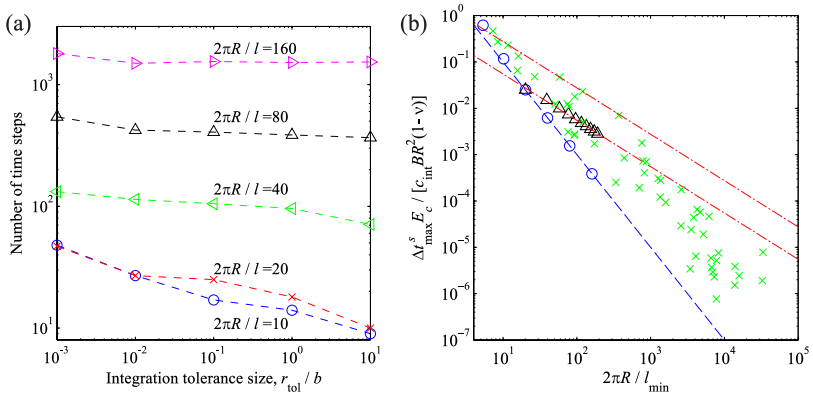

Figure 2. (*a*) Number of solution time steps for a prismatic loop collapsing from the initial radius $R_0 = 10^3\, b$ to the final radius $R_f = 0.1R_0$, as a function of absolute integration tolerance size, with $\alpha = 0.5$ and different numbers of elements. (*b*) Maximum stable time-step sizes for a prismatic loop with line tension only as a function of the smallest segment length; ○ are for a loop with uniformly sized segments, × for randomly sized segments, and △ for a loop with 19 uniformly sized segments and 1 small segment. The dashed line is a fit assuming $\Delta t^s_{max} \propto (2\pi R/l_{min})^{-2}$ and the dashed–dotted lines are fits assuming $\Delta t^s_{max} \propto (2\pi R/l_{min})^{-1}$.

where $|\lambda|_{max}$ is the maximum eigenvalue magnitude of the Jacobian $\boldsymbol{J}^{mob}$ of the mobility function $\boldsymbol{g}(\cdot)$, and c_{int} is an integrator-specific constant. For instance, with the Heun and RKF integrators, the theoretical values[3] of c_{int} are 2 [26] and 3.7 [27].

To examine the stability limit of DD simulations of a collapsing prismatic loop, we calculated the eigenvalues of $\boldsymbol{J}^{mob}$ (see appendix B) and the corresponding maximum stable time steps Δt^s_{max}. These calculations were performed for loops represented both by segments of uniform length and by segments of randomly chosen lengths. The eigenmode paired with $|\lambda|_{max}$ always corresponds to a radial, zig-zag motion, as was previously observed by [20] in their figure 4. Figure 2(*b*) presents the results with line tension only. For uniformly discretized loops, the resulting Δt^s_{max}, when expressed in dimensionless form, is found to follow the relationship (with line tension only)

$$\frac{\Delta t^s_{max}}{c_{int} B R^2 (1-\nu)/E_c} = 10\left(\frac{2\pi R}{l}\right)^{-2}. \tag{16}$$

This is shown by the circles and dashed line in figure 2(*b*). Furthermore, when all segments in the loop are equal in length except for one short segment of length l_{min}, the maximum stable time step is found to instead scale with $(2\pi R/l_{min})^{-1}$, as shown by the triangles and dashed–dotted line in figure 2(*b*). For loops discretized into segments of random lengths, the dependence of Δt^s_{max} appears to be in between $(2\pi R/l_{min})^{-1}$ and $(2\pi R/l_{min})^{-2}$. These same scaling trends are observed when elastic interactions are accounted for as well.

Given Δt^s_{max}, we can estimate the number of solution steps needed for an explicit integrator as a function of the smallest element size. When the time step is always stability limited, we find that for a shrinking prismatic loop described by the line-tension-only model and discretized

[3] Stability is most often analysed with a first order, linear, homogeneous ODE. Since any problem can be linearized (and diagonalized, if possible), this provides a fairly universal assessment of stability.

by uniform segments, the necessary number of solution steps is (see appendix D)

$$N_{\text{sol}} = \ln\left(\frac{R_0}{R_{\text{f}}}\right)\left[\frac{1}{10\,c_{\text{int}}}\left(\frac{2\pi R}{l}\right)^2 - 1\right] + \mathcal{O}\left(\left[\frac{2\pi R}{l}\right]^{-2}\right), \tag{17}$$

where R_{f} is the final radius. Notice that $2\pi R/l$ remains unchanged during the simulation, since we do not allow remeshing. Equation (17) predicts that the number of solution steps increases quadratically with $2\pi R/l$ for explicit integrators in the stability-limited regime.

3.1.3. Comparison among integrators. We first compare the performance of all four integrators, Heun, RKF, BS and implicit, for the line-tension-only model of the dislocation loop (i.e. no elasticity). Figure 3(*a*) plots the number of solution time steps as a function of the inverse of element size for all four of the integrators with $R_0 = 10^3\,b$ and $R_{\text{f}} = 0.1\,R_0$. The predictions of equation (17) are also plotted for the explicit integrators, with c_{int} values chosen as empirical fitting parameters; we found that $c_{\text{int}} = 2.33$, 4, and 4.9 for Heun, RKF, and BS, respectively, provide excellent fits when the number of segments is large. Note that the values for the Heun and RKF methods are very close to the theoretical values. Because equation (17) gives an accurate description of the number of solution steps as a function of segment length, in conjunction with the evidence in figure 2(*a*), we conclude that under these conditions the time steps are stability limited. Furthermore, for the implicit integrator the number of solution steps is almost entirely insensitive to the element size, confirming that it better tolerates the introduction of unstable, stiff modes.

Even though the number of solution steps enables a qualitative comparison between integrators, it is not an accurate measure of computational cost because different integrators require differing amounts of computation per time step. Higher order methods such as RKF and BS perform more calculations (calls to function $\boldsymbol{g}(\cdot)$) per step than the lower order Heun method. Also, an implicit method usually requires more calculations per step than an explicit method. To compare the costs, we have plotted the number of force calculations (calls to function $\boldsymbol{g}(\cdot)$) as a function of $2\pi R/l$ in figure 3(*b*). It can be seen that with coarser discretization (smaller $2\pi R/l$) the explicit methods are competitive. However, as the discretization of the loop is refined the implicit solver scales much more favorably and eventually becomes more efficient, with the cross-over point at about $2\pi R/l = 30$, which is close to the boundary of the stability-limited regime observed in figure 2(*a*). It is also worth noting that in the stability-limited regime (where the implicit method is the most efficient), the lower order Heun method performs the best among the explicit methods.

Similar conclusions can be made when elasticity is turned on in the DD simulations. Figures 3(*c*) and (*d*) present the same data as above with $\alpha = 0.5$. The computational cost is now measured in terms of the number of calls to segment–segment interactions. (To find the force on all nodes in a loop discretized into N segments requires $\mathcal{O}(N^2)$ calculations of segment–segment interactions.) When long-range elastic interaction is allowed, the full Jacobian matrix $\boldsymbol{J}^{\text{mob}}$ is so expensive to evaluate that it can make the implicit integrator less efficient than the explicit integrators. Therefore, we have used the Jacobian derived from the line-tension-only model as an approximation to the true Jacobian. Unfortunately, such an approximation deteriorates the efficiency of the implicit integrator, making it only slightly better than the explicit integrators, even in the stability-limited regime (see figure 3(*d*)). However, we found that by accounting for near-neighbor elastic interactions in the Jacobian (which incurs a small computational cost, which is accounted for in figure 3(*d*)), the efficiency of the implicit method is dramatically improved, as shown by the stars in figures 3(*c*) and (*d*). Elastic interaction terms were added to the Jacobian if two segments were within the *enhancement*

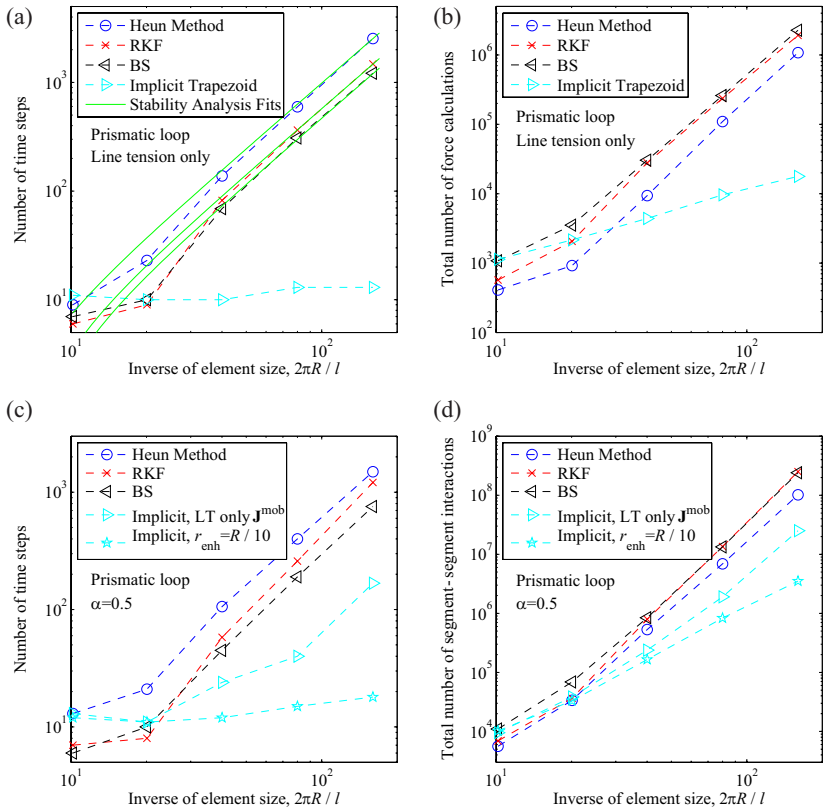

Figure 3. Comparison of (*a*), (*c*) number of time steps and (*b*), (*d*) number of force calculations as a function of inverse element size with a collapsing prismatic loop for all four solvers. (*a*) and (*b*) are with line tension only and (*c*) and (*d*) are with $\alpha = 0.5$.

radius, r_{enh}, of each other, here set to $r_{\text{enh}} = R/10$. When the implicit method uses an approximated Jacobian containing near-neighbor elastic interactions, the computational cost shows a more favorable scaling behavior than the explicit methods, though the improvement is less dramatic than that with the line-tension-only model. Including more elastic interactions in the approximated Jacobian does not seem to improve the efficiency of the implicit method.

3.2. Jogged FR source

3.2.1. Simulation setup. To evaluate integrator performance in a more complex scenario, we choose the geometry of a centrally jogged FR source of length L and height h that is initially of edge character, as depicted in figure 4(*a*). The jogged FR source provides three important features for this study: (1) it prevents collision and annihilation (which are beyond the scope of this study) even when it is fully activated; (2) if the jog height is sufficiently small a stable dipole may form, allowing the evaluation of integrator performance with closely approaching segments; and (3) the jog is a physical element that cannot be removed by a remeshing algorithm. The last point is significant because small physical segments do appear during large-scale bulk work hardening simulations and are believed to have an adverse effect on the time step. Of course when the jog height h is zero the geometry of a traditional, flat FR source is recovered.

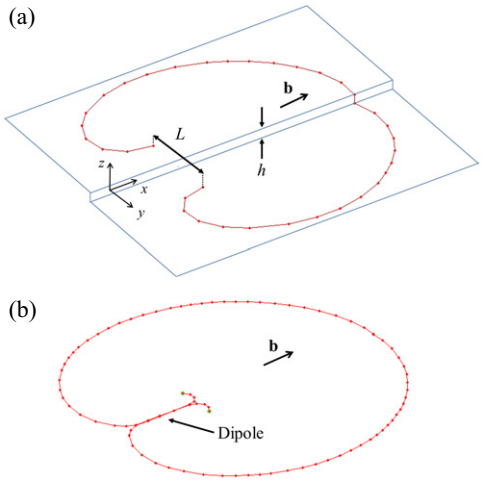

Figure 4. A jogged FR source before (*a*) and after (*b*) a dipole has formed. The FR source in (*a*) and (*b*) have different jog heights and are not shown to the same scale.

As shown in figure 4(*a*), the Burgers vector of the FR source is along the x-axis. Glide occurs in the x-y plane for the source arms and in the x–z plane for the jog. The nodes at either end of the jog are constrained to move only in the x-direction.

In all simulations unless stated otherwise, a constant shear stress of $\tau_{xz}/\mu = 0.0075$ is applied, and both core energy and elastic interactions are included with $\alpha = 0.5$. Remeshing is enabled in all simulations to control the length of all discretization segments to be within $l_{\min}$ and $l_{\max}$, and to control the area of all triangles spanned by neighboring segments to be within $A_{\min}$ and $A_{\max}$ [9, 12]. The remaining simulation parameters are $L/b = 500$, $r_c/b = 1$ and $\nu = 0.3$. Given $l_{\max}$ and r_{tol}, the other three remeshing parameters are determined by the following relationships: $l_{\max}/l_{\min} = 4$, $A_{\min} = 2r_{\text{tol}}\, l_{\max}$ and $A_{\max} = 2A_{\min} + (\sqrt{3}/8)\, l_{\max}^2$.

3.2.2. Before dipole formation. We will first consider the behavior of the FR source before the arms complete a full revolution and before the possibility of a stable dipole forming. Examining this case allows us to compare the integrator performances between the (expanding) FR source and the (collapsing) prismatic loop in section 3.1. The simulations here are conducted for a duration of $t\,\mu/B = 3 \times 10^5$ which corresponds to approximately 270° of revolution of the arms (figure 4(*a*)).

First we consider a traditional, jog-free FR source ($h = 0$). Figure 5 presents the plastic strain due to the bowing out of the source at different values of $L/l_{\max}$, normalized by the average (across all five data sets) of the solutions with the smallest segment size, ε_o. It can be seen that all integrators converge to approximately the same value as the segments are refined, indicating that the chosen error tolerances are reasonable. Figure 6(*a*) compares the number of solution steps using different integrators as a function of $L/l_{\max}$. Figure 6(*b*) compares the computational cost of the different integrators. In contrast to the collapsing loop simulations, the implicit method is most efficient regardless of element size, even when the Jacobian includes only the line tension terms. When the elastic interactions between segments within $r_{\text{enh}} = L/4$ are included in the approximated Jacobian, the efficiency of the implicit method is further improved, as shown by the stars in figure 6(*b*). Again, the addition of further elastic interactions in the Jacobian did not provide a performance gain. DD simulations with a relatively large jog height of $h = L/32$ show similar trends to the jog-free case shown here.

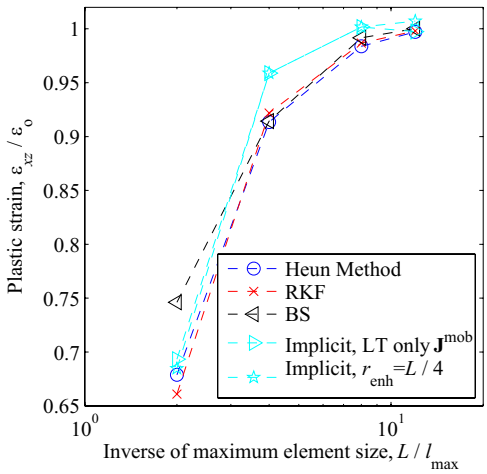

Figure 5. Plastic strain due to bowing out of the FR source as a function of the maximum segment size for all four integrators. For the implicit solver, results with a line-tension-only Jacobian and an elasticity enhanced Jacobian with $r_{\text{enh}} = L/4$ are shown. All data are normalized by the average plastic strain across all results with the smallest segment size, ε_o.

Next we consider a FR source containing a short jog. Figure 6(*c*) shows how the jog height affects the number of solution steps of each solver with jog heights from $h = L/16$ down to $h = L/1024$ and with L/l_{max} kept at 2. Figure 6(*d*) compares the computational cost in terms of the number of segment–segment interaction calculations. For all of the explicit integrators, both the number of solution steps and the computational cost increase linearly with the inverse of the jog height. This linear scaling is consistent with the case of a collapsing prismatic loop when only one segment is of a shorter length (figure 2(*b*)); here the jog has a much shorter length than all other segments. In this case, the jog introduces a destabilizing mode where the nodes at either end of the jog zig-zag forward and backward rapidly, when using an explicit integrator with a large time step. The most important conclusion from figures 6(*c*) and (*d*) is that the performance of the implicit integrator is unaffected by the jog height h. Furthermore, this performance can be achieved with a line-tension-only Jacobian, indicating that this unstable mode is dominated by line tension. The implicit method clearly out-performs all the explicit methods in this regime.

3.2.3. After dipole formation. After a jogged FR source is activated, a stable dipole can form when the two arms gliding on parallel planes meet, provided that h is small enough, as shown in figure 4(*b*). Due to strong and rapidly varying short-range elastic interactions between segments in the dipole, the dipole formation is expected to introduce a stiff mode and reduce the simulation time step[4]. In the following, we show that the dipole formation has a severe impact on both the explicit and implicit integrators considered here, so that further algorithmic development (e.g. subcycling) is needed to restore computational efficiency.

To evaluate the effects of dipole formation we extend the simulation time to $t\,\mu/B = 4.6 \times 10^5$. Figure 7(*a*) shows that, as soon as the dipole forms at about $t\,\mu/B = 4.2 \times 10^5$, the simulation time-step drops precipitously for all the integrators considered here. The time-step

[4] An eigenvalue analysis can easily be performed using linear elasticity theory of straight dislocations to demonstrate the maximum stable time step and resulting stiffness of a dipole.

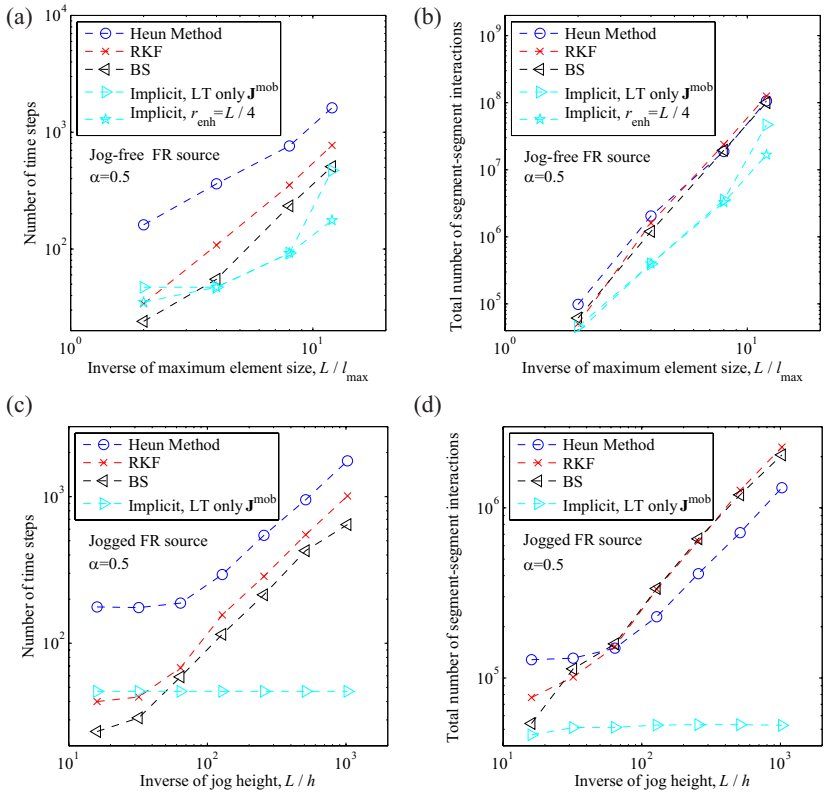

Figure 6. Comparison of (*a*) number of time steps and (*b*) number of segment–segment interactions as a function of inverse maximum element size with a jog-free FR source for all four solvers. For the implicit solver, results with a line tension only Jacobian and an elasticity enhanced Jacobian with $r_{\text{enh}} = L/4$ are shown. Comparison of (*c*) number of time steps and (*d*) number of segment–segment interactions as a function of inverse jog height with a jogged FR source for all four solvers. L/l_{max} was kept constant at 2 for these simulations. Implicit results with a line-tension-only Jacobian are shown.

drop with the implicit method when using the line-tension-only Jacobian (LT) is as significant as the explicit methods. This is not surprising because the Jacobian does not contain any information on the strong interaction between the dipole, which is the cause of the time-step drop here. When the Jacobian includes interactions between segments within $r_{\text{enh}} = L/4$ of each other (LT+E), large enough to include the dipole, the time step of the implicit integrator is improved. However, even in this case, the time step is still lower than the values before the dipole formation, and also suffers from periodic drops to even lower values. These drops coincide with the entrance of new segments into the dipole. We have confirmed that some amount of this time-step reduction with the implicit integrator is due to accuracy limitation since when the relative tolerance is increased to $f_{\text{tol}}/b = 0.25$ the time-step reduction is lessened (see figure 7(*a*), this is in contrast to increasing f_{tol} with explicit methods (not shown) where the time-step drop is unchanged), but stability does appear to be a factor as well.

Table 1 compares the computational cost of all four integrators in units of millions of segment–segment interaction calculations during the period after the dipole has formed with two different dipole heights. Without subcycling, the Heun method is most efficient with the

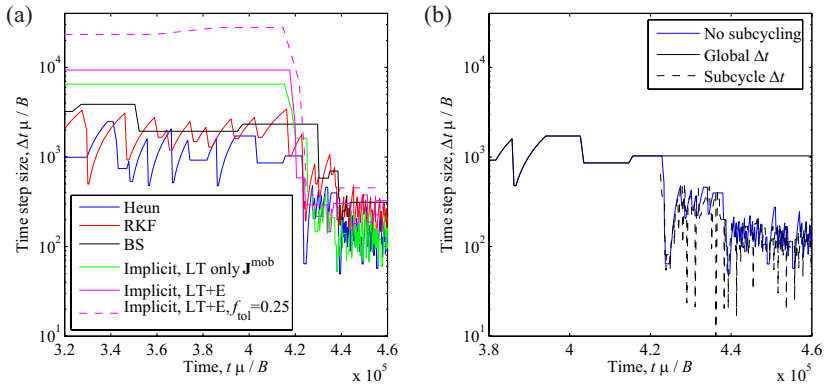

Figure 7. (*a*) Time-step size as a function of time for a jogged FR source with $L/h = 128$ and $L/l_{\max} = 2$. When the dipole forms, the time-step drops with all integrators. Results from three versions of implicit methods are shown here: (1) using line-tension-only Jacobian (LT), (2) including some elastic interaction terms in the Jacobian (LT+E), and (3) using an LT+E Jacobian with $f_{\text{tol}}/b = 0.25$. An enhancement radius of $r_{\text{enh}} = L/4$ was used for both LT+E calculations. (*b*) Time-step size as a function of time using the Heun method with and without subcycling. The subcycling algorithm clearly partitions the nodes based on time-step size.

Table 1. Comparison of dipole computational costs (total cost after dipole forms) in millions of segment–segment interactions from jogged FR source simulations. Two implicit methods are tested. The first one uses an approximated Jacobian based on the line-tension (LT) model. The second one (LT+E) includes the elastic interactions between segments within a distance of $r_{\text{enh}} = L/4$ of each other. For the subcycling results, a grouping radius of $r_{\text{g}} = 8h$ was used for both Heun method calculations, $r_{\text{g}} = 20h$ for implicit LT+E with $L/h = 128$, and $r_{\text{g}} = 30h$ for implicit LT+E with $L/h = 256$.

	Without subcycling					With subcycling	
	Heun	RKF	BS	Implicit with LT Jacobian	Implicit with LT+E Jacobian	Heun	Implicit with LT+E Jacobian
$L/h = 128$	4.08	5.46	6.44	7.65	5.04	0.69	0.23
$L/h = 256$	14.0	22.9	36.0	26.5	7.65	1.28	0.26

taller jog and implicit LT+E with the shorter jog. We note that halving the jog height causes the cost with the explicit and implicit with LT Jacobian methods to increase by a factor of between 3 and 6, but results in only a modest increase using implicit with LT+E Jacobian. This is because while the other methods are stability limited by the dipole, the implicit method (with LT+E Jacobian) is accuracy limited.

3.2.4. Using subcycling. Given that the time-step drop is observed even with the implicit method, as shown in figure 7(*a*), it alone does not completely address the efficiency challenge created by the dipole formation. We therefore seek an additional improvement for our time integration algorithm to further increase computational efficiency, and for this we turn to subcycling (section 2.5).

For the jogged FR source simulation, we wish to assign all nodes in the dipole to group 1 (subcycling group) and all remaining nodes to group 0 (global group). The idea here is that since only the nodes in the dipole require small time steps, we should isolate them from the rest of the system. This grouping is accomplished using the proximity condition discussed in section 2.5, with the grouping radius, r_g, selected so that the global time step is unaffected by the dipole formation; specific r_g values were determined by trial and error.

To demonstrate the effectiveness of this subcycling algorithm, DD simulations were performed with both the Heun and implicit methods. The resulting global and subcycle time-step sizes using the Heun method are presented in figure 7(*b*) along with the result without subcycling. Even after dipole formation, the global time step stays at the same level as before. The local time steps for group 1 are about the same as the (global) time steps when subcycling is not applied, as in figure 7(*a*). (The time-step size of the last subcycle may be much smaller, only to prevent the clock of group 1 from going beyond the clock of group 0.) However, the subcycling results in a tremendous efficiency gain because during the subcycling only local interactions among the nodes in group 1 are updated. To quantify the efficiency of subcycling, we have calculated the fraction of the computational cost spent on subcycling. With the Heun method at the end of the simulation when the dipole is longest, 16% of the nodes belong to group 1 and they incur only 25% of the total segment–segment interaction cost.

Table 1 provides a comparison of the computational cost with and without subcycling when either the Heun or the implicit integrator with LT+E Jacobian is used. Subcycling yields a tremendous computational savings, reducing the cost by a factor of between 11 and 30. The implicit method using the LT+E Jacobian combined with subcycling is the most efficient approach. We also point out that even with subcycling, the cost with the Heun method doubles when the jog height is halved—this is the same effect (from jog oscillation) that was observed before in figure 6(*d*). Again, the implicit solver is insensitive to the jog height change making the cost with subcycling constant. This constant cost, independent of minor discretization details, indicates a robust integration algorithm. We can conclude that most of the computational gain in the present case of dipole formation is due to subcycling.

Finally, we have investigated the effect of subcycling and operator-splitting on jogged FR source simulation accuracy. We have run a variety of simulations using the Heun method examining various dipole behaviors when $L/h = 128$. Firstly, figure 8(*a*) displays the dipole length, defined as the distance between where the arms cross each other, as a function of time with the simulation parameters discussed above, with and without subcycling. It can be seen that the subcycling curve very closely matches the result from brute-force integration without subcycling. Next, we conducted simulations using the fully formed dipole as the initial conditions and increased the applied stress to $\tau_{xz}/\mu = 0.03$. This caused the dipole to break apart. We have plotted the length (l_d) and width (w_d) spanned by the dipole for this case as a function of time with and without subcycling in figure 8(*b*). Again, subcycling accurately captures the dipole behavior. Finally, we have conducted a study to see at what stress the FR source arms overshoot each other rather than forming a stable dipole. A series of FR source simulations were conducted with the applied stress slowly increased until a stable dipole no longer formed—this was deemed the dipole threshold stress. We found that the threshold stresses with and without subcycling were $\tau_{xz}/\mu = 0.010\,60$ and $0.010\,82$, respectively. The error introduced by subcycling is about 2%. We conclude that the subcycling algorithm presented above does not introduce significant errors or artifacts into the jogged FR source simulations.

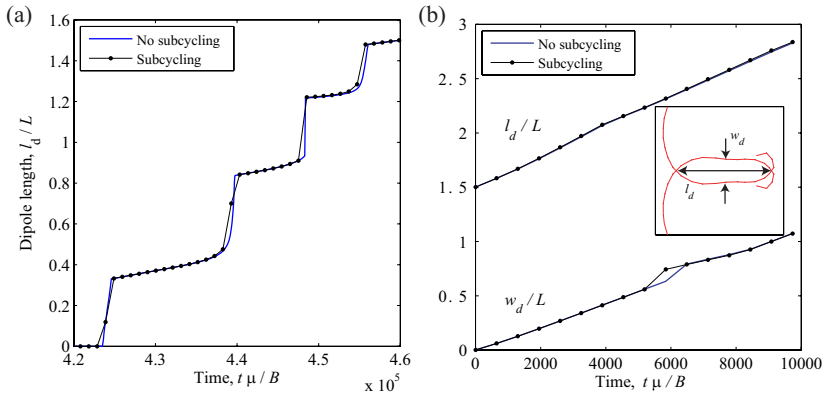

Figure 8. Comparison between jogged FR source dipole behavior with and without subycling using the Heun method. (a) Dipole length versus time as the dipole forms with an applied stress of $\tau_{xz}/\mu = 0.0075$. (b) Dipole length (l_d) and width (w_d) versus time as a preformed dipole breaks with an applied stress of $\tau_{xz}/\mu = 0.03$. The grouping radius for subcycling is $r_\mathrm{g} = 8h$. The other parameters are $L/h = 128$, $L/l_\mathrm{max} = 2$, and $r_\mathrm{tol}/b = 0.1$. Data with subcycling is shown at global time steps only.

4. Discussion

We have demonstrated multiple mechanisms that can dramatically reduce the computational efficiencies of standard explicit integrators in line DD simulations. There is no single solution that can effectively respond to all of these mechanisms. This is perhaps why efficient time integration in DD simulations of work hardening has remained a significant challenge to date. We believe the efficiency-limiting mechanisms identified in the two test cases considered here are representative of those in large-scale work hardening simulations. They include: zig-zag oscillation of discretization nodes on a dislocation arm, oscillation of a jog or any short segment that cannot be removed by remeshing, and the relative motion of two dislocations in a dipole.

Each of the efficiency-limiting mechanisms has a characteristic length scale, e.g. the shortest segment comprising the dislocation arm, the length of the jog, and the height of the dipole. When this characteristic length scale is sufficiently small, the time-step size with an explicit integrator becomes insensitive to the specified error tolerance. Instead, it is solely controlled by this length scale and the simulation is said to have entered the stability-limited regime.

The implicit (trapezoid) integrator exhibits superior efficiency in the stability-limited regime, provided that its Jacobian has the information relevant to any destabilizing modes that are present (i.e. line tension or short-range elastic interactions). In the small test cases studied here, we have shown that the computational cost for constructing this approximated Jacobian is relatively small, and the memory requirement is also small because it is sparse. For large-scale simulations, because only line tension and near-neighbor elastic interaction terms need to be included to gain maximum efficiency, this cost is only expected to grow as $\mathcal{O}(N)$. Therefore we expect that an implicit integrator can improve the efficiency of large-scale DD simulations of work hardening, assuming that these simulations are stability-limited by the presence of short segments. The progressive reduction of average time-step sizes observed in existing ParaDiS simulations with increasing strain correlates with the production of short segments, supporting the hypothesis that these simulations are indeed stability-limited.

However, an implicit integrator alone is not expected to deliver optimal efficiency, as is demonstrated by the jogged FR source forming a stable dipole. Dislocations are expected to form relatively stable complexes (such as dipoles and multi-junctions [15]) during work hardening simulations. The nodal oscillations within these complexes are of minor importance to the overall behavior of the dislocation ensemble as long as they remain stable. Yet both explicit and implicit methods require much smaller time steps to integrate the equations of motion of these nodes than that necessary for the other remaining nodes. An effective method to handle this situation is subcycling, where the nodes in the stable complexes are grouped together and allowed to take multiple smaller, local time steps, while the remaining nodes take a single larger, global time step. In terms of large-scale work hardening simulations, the subcycling method looks promising. The sophistication of the algorithm will likely have to be increased to maximize the efficiency gain, however (e.g. finer scale grouping for specific segment pairs). A general algorithm that can effectively identify local nodal clusters that require smaller time steps must also be devised. Finally, subcycling will have to be made compatible with any parallelized code structures, with care taken to keep computational loads balanced across processors.

There is some concern in the literature that remeshing, i.e. adding or removing discretization nodes, can introduce error and limit the time-step size [9]. For example, a new node is usually added at the midpoint of an existing segment, while a more accurate location may be away from the midpoint to better represent a curved dislocation arm. To address this concern, we performed a simple study with the Heun method comparing the iteration and convergence behavior during jogged FR source time steps with and without a remeshing event having just occurred. We found that without remeshing the iteration of equation (3) required more than one iteration when convergent (no need to cut the time step) for 4% of the steps and experienced time-step cuts for 18% of the steps. In contrast, immediately after remeshing, iteration was required (didn't converge after the first iteration) during 68% of time steps and the time step was reduced for 29% of the steps to achieve convergence. For reference, remeshing occurred during 6% of the time steps. Furthermore, by comparing analytical and simulation solutions with the collapsing prismatic loop we have observed large error spikes of an order of magnitude immediately following remesh events. We therefore conclude that a remeshing event can locally increase computational cost, but the effect is not at the same scale as the other mechanisms we have elucidated here.

5. Conclusions

We have compared the efficiencies of three explicit integrators (Heun, RKF, BS) and one implicit integrator (trapezoid) in DD simulations of the collapse of a prismatic loop and the activation of a jogged FR source. With all explicit methods, the time-step size reduces with decreasing dislocation segment lengths, due to an increased system stiffness. The performance of the implicit integrator is less sensitive to these unstable modes, and in one case (jog oscillation) completely immune to the effect. The Jacobian needed to achieve maximal efficiency with the implicit integrator can be constructed by taking account of only short-range elastic and core interactions, thus without appreciable increase of computational cost. On the other hand, the formation of a stable dipole presents challenges to both explicit and implicit methods. Computational efficiency is restored by implementing a time-step subcycling approach, in which a small cluster of nodes takes smaller time steps than the global time step used for the remaining nodes. We believe that the application of the implicit integrator and subcycling method explored here in large-scale DD simulations can significantly improve the

efficiency of work hardening simulations and ultimately lead to the higher strains relevant to engineering applications.

Acknowledgments

This work was supported by the US Department of Energy, Office of Basic Energy Sciences, Division of Materials Sciences and Engineering under Award No. DE-SC0010412 (WC), and by Sandia National Laboratories (RBS). Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the US Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. We would like to thank William P Kuykendall for conducting two-dimensional DD simulations in support of the stability analysis of dipoles. We thank Dr A Arsenlis at Lawrence Livermore National Laboratory for useful discussions.

Appendix A. Integrator details

A.1. RKF method

The RKF method [24] is fifth-order accurate and uses the following algorithm:

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k + \Delta t \left(\frac{16}{135}\boldsymbol{q}^1 + \frac{6656}{12825}\boldsymbol{q}^3 + \frac{28561}{56430}\boldsymbol{q}^4 - \frac{9}{50}\boldsymbol{q}^5 + \frac{2}{55}\boldsymbol{q}^6 \right) \tag{A.1}$$

$$e_i = \left\| \Delta t \left(\frac{1}{360}(\boldsymbol{q}^1)_i - \frac{128}{4275}(\boldsymbol{q}^3)_i - \frac{2197}{75240}(\boldsymbol{q}^4)_i + \frac{1}{50}(\boldsymbol{q}^5)_i + \frac{2}{55}(\boldsymbol{q}^6)_i \right) \right\| \tag{A.2}$$

where e_i is an estimate of truncation error for node i, and

$$\begin{aligned}
\boldsymbol{q}^1 &= \boldsymbol{g}\left(\boldsymbol{r}_k\right) \\
\boldsymbol{q}^2 &= \boldsymbol{g}\left(\boldsymbol{r}_k + \frac{\Delta t \boldsymbol{q}^1}{4}\right) \\
\boldsymbol{q}^3 &= \boldsymbol{g}\left(\boldsymbol{r}_k + \Delta t\left(\frac{3}{32}\boldsymbol{q}^1 + \frac{9}{32}\boldsymbol{q}^2\right)\right) \\
\boldsymbol{q}^4 &= \boldsymbol{g}\left(\boldsymbol{r}_k + \Delta t\left(\frac{1932}{2197}\boldsymbol{q}^1 - \frac{7200}{2197}\boldsymbol{q}^2 + \frac{7296}{2197}\boldsymbol{q}^3\right)\right) \\
\boldsymbol{q}^5 &= \boldsymbol{g}\left(\boldsymbol{r}_k + \Delta t\left(\frac{439}{216}\boldsymbol{q}^1 - 8\boldsymbol{q}^2 + \frac{3680}{513}\boldsymbol{q}^3 - \frac{845}{4104}\boldsymbol{q}^4\right)\right) \\
\boldsymbol{q}^6 &= \boldsymbol{g}\left(\boldsymbol{r}_k + \Delta t\left(-\frac{8}{27}\boldsymbol{q}^1 + 2\boldsymbol{q}^2 - \frac{3544}{2565}\boldsymbol{q}^3 + \frac{1859}{4104}\boldsymbol{q}^4 - \frac{11}{40}\boldsymbol{q}^4\right)\right).
\end{aligned}$$

Each $\boldsymbol{q}^i$ is incrementally calculated to gradually build-up the final solution and error estimates. If there were explicit time dependence in the mobility function $\boldsymbol{g}(\cdot)$, that would also be reflected in the $\boldsymbol{q}^i$ expressions.

A.2. BS method

The BS method adopted here uses the modified midpoint method. Consider finding the solution from the current time t_k to time t_{k+1} and breaking this interval into $m = 2^n$ equal subintervals of size h, where n is an integer:

$$h = \frac{t_{k+1} - t_k}{m} \tag{A.3}$$

The modified midpoint method is then defined as follows:

$$\boldsymbol{r}^1 = \boldsymbol{r}^0 + h\,\boldsymbol{g}(\boldsymbol{r}^0) \tag{A.4}$$

$$\boldsymbol{r}^{j+1} = \boldsymbol{r}^{j-1} + 2h\,\boldsymbol{g}(\boldsymbol{r}^j), \qquad 1 \leqslant j < m \tag{A.5}$$

$$\boldsymbol{r}^f = \frac{\boldsymbol{r}^m}{2} + \frac{\boldsymbol{r}^{m-1} + h\,\boldsymbol{g}(\boldsymbol{r}^m)}{2} \tag{A.6}$$

where $\boldsymbol{r}^0 = \boldsymbol{r}(t_k)$ and $\boldsymbol{r}^f = \boldsymbol{r}(t_{k+1})$. The estimate in the first subinterval uses the forward Euler method (equation (A.4)). Estimates over subsequent subintervals use the midpoint method (equation (A.5)). The final estimate, $\boldsymbol{r}^f$, is the average of the last midpoint estimate and an estimate using the backward Euler method (equation (A.6)). These modified midpoint solutions at different n values are then extrapolated to higher order estimates with Richardson extrapolation. With each iteration, the order of accuracy increases by 2. Since the modified midpoint method is second order accurate, the extrapolation relationship is

$$A_{rs} = \frac{4^{s-1}A_{r,s-1} - A_{r-1,s-1}}{4^{s-1} - 1}, \qquad 2 \leqslant s \leqslant r \tag{A.7}$$

where the A_{rs} values with $s = 1$ correspond to modified midpoint estimates, i.e. $A_{r1} = \boldsymbol{r}^f (n = r)$. This extrapolation also provides an error estimate, except when $n = 1$ (no extrapolation is possible yet) and the error is instead estimated by comparing the two terms in equation (A.6):

$$e_i = \begin{cases} \left\| (\boldsymbol{r}^m)_i - \left[(\boldsymbol{r}^{m-1})_i + h\,\boldsymbol{g}(\boldsymbol{r}^m)_i \right] \right\|, & r = 1 \\ \left\| \dfrac{(A_{r,s-1})_i - (A_{r-1,s-1})_i}{4^{s-1} - 1} \right\|, & r > 1. \end{cases} \tag{A.8}$$

The BS method iterates until the solution converges or the pre-specified maximum number of iterations is reached. For further information on this method see [24].

A.3. Time step and iteration control

All of the methods used in this work allow for the time-step size Δt to vary, and all except RKF allow for iteration. We must therefore decide how to control time-step adjustment and iteration. Here we opt for relatively simple approaches. Despite this simplicity, *we have confirmed that the simulation and benchmark results reported are robust and insensitive to the details of time-step adjustment and iteration control*.

For the iterative explicit solvers (Heun and BS methods), we choose to increase Δt when the number of iterations needed for convergence falls below some minimum, $n_{\text{iter}}^{\text{min}}$. For the RKF integrator, the time step is increased if the maximum nodal error at convergence is below some fraction, f_{inc}, of the specified absolute integration tolerance, r_{tol}. We also prevent Δt from being increased for all solvers if it was just reduced in the current time step in order to achieve convergence. To control time-step reduction, with RKF we simply reduce Δt if the solution fails to converge at the given step size, and with the iterative explicit methods we cut the time step if the number of iterations reaches some threshold, $n_{\text{iter}}^{\text{cap}}$, the maximum number of iterations allowed. To impose the time-step changes, we employ constant adjustment factors F_{inc} and F_{dec} for increasing and decreasing the time step, respectively. In all simulations with explicit solvers we set these parameters as follows: $n_{\text{iter}}^{\text{min}} = 2$, $f_{\text{inc}} = 0.1$, $n_{\text{iter}}^{\text{cap}} = 3$, $F_{\text{inc}} = 1.2$ and $F_{\text{dec}} = 0.5$.

With the implicit trapezoid solver, we need to control iteration of the Newton–Raphson method and the frequency of the Jacobian calculation. We apply the same iteration control scheme as for the explicit iterative solvers but with $n_{\text{iter}}^{\text{cap}} = 15$. When employing an implicit solver, it is customary to reuse the same Jacobian for many time steps because the cost

associated with a slower convergence rate due to an outdated Jacobian is usually offset by the computational savings from not having to recalculate it [22]. Thus, when the solution does not converge at a given step size we have two options: update the Jacobian or cut the time step. For the simple test cases presented here we have found the following approach efficient. First, we check to see how old the Jacobian is. If it was recalculated during the last time step, we proceed directly to cutting the time step. If it is older than this, we first recalculate the Jacobian and reattempt the solution at the same step size. If convergence fails again, the step size is cut. Furthermore, we only allow the Jacobian to be reused for a maximum of five consecutive time steps. In addition to limiting the total number of iterations, we monitor whether the solution residual is decreasing as the solver iterates. If the residual increases for three consecutive iterations, the iteration is aborted and handled as described above. Lastly we point out that when new nodes are added or topological changes made, the Jacobian needs to be updated to reflect these changes.

Appendix B. Jacobian evaluation

The total force on a node i has contributions from elastic interactions, core energy (line tension), and applied stress, i.e.

$$\boldsymbol{f}_i^{\text{tot}} = \boldsymbol{f}_i^{\text{el}} + \boldsymbol{f}_i^{\text{core}} + \boldsymbol{f}_i^{\text{app}}. \tag{B.1}$$

Given the mobility law equation (14), the mobility function for node i is,

$$\boldsymbol{g}_i(\boldsymbol{r}) = \frac{\boldsymbol{f}_i^{\text{tot}}}{B \sum_j L_{ij}/2} \tag{B.2}$$

where L_{ij} is the length of segment i–j and the sum is over all nodes j connected to node i. Hence the Jacobian of the mobility function is

$$J_{mn}^{\text{mob}}(\boldsymbol{r}) = \frac{2}{B}\left[f_m^{\text{tot}} \frac{\partial \left(\sum_j L_{mj}\right)^{-1}}{\partial r_n} + \frac{1}{\sum_j L_{mj}} \frac{\partial f_m^{\text{tot}}}{\partial r_n} \right]. \tag{B.3}$$

The first term in the square brackets defines how segment length changes affect the nodal velocities and thus applies only locally—its contribution to the Jacobian is nonzero only if nodes m and n are nearest line neighbors. The second term defines the impact of force changes on the nodal velocities. When elastic interactions are considered every element of the Jacobian could in general be nonzero since dislocation stress fields are long-ranged. If the drag coefficient is taken to be a function of the dislocation character, as is often the case in body-centered-cubic metals [12, 14], additional terms would arise to account for those effects.

For convenience we will further decompose the Jacobian $\boldsymbol{J}^{\text{mob}}$. We begin by defining the *partial Jacobians* for the length and force contributions, respectively

$$\boldsymbol{J}^{\text{mob}} = \boldsymbol{J}^{\text{L}} + \boldsymbol{J}^{\text{f}} \tag{B.4}$$

and separate the force contributions from elastic interactions, core energy and applied stress,

$$\boldsymbol{J}^{\text{f}} = \boldsymbol{J}^{\text{f,el}} + \boldsymbol{J}^{\text{f,core}} + \boldsymbol{J}^{\text{f,app}}. \tag{B.5}$$

Specifically,

$$J_{mn}^{\text{L}}(\boldsymbol{r}) = \frac{2}{B} f_m^{\text{tot}} \frac{\partial \left(\sum_j L_{mj}\right)^{-1}}{\partial r_n} \tag{B.6}$$

$$J_{mn}^{\text{f,co}}(\boldsymbol{r}) = \frac{2}{B} \frac{1}{\sum_j L_{mj}} \frac{\partial f_m^{\text{co}}}{\partial r_n}, \tag{B.7}$$

where the superscript co = el, core, or app refers to each force contribution, respectively.

We now handle these partial Jacobians individually by considering the contributions from each individual segment and assembling their sub-Jacobians ($\boldsymbol{j}$) into the global Jacobian ($\boldsymbol{J}$). In the following, we define a local coordinate system for each segment such that it is confined to the x–y plane and the Burgers vector is $\boldsymbol{b} = b\,\boldsymbol{e}_x$. This corresponds to pure glide motion. Each sub-Jacobian can then be calculated in this coordinate system and then transformed into the global coordinates before being added to the global Jacobian. Our ordering convention in each sub-Jacobian is $\{x_1, y_1, x_2, y_2\}$ (z coordinates are omitted because their contributions are zero due to the glide constraint), where the subscripts 1 and 2 denote the two nodes of the segment (numbered arbitrarily).

The length change partial Jacobian for a segment of length L with one neighbor segment at each node is

$$\boldsymbol{j}^{\mathrm{L}} = \begin{bmatrix} -\widehat{\boldsymbol{j}}_1^{\mathrm{L}} & \widehat{\boldsymbol{j}}_1^{\mathrm{L}} \\ \widehat{\boldsymbol{j}}_2^{\mathrm{L}} & -\widehat{\boldsymbol{j}}_2^{\mathrm{L}} \end{bmatrix} \tag{B.8}$$

where

$$\widehat{\boldsymbol{j}}_i^{\mathrm{L}} = \frac{2(-1)^i}{BL(\sum_j L_{ij})^2}\boldsymbol{f}_i^{\mathrm{tot}}\,(\boldsymbol{r}_2 - \boldsymbol{r}_1)^{\mathrm{T}} = \frac{2(-1)^i}{BL(\sum_j L_{ij})^2}\begin{bmatrix} xf_{ix}^{\mathrm{tot}} & yf_{ix}^{\mathrm{tot}} \\ xf_{iy}^{\mathrm{tot}} & yf_{iy}^{\mathrm{tot}} \end{bmatrix} \tag{B.9}$$

and $x \equiv x_2 - x_1$, $y \equiv y_2 - y_1$.

Expressions for $\boldsymbol{J}^{\mathrm{f,core}}$ and $\boldsymbol{J}^{\mathrm{f,app}}$ have been found analytically, while the values of $\boldsymbol{J}^{\mathrm{f,el}}$ are here computed numerically by finite difference. Analytic expressions for $\boldsymbol{J}^{\mathrm{f,el}}$ between two straight segments have been derived as well by others [40]. The core energy contribution, for the DeWit and Koehler line energy model [41] is

$$\boldsymbol{j}^{\mathrm{f,core}} = \begin{bmatrix} \widehat{\boldsymbol{j}}_1^{\mathrm{f,core}} & -\widehat{\boldsymbol{j}}_1^{\mathrm{f,core}} \\ -\widehat{\boldsymbol{j}}_2^{\mathrm{f,core}} & \widehat{\boldsymbol{j}}_2^{\mathrm{f,core}} \end{bmatrix} \tag{B.10}$$

and the applied stress contribution is

$$\boldsymbol{j}^{\mathrm{f,app}} = \begin{bmatrix} 0 & \widehat{\boldsymbol{j}}_1^{\mathrm{f,app}} \\ \widehat{\boldsymbol{j}}_2^{\mathrm{f,app}} & 0 \end{bmatrix} \tag{B.11}$$

where

$$\widehat{\boldsymbol{j}}_i^{\mathrm{f,core}} = \frac{2E_{\mathrm{c}}b^2((1+\nu)x^2 + (1-2\nu)y^2)}{BL^5(1-\nu)\sum_j L_{ij}}\begin{bmatrix} -y^2 & xy \\ xy & -x^2 \end{bmatrix} \tag{B.12}$$

$$\widehat{\boldsymbol{j}}_i^{\mathrm{f,app}} = \frac{(-1)^i\, s}{B\sum_j L_{ij}}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{B.13}$$

and

$$s = \left[\frac{(\boldsymbol{r}_2 - \boldsymbol{r}_1)}{L} \times \boldsymbol{f}_{\mathrm{g}}^{\mathrm{app}}\right] \cdot \boldsymbol{e}_z \tag{B.14}$$

$$\boldsymbol{f}_{\mathrm{g}}^{\mathrm{app}} = \boldsymbol{f}^{\mathrm{app}} - \left(\boldsymbol{f}^{\mathrm{app}} \cdot \boldsymbol{e}_z\right)\boldsymbol{e}_z. \tag{B.15}$$

$\boldsymbol{e}_z$ is the unit vector along the z-axis (and is present because of our local coordinate system choice) and $\boldsymbol{f}^{\mathrm{app}}$ is the Peach–Koehler force on the segment due to the applied stress.

When the Burgers vector is instead $\boldsymbol{b} = b\,\boldsymbol{e}_z$, as in the case of a prismatic loop collapsing by climb, the core energy contribution to the Jacobian becomes

$$\widehat{\boldsymbol{j}}_i^{\mathrm{f,core}} = \frac{2E_{\mathrm{c}}b^2}{BL^3(1-\nu)\sum_j L_{ij}}\begin{bmatrix} -y^2 & xy \\ xy & -x^2 \end{bmatrix}. \tag{B.16}$$

$\widehat{\boldsymbol{j}}_i^{\mathrm{L}}$ is not affected by this Burgers vector change and can be computed as given above.

For the elastic contribution partial Jacobian we employ a centered finite difference scheme that is second order accurate,

$$\frac{\partial f_m^{\mathrm{el}}}{\partial x_n} = \frac{f_m^{\mathrm{el}}(\boldsymbol{r} + \delta\, \boldsymbol{e}_n) - f_m^{\mathrm{el}}(\boldsymbol{r} - \delta\, \boldsymbol{e}_n)}{2\,\delta}, \tag{B.17}$$

where δ is a perturbation parameter and $\boldsymbol{e}_n$ is the nth standard basis vector. δ values in the range $0.1 - 1b$ were used, which was selected based on solver performance. The value giving the highest performance was observed to scale with segment size. It should be noted that the implicit solver performance may be degraded by the numerical error from these finite difference approximations.

One computationally significant detail with Jacobian calculation is the cost of accounting for elastic interactions. If all segment–segment interactions are included in the Jacobian, the cost scales as $\mathcal{O}(N^2)$, in the same way as the cost of force calculations (but with a higher prefactor). However, we have found that it is sufficient to include only elastic interactions between segments that are within an *enhancement radius*, r_{enh}, of each other. By changing r_{enh} we can empirically determine over what range interactions are significant for an efficient implicit integrator in their contribution to the Jacobian, thus eliminating the costly inclusion of all elastic interactions.

Appendix C. Analytical solution for collapsing prismatic loop

To quantify the error of DD simulations, we have obtained the analytic expressions for the radius of a collapsing circular prismatic loop as a function of time. The radius of the loop as a function of time, $R(t)$, obeys the following ODE

$$B\frac{\mathrm{d}R}{\mathrm{d}t} = f^{\mathrm{core}}(R) + f^{\mathrm{el}}(R), \tag{C.1}$$

where f^{core} and f^{el} are core and elastic [30, 31] forces, respectively,

$$f^{\mathrm{core}}(R) = -\frac{E_{\mathrm{c}}}{R(1-\nu)} \tag{C.2}$$

$$f^{\mathrm{el}}(R) = -\frac{\mu b^2}{4\pi R(1-\nu)} \ln\left(\frac{8R}{r_{\mathrm{c}}}\right) \tag{C.3}$$

and r_{c} is the core radius in the non-singular [13] theory of dislocations. The solution $R(t)$ can be expressed by the following implicit relation,

$$0 = \left(\frac{r_{\mathrm{c}}}{b}\right)^2 \exp\left(-8\pi\alpha\right) \mathrm{Ei}\left[8\pi\alpha + 2\ln\left(\frac{8R}{r_{\mathrm{c}}}\right)\right] + \frac{16\mu\, t}{\pi B(1-\nu)} + C(R_0), \tag{C.4}$$

where $\mathrm{Ei}(\cdot)$ is the exponential integral and $\alpha = E_{\mathrm{c}}/(\mu\, b^2)$. The integration constant $C(R_0)$ can be evaluated by specifying the initial radius, R_0, at $t = 0$.

The solution for zero core energy is obtained simply by substituting $\alpha = 0$ into equation (C.4). The solution for the line-tension-only model (i.e. ignoring elastic interactions) can be obtained explicitly, (analogous to grain growth [42])

$$R(t) = \sqrt{R_0^2 - \frac{2E_{\mathrm{c}}\, t}{B(1-\nu)}}. \tag{C.5}$$

Analytic solutions for various combinations of elastic and core energies are plotted in figure 1(*b*).

Appendix D. Explicit solver analysis with a collapsing prismatic loop

In this appendix we derive an expression for the number of time steps needed for an explicit integrator to simulate the collapse of a prismatic loop from some initial radius R_0 to some final radius R_f in the stability-limited regime. The expression is obtained for the line-tension-only model. Making this prediction is nontrivial because the maximum stable time step is a function of both the loop radius and the element size. We start with equation (16) for the maximum stable time step for a loop discretized by uniform elements. Next, we assume that the total simulation time is the sum of the maximum stable time steps from the simulation

$$t_{\text{sim}} = \sum_{i=1}^{N_{\text{sol}}} \Delta t^s_{i,\max}, \tag{D.1}$$

where N_{sol} is the number of time steps taken with an explicit integrator and $\Delta t^s_{i,\max}$ is the maximum stable time step of the ith solution step. Now we assume time integration is perfect, which is most likely to be valid when discretization error is small and the solution is severely stability limited, both of which are true when the number of elements is large. We can then equate t_{sim} with the exact time t_{exact}, which can be obtained from the analytic solution, equation (C.5), and obtain

$$\sum_{i=0}^{N_{\text{sol}}-1} R_i = \frac{\left(R_0^2 - R_f^2\right)(2\pi R/l)^2}{20 c_{\text{int}}}. \tag{D.2}$$

Again, assuming no numerical error in every integration step, we can write the following recursive relation,

$$R_i^2 = R_{i-1}^2 - \frac{2E_c}{B(1-\nu)} \Delta t^s_{\max}(R_{i-1}). \tag{D.3}$$

Combining equations (D.1), (D.3) and (16), we have

$$\sum_{i=0}^{N_{\text{sol}}-1} R_i = R_0^2 \sum_{i=0}^{N_{\text{sol}}-1} \left(1 - \frac{20\, c_{\text{int}}}{(2\pi R/l)^2}\right)^i. \tag{D.4}$$

where the sum can be performed analytically to give a closed form expression. Equating equation (D.2) with equation (D.4), N_{sol} can be obtained as

$$N_{\text{sol}} = \frac{2\ln(R_f/R_0)}{\ln\left[1 - \frac{20\, c_{\text{int}}}{(2\pi R/l)^2}\right]}. \tag{D.5}$$

Expanding this expression about $2\pi R/l = 0$ gives the final expression, equation (17).

References

[1] Kubin L P, Canova G, Condat M, Devincre B, Pontikis V and Brechet Y 1992 Dislocation microstructures and plastic flow: a 3D simulation *Solid State Phenom.* **23–24** 455

[2] Kubin L P 1993 Dislocation paterning during multiple slip of fcc crystals: a simulation approach *Phys. Status Solidi* a **135** 433

[3] Verdier M, Fivel M and Groma I 1998 Mesoscopic scale simulation of dislocation dynamics in fcc metals: principles and applications *Modelling Simul. Mater. Sci. Eng.* **6** 755–70

[4] Schwarz K W 1999 Simulations of dislocations on the mesoscopic scale: I. Methods and examples *J. Appl. Phys.* **85** 108

[5] Devincre B, Kubin L P, Lemarchande C and Madec R 2001 Mesoscopic simulations of plastic deformation *Mater. Sci. Eng.* A **309–310** 211–9
[6] Politano O and Salazar J M 2001 A 3D mesoscopic approach for discrete dislocation dynamics *Mater. Sci. Eng.* A **309–310** 261–4
[7] Weygand D, Friedman L H, van der Giessen E and Needleman A 2001 Discrete dislocation modeling in three-dimensional confined volumes *Mater. Sci. Eng.* A **309–310** 420–4
[8] Ghoniem N M 2005 A perspective on dislocation dynamics (*Handbook of Materials Modeling*) ed S Yip (Dordrecht: Springer) pp 2871–7
[9] Arsenlis A, Cai W, Tang M, Rhee M, Oppelstrup T, Hommes G, Pierce T G and Bulatov V V 2007 Enabling strain hardening simulations with dislocation dynamics *Modelling Simul. Mater. Sci. Eng.* **15** 553
[10] Shin C S, Fivel M C, Verdier M and Kwon S C 2006 Numerical methods to improve the computational efficiency of discrete dislocation dynamics simulations *J. Comput. Phys.* **215** 417–29
[11] Wang Z, Ghoniem N, Swaminarayan S and LeSar R 2006 A parallel algorithm for 3D dislocation dynamics *J. Comput. Phys.* **219** 608–21
[12] Bulatov V V and Cai W 2006 *Computer Simulations of Dislocations* (Oxford: Oxford University Press) (http://micro.stanford.edu auxiliary material)
[13] Cai W, Arsenlis A, Weinberger C R and Bulatov V V 2006 A non-singular continuum theory of dislocations *J. Mech. Phys. Solids* **54** 561–87
[14] Cai W and Bulatov V V 2004 Mobility laws in dislocation dynamics simulations *Mater. Sci. Eng.* A **387–389** 277–81
[15] Bulatov V V *et al* 2006 Dislocation multijunctions and strain hardening *Nature* **440** 1174
[16] Cai W, Bulatov V V, Pierce T G, Hiratani M, Rhee M, Bartelt M and Tang M 2004 Massively-parallel dislocation dynamics simulations *Solid Mechanics and Its Applications* ed H Kitagawa and Y Shibutani vol 115 (Dordrecht: Kluwer) p 1
[17] Devincre B, Hoc T and Kubin L 2008 Dislocation mean free paths and strain hardening of crystals *Science* **320** 1745
[18] Ghoniem N M, Tong S-H and Sun L Z 2000 Parametric dislocation dynamics: a thermodynamics-based approach of mesoscopic plastic deformation *Phys. Rev.* B **61** 913
[19] Devincre B, Madec R, Monnet G, Queyreau S, Gatti R and Kubin L 2011 Modeling crystal plasticity with dislocation dynamics simulations: the 'microMegas' code (http://zig.onera.fr/mm_home_page)
[20] Huang J and Ghoniem N M 2002 Accuracy and convergence of parametric dislocation dynamics *Modelling Simul. Mater. Sci. Eng.* **10** 1–19
[21] Gear C W 1971 *Numerical Initial Value Problems in Ordinary Differential Equations* (Englewood Cliffs, NJ: Prentice-Hall)
[22] Shampine L F 1994 *Numerical Solution of Ordinary Differential Equations* (Boca Raton, FL: CRC Press)
[23] Chapra S C and Canale R P 2009 *Numerical Methods for Engineers* (New York: McGraw-Hill)
[24] Schilling R J and Harris S L 2000 *Applied Numerical Methods for Engineers* (Pacific Grove, CA: Brooks/Cole)
[25] Bradie B 2006 *A Friendly Introduction to Numerical Analysis* (Upper Saddle River, NJ: Pearson Prentice Hall)
[26] Moin P 2010 *Fundamentals of Engineering Numerical Analysis* (New York: Cambridge University Press)
[27] Shampine L F and Watts H A 1976 *Practical Solution of Ordinary Differential Equations by Runge–Kutta Methods* SAND76-0585 (Albuquerque, NM: Sandia National Laboratories)
[28] The DDLab program http://micro.stanford.edu/~caiwei/Forum/2005-12-05-DDLab/
[29] The ParaDiS program http://paradis.stanford.edu
[30] Kroupa F 1960 Circular edge dislocation loop *Czech. J. Phys.* B **10** 284–93
[31] Hirth J P and Lothe J 1982 *Theory of Dislocations* (New York: Wiley)
[32] Hull D and Bacon D J 2001 *Introduction to Dislocations* (Oxford: Butterworth-Heinemann)

[33] Belytschko T and Mullen R 1976 Mesh partitions of explicit–implicit time integrators *Formulations and Computational Algorithms in Finite Element Analysis* ed K Bathe *et al* (Cambridge, MA: MIT Press)
[34] Belytschko T, Jen H-J and Mullen R 1979 Mixed methods for time integration *Comput. Methods Appl. Mech. Eng.* **17/18** 259–75
[35] Neal M O and Belytschko T 1989 Explicit–explicit subcycling with non-integer time step ratios for structural dynamic systems *Comput. Struct.* **31** 871–80
[36] Weygand D, Friedman L H, 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
[37] Kubin L P 2013 *Dislocations, Mesoscale Simulations and Plastic Flow* (Oxford: Oxford University Press)
[38] Geiser J 2009 *Decomposition Methods for Differential Equations* (Boca Raton, FL: CRC Press)
[39] Gravouil A and Combescure A 2001 Multi-time-step explicit–implicit method for non-linear structural dynamics *J. Numer. Methods Eng.* **50** 199–225
[40] Arsenlis A 2013 private communication
[41] DeWit G and Koehler J S 1959 Interaction of dislocations with an applied stress in anisotropic crystals *Phys. Rev.* **116** 1113–20
[42] Callister W D and Rethwisch D G 2008 *Fundamentals of Materials Science and Engineering* (New York: Wiley)