Chapter 13: Mesoscale Dynamics: Dislocation Dynamics examples

I. Introduction

In Chapter 13 we discuss how we can define dynamical simulations at the mesoscale, starting with the identification of the *entities* that will be modeled. In these exercises, we focus on dislocations as the fundamental entities of the simulation. While fully three-dimensional simulations are now the norm, much interesting work has been done using simple two-dimensional models, as discussed in the textbook. Here we show how the implementation of a simple 2D dislocation dynamics simulation is a simple extension of the molecular dynamics codes developed in Chapter 6. We will start with a very straightforward approach, which we will see is not correct owing to the choice of boundary conditions. We will then show how that code can be corrected. We finally introduce the beginnings of what it takes to do three-dimensional dislocation modeling by an approximate calculation of a Frank-Read source.

We note that Professor Wei Cai at Stanford has a much more elaborate set of MATLAB codes to do 3D dislocation dynamics at his web site
http://micro.stanford.edu/~caiwei/Forum/2005-12-05-DDLab/
We encourage interested students to take advantage of his resources.

II. A model system

We will model a very simple system - a series of parallel, straight edge dislocations all with line directions along the $\pm\hat{z}$ direction and Burgers vector along the $\pm\hat{x}$ direction.[1] In the *xy* plane, these dislocations appear as points. We place the dislocations are in a square simulation cell in the *xy* plane with periodic boundary conditions. The motion of the dislocations is restricted to the glide plane, i.e., in the $\pm\hat{x}$ direction. The force per length of dislocation in the $\pm\hat{x}$ direction on dislocation *i* from dislocation *j* is given by the Peach-Koehler force,[2]

$$F_x(i) = bb_i\sigma_{xy}(j) \text{ ,} \tag{1}$$

where b is the magnitude of the Burgers vector, b_i is the sign of the Burgers vector (± 1), and $\sigma_{xy}(j)$ is the stress from dislocation *j* evaluated at dislocation *i*, which is

$$\sigma_{xy}(j) = \frac{Gbb_j}{2\pi(1-\nu)} \frac{x_{ji}\left(x_{ji}^2 - y_{ji}^2\right)}{\left(x_{ji}^2 + y_{ji}^2\right)^2} \text{ .} \tag{2}$$

In Eq.(2), $x_{ji} = x_i - x_j$ and $y_{ji} = y_i - y_j$. Since we are interested in generic behavior, we will introduce the unit of stress $\sigma_o = Gb/2\pi(1-\nu)$ and write all stresses in scaled units, i.e., as

$$\sigma_{xy}(j) = b_j \frac{x_{ji}\left(x_{ji}^2 - y_{ji}^2\right)}{\left(x_{ji}^2 + y_{ji}^2\right)^2} \tag{3}$$

The force per length acting on dislocation *i* is

$$F_x(i) = \frac{Gb^2 b_j b_i}{2\pi(1-\nu)} \frac{x_{ji}\left(x_{ji}^2 - y_{ji}^2\right)}{\left(x_{ji}^2 + y_{ji}^2\right)^2} \quad , \tag{4}$$

which in scaled units is

$$F_x(i) = b_j b_i \frac{x_{ji}\left(x_{ji}^2 - y_{ji}^2\right)}{\left(x_{ji}^2 + y_{ji}^2\right)^2} \ . \tag{5}$$

The net force on dislocation *i* from the rest of the simulations (including the periodic lattice) is

$$\frac{F_x^T(i)}{L} = b_i \sum_{\vec{R}} \sum_{j \neq i=1}^{N} b_j \frac{x_{jiR}\left(x_{jiR}^2 - y_{jiR}^2\right)}{\left(x_{jiR}^2 + y_{jiR}^2\right)^2} \ , \tag{6}$$

where $x_{jiR} = R_x + x_i - x_j$ and $y_{jiR} = R_y + y_i - y_j$, where $\vec{R} = \left(R_x, R_y\right)$ is a lattice vector.

Our dislocation dynamics code is a variant of the molecular dynamics code introduced in Chapter 6. A major difference is the solution of the equations of motion. As discussed in Chapter 13 of the textbook, dislocation motion does not conserve energy; there are dissipative forces on dislocations that arise from the generation of phonons, etc. The equation of motion becomes

$$m\frac{d^2 x_i}{dt^2} = F_x(i) - Bv_i \ , \tag{7}$$

where *B* is the friction coefficient and v_i is the velocity of the dislocation in the $\pm\hat{x}$ direction. If the damping is large (high friction), then the terminal velocity is reached in a time that is small compared to the time step of the simulation. This case, called the overdamped limit, allows us to ignore the initial inertial movement and use the relation

$$v_i = \frac{F_x(i)}{B} = MF_x(i) \ , \tag{8}$$

where M=1/B is the mobility. In the following code, we ignore M, incorporating it into the time scale. A simple solution to Eq.(7) is the Euler equation, which yields

$$x_i(t + \Delta t) = x_i(t) + v_i(t)\Delta t \quad . \tag{9}$$

where Δt is the time step. We note that other, more sophisticated, solutions are available.[3]

One complication when using Eq.(6) is that when dislocations are close to each other, the forces can be very high, which can in turn lead to unphysically large changes in position when using a fixed time step. We could avoid that problem by using a small time step, but that would be inefficient for times when the forces are small. Thus, in the code below, we calculate the maximum stress at each time step ($F_{\max} = \max\left(\left|F_x(i)\right|\right)$) and set the time step by

$$\Delta t = \Delta x_{\max} / F_{\max} \quad , \tag{10}$$

where $\Delta x_{\max}$ is a parameter that sets the maximum distance a dislocation can move in a time step. Note that this solution is not ideal. At the beginning of the calculation, we may want to move the dislocations by a relative large amount to move quickly towards a stable structure ($\Delta x_{\max}$ relatively large). However, near a potential minimum (zero force) the dislocations could have relatively small motions, so we expect that we would want to have $\Delta x_{\max}$ relatively small. The code could be modified to have as input the final positions of a previous run (replacing the `initDD` call).[1] Then a series of calculations could be done, introducing a smaller value of $\Delta x_{\max}$ as the system nears a stable minimum.

II. a. 2D Dislocation Dynamics Simulation: Implementation of a simple model

Examining Eq.(5) we see that the force is long ranged (~1/r at large separations), which requires us to use a special technique to evaluate the lattice sums, as discussed below. For now, however, we will ignore any issues with the long-ranged nature of the force and just use the standard lattice sums that we introduced in Chapter 3.

The first step is to create the initial dislocation structure. In a routine called `initDD`, we place dislocations at random in the simulation cell with even numbers dislocations with positive and negative Burgers vectors. In the code, `n` is the number of dislocations and the positions in the square cell span from `0` to `D`.

```
function[x,y,b] = initDD(n,D)
%  we use the rand function, which creates a 1D array of random
%  numbers between 0 and 1
  x = rand(n,1)*D;
  y = rand(n,1)*D;
  b = zeros(n,1);
% assign b with equal numbers of +1 and -1
  for i=1:n/2
    b(i) = 1;
  end
  for i=n/2+1:n
    b(i) = -1;
  end
```

[1] Note that since the *y* positions and the Burgers vectors do not change in a simulation, only the *x* positions need to be modified from run to run.

In the main code, in the file `DD2D.m`, we call `sumDD`, in which we sum the forces on each dislocation. This routine is a modified version of the code for the Lennard-Jones atom in Chapter 6. We evaluate the sum in Eq.(6) using the minimum image convention and a cutoff set to have the box size, i..e, $r_c = D/2$. Note that the time step is set by the maximum force.

The code:

```
%  input:  ndis = the number of dislocations
%          nsteps = number of time steps
%          dxmax = maximum move per time step
%
%  output: xi=initial x positions,x = final positions,y,b
%          fx = final forces on each dislocation
%          xdm = the maximum distance a dislocation has moved in the
%          calculation

function[xi,x,y,b,vex,xdm] = DD2D(ndis,nsteps,dxmax)
% set size of system (arbitrary)
D = 1000;
% set cutoff to be 1/2 the box length
rc = D/2;
% initial positions
[x,y,b] = initDD(ndis,D);
% store initial position
xi = x;

% start the time steps
for j=1:nsteps
[fx,fmax] = sumDD(ndis,D,rc,x,y,b);
  dt = dxmax/fmax;
  for i=1:ndis
    x(i) = x(i) + fx(i)*dt;
    if x(i) > D
        x(i) = x(i) - D;
    end
    if x(i) < 0
        x(i) = x(i) + D;
    end
  end
end
xd = x - xi;               % change in position over the run
xd = xd - D*round(xd/D);  % remove movement across periodic boundaries
xdm=max(abs(xd));          % find maximum movement
```

The total force comes from the function in `sumDD.m`. It is based on the use of the minimum-image convention plus a cutoff (`rc`) set to the box length (`D`) divided by 2.

```
function[fx,fmax]= sumDD(n,D,rc,x,y,b)
%  set force component to 0
fx=zeros(n,1);

for i = 1:n-1  % note limits
    for j=i+1:n  %  note limits
% mimimum image convention
        dx = x(i) - x(j);
        dy = y(i) - y(j);
        dx = dx - D*round(dx/D);
        dy = dy - D*round(dy/D);
        dsq = dx^2 + dy^2;
        dist = sqrt(dsq);
        if dist <= rc
            ffx = b(i)*b(j)*dx*(dx^2-dy^2)/dsq^2;
            fx(i) = fx(i) + ffx;
%  add -f to sum of force on j
            fx(j) = fx(j) - ffx;
        end
    end
end
% calculate maximum value of force
fmax=max(abs(fx));
```

The output of the simulation includes the initial (`xi`) and final (`x,y,b`) configurations. There are many ways to plot these. One simple way is to use the MATLAB function `scatter` with filled symbols and use the value of the Burgers vector to set the color, e.g.,

```
scatter(x,y,50,b,'d','fill');axis
square
```

This yields output like that shown in the figure (the initial configuration for 500 dislocations in a simulation cell of size *a*=1000).

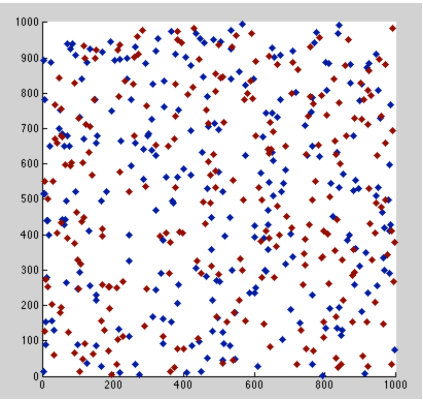

Figure 1: Initial random configuration of a set of edge dislocations with Burgers vectors along the x axis. Colors indicate the sign of the dislocation.

***Exercise**:*

1. Using the code provided, run simulations until the microstructure has approximately converged. The final forces (`fx`) should be for the most part small,

though some dislocations may have larger forces acting on them, which generally arise from strong local interactions in dislocation dipoles, which are two dislocations with opposite sign that are close to each other on different slip planes. The forces on the dislocations in the dipoles are large and opposite in sign, and will show up as spikes in a plot of the final forces. Since the forces are large, and we do not directly include damping when using a linear force-velocity dynamics, we will generally see oscillations in the positions of closely bound dipoles.

In the simulation shown in the figure, $N = 500$ dislocations were included. Vary the size of the Δx_{max} parameter. Does its choice affect the rate of convergence?

When you are satisfied that the simulation is at least relatively close to its converged state, plot the final structure and comment. Compare to results shown in Reference [1].

II. b 2D Dislocation Dynamics Simulation: Implementation with full periodicity

In the simulations from Problem I, you should have seen a very curious microstructure; the dislocations tend to form vertical lines made up of like-signed dislocations. Note the pattern. You should have seen 4 lines of dislocations, alternating in sign. Note that the lines with the same sign are separated by half the box length, i.e., at the cutoff distance. These are not realistic microstructures. As we shall see, that structure is an artifact of the use of a cutoff in a system with long-ranged interactions. This point is discussed in [1].

While there are a number of ways to incorporate the effects of the long-ranged interactions, we use an approach introduced in [1]. Suppose we are interested in the force from dislocation *a* acting on dislocation *b*, as described in Figure 2. What we actually need, based on Eq.(6), is the force from *a* and *all its replicas* throughout space. Consider first just the replicas of *a* in the *y* direction, which corresponds to a repeating set of dislocations separated by *D*, the size of the simulation cell. This type of structure is just a tilt grain boundary. Hirth and Lothe [2] show that the stress from that line of dislocation is

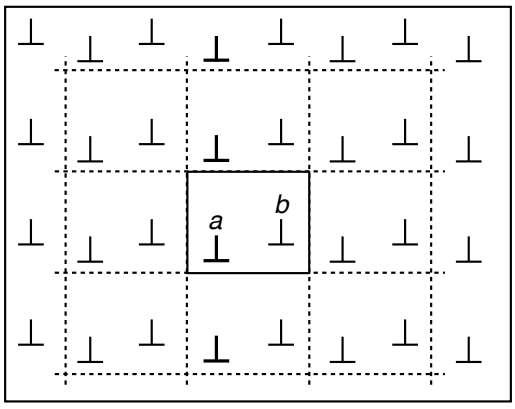

Figure 2: 2D periodic array with 2 dislocations per the simulation cell. The stress field from dislocation *a* and its images can be taken as a sum of the stress fields of parallel lines of periodic dislocations aligned along the *y-axis.*

$$\sigma_{xy}(j) = \frac{Gb_j}{2\pi D(1-\nu)} \frac{t(\cosh t \cos u - 1)}{(\cosh t - \cos u)^2} \quad , \qquad (11)$$

where $t = 2\pi x_{ji}/D$ and $u = 2\pi y_{ji}/D$. An impor-

tant feature of the expression in Eq.(11) is that at large x, (remembering that $\cosh(t) = \left(e^{-t} + e^{t}\right)/2$)

$$\sigma_{xy}(j) \sim \frac{t \cosh t \cos u}{\cosh^2(t)} \sim t e^{-t} \cos u \ . \tag{12}$$

Thus, the stress from a line of dislocation falls off exponentially in the direction perpendicular to the line. (Note that this argument still works if $x_{ji} < 0$.) Thus, we can find the total stress from j and its replicas by summing over the vertical lines (see Figure 2),

$$\sigma_{xy}(j) = \frac{Gb_j}{2\pi D(1-\nu)} \sum_{n=-m}^{m} \frac{t_n\left(\cosh t_n \cos u - 1\right)}{\left(\cosh t_n + \cos u\right)^2} \ , \tag{13}$$

where $t_n = 2\pi\left(x_{ji}/D + n\right)$. Because of the short range nature of the stress from the line seen in Eq.(10), we can use a relatively small value for m, i.e., we do not need to include many lines in the $\pm\hat{x}$ direction.

Implementation of Eq.(13) in the 2D dislocation dynamics code is straightforward. One simply replaces the sum based on Eq.(6) (i.e., that based on the minimum image convention) by a direct sum over the dislocations, where each stress call involves the sum over a series of vertical lines in Eq.(13). We note that this approach is appreciably slower than using the minimum image convention in Section II. However, this approach includes all interactions in a proper way.

There are few changes in the main code (in `DD2Dp.m`). We added an input parameter that sets the value for m in Eq.(11) and changed the call for the routine that calls the summation of the force (in `sumDDp.m`). The force calculation dispenses with the minimum image convention and replaces it with a sum over the stresses given in Eq.(11):

```
function[fx,fmax]= sumDDp(n,a,m,x,y,b)
%  set force component to 0
fx=zeros(n,1);
for i = 1:n-1  % note limits
    for j=i+1:n  %  note limits
% stress relative to the dislocation lines
          dx = -x(j) + x(i);
          dy = -y(j) + y(i);
          [stress] = stressp(dx,dy,a,m);
          pkf = b(i)*b(j)*stress;
          fx(i) = fx(i) + pkf;
          fx(j) = fx(j) - pkf;
    end
end
% calculate maximum value of force
fmax=max(abs(vx));
```

In `stress.m` we calculate the total stresses of Eq.(11):

```
function[stress]= stressp(dx,dy,a,m)
u = 2*pi*dy/a;
t = 2*pi*dx/a;
stress = 0;
for n = -m:m
  tn = t - 2*pi*n;
  f = tn*(cosh(tn)*cos(u)-1)/(cosh(tn)-cos(u))^2;
  stress = stress + f/a;
end
```

The parameters are as defined in Eq.(11).

II.c Interaction with an external stress field

The interaction of a dislocation (*i*) with an external stress field τ is just $b_i \tau$, so plus-signed dislocations are pushed in one direction and like-signed dislocations the other. The code can be modified to include an external stress, for example by changing it to something like:

```
for j=1:nsteps
  [fx,fmax] = sumDDp(ndis,a,mm,x,y,b);
  dt = dxmax/fmax;
  for i=1:ndis
    f = fx(i) + sig*b(i);
    x(i) = x(i) + f*dt;
    if x(i) > a
        x(i) = x(i) - a;
    end
    if x(i) < 0
        x(i) = x(i) + a;
    end
  end
end
```

The plastic strain under stress can be calculated by first relaxing the dislocation positions and storing those positions (e.g., as $\{x_o\}$). After a stress is applied, the dislocations will move along their slip planes, from which one can calculate their change in position $\Delta x_i = x_i - x_{io}$. Based on the standard analysis of strain created by the movement of dislocations (see, for example, the book by Hull and Bacon [5]), the plastic strain is

$$\gamma_p = \frac{1}{D^2} \sum_{i=1}^{N} b_i \Delta x_i \,. \tag{14}$$

Note that periodic boundary conditions must be taken into account when calculating Δx_i .

II. d Scaling

We have introduced this model with reduced units, removing all the materials parameters. The only length scale left in the problem is the mean inter-dislocation spacing, , where the dislocation density is $\rho = N / D^2$, and N is the number of dislocations. Note that $\rho^{-1/2} = D / \sqrt{N}$.

Exercises:

Note: These calculations increase in complexity .

2. Using a code based on Eq.(11)-(13), place an even number of dislocations randomly into a simulation cell. Choose an initial value for $\Delta x_{\max}$. Run simulations until the microstructure has approximately converged. It might be easiest to modify the code so that you an restart from an earlier run. It is easy to do by adding the x, y, and b variables as input and output. Thus you can take the output from one run can serve as the input for the next. (See footnote on page 3)

   For a converged structure, the final forces (`fx`) should be for the most part small, though some dislocations may have larger forces acting on them. These large forces generally arise from strong local interactions between dislocation dipoles - pairs of dislocations with the opposite sign. In practice, we will see oscillations in the positions of closely bound dipoles. Since the dipoles are two dislocations with opposite sign that are close to each other, in real materials they might annihilate, which is not taken into account in this code. When you are satisfied that the simulation is at least relatively close to its converged state, plot the final structure and compare to results from Problem 1 and those shown in Reference [1]. Vary $\Delta x_{\max}$ to examine how its choice affects the final results.

3. In reference [1] the authors introduced a correlation function to describe the local order in the dislocations. For a dislocation at the origin, it gives the probability that there is another dislocation at a position (*x,y*) relative to the origin. It is thus similar to the radial distribution function discussed in the text for atomistic simulations and whose implementation is described in the problems for Chapter 6. The correlation function introduced for dislocations is somewhat different, however, because the interactions between dislocations of the same sign differ from those between dislocations of opposite sign. Coupled with the anisotropy of the dislocation stress fields, the local ordering for like-signed dislocations is very different than that for dislocations with different signs.

Based on the calculation for g(r) given in the exercises for Chapter 6, calculate the weighted distribution function of [1] by calculating $g^{++}(x,y)$ based on dislocations with the same sign and $g^{+-}(x,y)$ for dislocations with opposite signs. The net correlation function can be defined as $g(x,y) = g^{++}(x,y) - g^{+-}(x,y)$. Compare $g(x,y)$ for calculations with different dislocation densities and the compare by scaling the *x* and *x* axes by the mean interdislocation spacing, $\rho^{-1/2}$. How do they compare?

1. Modify the code to include an external stress (see Section II.c). Vary the external stress and examine how the dislocation structures change. Compare the distribution functions as calculated in Problem 3 for systems under varying stress. Determine the stress-strain behavior and comment.

2. A simple model for a tilt grain boundary can be created by creating a fixed line of edge dislocations in the y direction. For example, suppose for a cell of size D that we placed a series of dislocations with Burgers vector b_g at positions $(x_0, i\,a)$, where i is an integer that spans from 1 to D/a. As long as D/a is an integer, then (with the periodic boundary conditions), this would create an infinite line of dislocations of spacing a. The stress from this line of dislocations is given by Eq.(11), with D replaced by a. The angle of the boundary is given by $2\sin(\theta/2) = b_g/a$.[3] For a boundary with $a \ll D$, from Eq.(12) we see that the stress would be very short-ranged (so no need to sum over the lines of the replica grain boundaries). To avoid issues with lattice rotations, it may make the most sense to place one grain boundary at $x_o = D/4$ with $b = b_g$ and another at $x_o = 3D/4$ with $b = -b_g$. Implement this simple model for a grain boundary and model the deformation structure development as a function of applied stress. Determine the stress-strain behavior and compare with Problem 4.

III. Simulation of Frank-Read source

Dislocation-based plasticity is decidedly three-dimensional and the simple model of "point dislocations" described in Section II is inadequate. In this section we will just scratch the surface on how one can extend those simple models to fully-three-dimensional dislocation dynamics. We introduce the basic concepts by creating a simu-

lation of a Frank-Read source. We emphasize again that what is presented here is just the beginnings of a much more complex simulation method. As noted above, Professor Wei Cai at Stanford has a much more elaborate set of MATLAB codes to do 3D dislocation dynamics based on [3] at his web site[2].

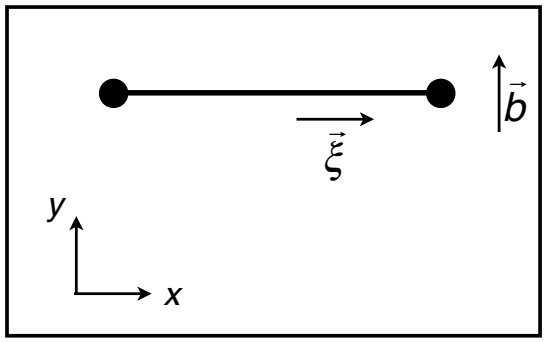

Figure 3: Schematic of the Frank-Read source reflecting the coordinate system.

For this example, we will consider a Frank-Read source that consists of a dislocation line pinned at each end. These pinning points could be sessile junctions or points at which the dislocation leaves the glide plane. We will not worry about the details, but will just assume that the ends of the dislocation segment are fixed. The glide plane of the Frank-Read source will be taken as the *xy* plane and the initial dislocation line is aligned along the *x* axis, as shown in Figure 3. As a stress is applied to the Frank-Read source, the dislocation bows out and eventually wraps back around the fixed sites, as shown in Figure 4. Note that the Burgers vector in Figure 4 is the same for the entire dislocation. The line direction follows the dislocation line from the fixed point on the left to that on the right. Thus, the line direction of the two segments below the line between the fixed points are in opposite directions. Those segments would annihilate, leaving an expanding loop and a new Frank-Read source. Our goal in this section is to model the operation of the source, at least to the configuration shown in Figure 4.

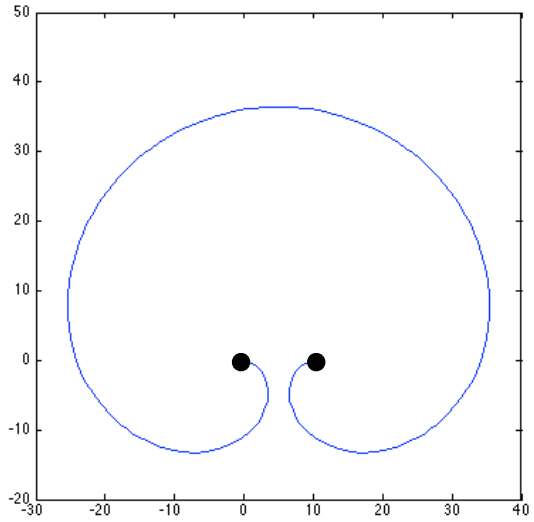

Figure 4. Frank-Read source.

The model we will use is very simple. We will ignore the elastic interactions between different segments of the dislocation and will only consider the interaction of the dislocation with an external stress field. We will approximate the effects of the line energy o the dislocation by assuming a simple line-tension model, which you may have seen in introductory classes on the mechanical behavior of materials.

IIIa. Effects of an external stress

Suppose we apply an external stress σ to this system. The general stress tensor is

[2] http://micro.stanford.edu/~caiwei/Forum/2005-12-05-DDLab/

$$\sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix}. \tag{15}$$

For our sample problem we take the Burgers vector to be along the *y* axis, $\vec{b} = b(0,1,0)$, as indicated Figure 3. The initial line direction is $\hat{\xi} = (1,0,0)$. Since the Burgers vector is perpendicular to the line direction, the dislocation segment is (initially) an edge dislocation.

As the dislocation responds to an external stress, it will begin to move. Since the ends are fixed, it will bow out and thus segments of the dislocation will take on a general orientation in the *xy* plane, i.e., the unit vector along the line direction will be of the form

$$\hat{\xi} = (\xi_x, \xi_y, 0), \tag{16}$$

where $\hat{\xi} \cdot \hat{\xi} = \xi_x^2 + \xi_y^2 = 1$.

The force per length on a dislocation is given by the Peach-Koehler force

$$\frac{\vec{F}}{L} = (\vec{b} \cdot \sigma) \times \hat{\xi} \ . \tag{17}$$

Inserting the stress tensor from Eq.(15), the Burgers vector, and the general line direction from Eq.(16) into Eq.(17), we find that the force on the dislocation in the *xy* plane is

$$\frac{\vec{F}}{L} = b\sigma_{yz}(-\xi_y, \xi_x) = b\,\tau(-\xi_y, \xi_x) \quad , \tag{18}$$

where we use $\tau = \sigma_{yz}$. Note that the magnitude of the force is

$$\frac{F}{L} = \left|\left|\frac{\vec{F}}{L}\right|\right| = b\tau \ , \tag{19}$$

where for a given stress, $b\tau$ is just a constant. We will use a parameter for $b\tau$ in our model.

Consider Eq.(18). At the beginning, the dislocation is aligned along the *x* axis and the only force on the dislocation is in the *y* direction. However, since the ends are fixed, the

segments near the fixed points will have both an *x* and a *y* component, leading to forces on the nodes in the *x* direction.

IIIb. Representation of the dislocation

Dislocations are, in general, curved, not straight. However, we will approximate the shape of the dislocations using a set of connected straight segments, as shown In Figure 5. That figure also introduces the notation that will be used in the MATLAB code described below. In the figure, a series of nodes has been distributed along a dislocation curve, which have then been connected by straight segments. The slightly larger black nodes at the two ends are fixed. We will find the net forces on the other nodes and will determine their motion in response to those forces, in much the same way as we moved atoms in molecular dynamics or the straight dislocations in Section II. As the nodes move, the straight segments will, of course, also change. We will foreshadow the discussion by noting that the nodes are not independent - they are abstractions representing arbitrary places on the dislocation. Thus, their dynamics show differences from what we have seen elsewhere in this course.

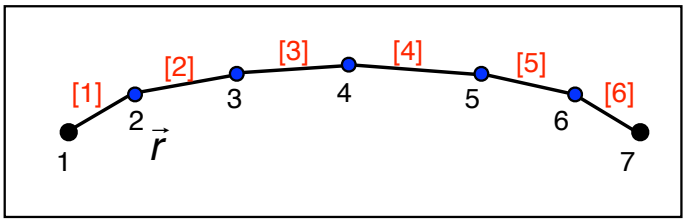

Figure 5. Schematic view showing representation of a dislocations as a series of straight segments. Also shown are the numbering schemes used in the text.

Note the two numbering systems in Figure 5. The black numbers indicate the nodes, running sequentially from the fixed point on the left to that on the right. Their positions are indicated by $\vec{r}$. The segment numbers are indicated in the figure as [i]. Note that if there are *n* nodes total, there are $n_s = n - 1$ segments.

The lengths of the segments are given by (in the notation of Figure 5)

$$\ell_i = \left|\left|\vec{r}_{i+1} - \vec{r}_i\right|\right| = \left(\left(\vec{r}_{i+1} - \vec{r}_i\right)\cdot\left(\vec{r}_{i+1} - \vec{r}_i\right)\right)^{1/2}, \tag{20}$$

while the line directions of the segments are

$$\hat{\xi}_i = \frac{\vec{r}_{i+1} - \vec{r}_i}{\ell_i} \tag{21}$$

IIIc. Forces on the nodes.

Based on Eqs.(18-21), the force on all points along the *i*th segment is the same, so the total force on the segment is

$$\vec{F}_i^s = b\,\tau\left(-\xi_{iy},\xi_{ix}\right)\ell_i \ . \tag{22}$$

In Chapter 10 of Reference [3], Bulatov and Cai describe in detail how one can calculate the force on a node based on the forces on the segments connected to the node. They are primarily concerned with cases in which the force changes along the length of the segment, which would be the usual case when the dislocations interact with each other. They showed that the net force on a node is a weighted average of the forces on the segments adjacent to the node. In the present model, there are no interactions between dislocations and the force on each segment is the uniform force given by Eq.(22). In this case, the force on a node is just the simple average of the forces on the adjacent segments,

$$\vec{F}_i^n = \left(\vec{F}_{i-1}^s + \vec{F}_i^s\right)/2 \ . \tag{23}$$

There is also a force associated with the linear change in energy as the length of the dislocation changes. If the energy per length of dislocation is E_ℓ, then the force on a node is just E_ℓ times the unit vector from that node to its adjacent node, i.e., the force is along the direction needed to decrease the length of the segment. This force can be written as

$$\vec{F}_i^\ell = E_\ell\left(-\hat{\xi}_{i-1} + \hat{\xi}_i\right) \ . \tag{24}$$

The net force on a node (in this simple model) is

$$\vec{F}_i^T = \vec{F}_i^n + \vec{F}_i^\ell \ . \tag{25}$$

IIId. Equations of motion.

In Section II we describe a model in which the structure of individual dislocations is fixed and they move as independent entities. In atomistic simulations, the atoms also move as independent entities. The equations of motion are straight forward; the forces on each entity are calculated and the velocities are given as solutions to Newton's equations. For dislocations, which have strong frictional forces, we often assume the overdamped limit of Eq.(8), which leads to the simple equation for the change of position shown in Eq.(9).

Consider a node in Figure 5. It is not a particle - it is just a marker that indicates a position on the dislocation. As such, the nodal positions and velocities are not independent of the other nodes and the velocities cannot be written in the simple form of Eq.(8). Bulatov and Cai [3] give expressions for the velocity that take into account the interconnec-

tions between the nodes. Their approximate expression for the nodal velocity, which we use in these examples, is

$$\vec{v}_i^{\,n} \approx \frac{\vec{F}_i^{\,n}}{B\left(\ell_{i-1} + \ell_i\right)/2} \ , \tag{26}$$

i.e., the force in Eq.(8) divided by the average of the lengths of the segments adjacent to the node. We use the simple Euler expression from Eq.(9) to propagate the nodal positions, though, as mentioned above, more sophisticated methods are available.[3]

IIIe. Remeshing

Consider the initial Frank-Read source in Figure 3 and compare that to a later stage in Figure 4. Suppose we start a calculation with a fixed number of nodes. As a later configuration is reached, such as in Figure 4, the dislocation has greatly increased in length. If the number of nodes is fixed, then the separation between the nodes will increase and the dislocation may not be modeled accurately. To preserve accuracy, new nodes will need to be added. On the other hand, if the separation between nodes becomes too small, then nodes might need to be removed. If dislocations formed junctions, then nodes would need to be eliminated and, perhaps, new ones added. This general procedure is often referred to as *remeshing*. Bulutov and Cai [3] discuss various remeshing procedures at length. For this exercise, we will only invoke the addition of nodes as the separation between them increases. We will simply assume that if the distance between two adjacent nodes is greater than some prescribed value, then a new node will be inserted halfway between the existing nodes.

IIIf. Implementation

We emphasize that this code is designed explicitly for modeling the beginnings of a Frank-Read source. It does not include the annihilation and remeshing necessary to create a growing loop and the activation of the source for a second time. We leave that to the exercises. In the following, the external stress and Burgers vector are lumped into a single constant `btau` = $b\,\tau$, the energy per length of dislocation is `el` = E_ℓ, the damping (friction) coefficient is `b` = B, and the time step is `dt`.

We start be assuming the position of the fixed nodes and the initial node distribution. As a simple example, we assume the fixed nodes are located at (0,0) and (10,0). We then put a node at each integer point in between the fixed nodes, e.g.,

$$\vec{r} = \{(0,0),(1,0),(2,0),(3,0),(4,0),(5,0),(6,0),(7,0),(8,0),(9,0),(10,0)\}$$

In a MATLAB code, that could be written as

```
r = [0 0;1 0;2 0;3 0;4 0;5 0;6 0;7 0;8 0;9 0;10 0];
```

Thus, the total number of nodes is `n` = 11 and the number of segments is `ns` = `n` - 1 = 10. Only the inner 9 nodes can move.

For a given distribution of nodes, we need to calculate the lengths and line directions of the segments. From these values, we can use Eq.(22) to calculate the net force on each segment. In MATLAB, those calculations could look like

```
for i=1:n-1
  dr(i,:) = r(i+1,:)-r(i,:);
  li(i) = sqrt(dot(dr(i,:),dr(i,:)));
  xi(i,:) = dr(i,:)/li(i);                                    (A)
  fi(i,:) = btau*[-xi(i,2) xi(i,1)]*li(i);
end
```

Note the construction `r(i,:)`, which indicates the complete i^{th} entry in the vector `r`, which would be the pair of (x_i,y_i) values for that position. Once we have calculated the values for the segments (Loop A), we can determine the forces on nodes using Eq.(23)-(25). From these forces, the velocity of the nodes is given by Eq.(26). Once we have the velocities, we can calculate the new positions as in Eq.(9). All of these values can be calculated in a single loop, remembering to only include the interior (non-fixed) nodes, yielding the new nodal positions for this time step.

```
for i=2:n-1
   fn1(i,:) = (fi(i-1,:) + fi(i,:))/2;
   fl(i,:) = el*(-xi(i-1,:) + xi(i,:));                       (B)
   fn(i,:) = fn1(i,:) + fl(i,:);
   vn(i,:) = 2*fn(i,:)/(b*(li(i-1)+li(i)));
   rnew(i,:) = r(i,:) + vn(i,:)*dt;
   r(i,:) = rnew(i,:);
end
```

Loops A and B will be evaluated for each time step, yielding the change in nodal positions.

As you will see in the exercises, having a fixed set of nodes yields rather inaccurate results over time (the length increases with time). We can remesh the nodes as described above by inserting an extra node midway between any connected nodes whose separation is greater than some parameter (`lmax`). We need to be careful to put the nodes in a sequential order so that we maintain the correct topology. One way to do this is to create a new array of nodal positions (`rs` in this example) in the correct order and then to rename that as `r`. Note that as we have written this code, we need to add the final fixed mode, which we do using the `cat` function.

```
% remesh
  k1=1;
  for i=1:n-1
    rs(k,:) = r(i,:);
    dr(i,:) = r(i+1,:)-r(i,:);
    lir(i) = sqrt(dot(dr(i,:),dr(i,:)));
    if lir(i) > lmax
      k1 = k1 + 1;
      rs(k1,:) = r(i,:) + dr(i,:)/2;
```

```
    end
    k1 = k1 + 1;
  end
  r=cat(1,rs,xr);
  n=length(r);
```

We have provided two working codes along with these exercies. On code has a fixed number of nodes and one remeshes the nodes as the simulation proceeds.

***Exercises**:*

A caveat: We do NOT include dislocation-dislocation interactions in the Frank-Read source code. Thus the dislocation behavior is not correct, especially when it loops back around the fixed points and the segments approach each other. Not only do they not interact as they should, there is no annihilation, which is needed for further operation of the loop.

6. Using a code without remeshing, assume values for the materials constants and run the simulation for a loop. We note that realistic parameters are not needed to study generic behavior (e.g., setting the materials parameters to 1 is not a bad way to begin). Start with a coarse mesh, as suggested above. Choose a time step that yields reasonable numerical solution (e.g., for all constants set to 1, a time step of `dt` = 0.1) should be sufficient). Compare the simulated results to the configuration in Figure 4. Note that if the simulation is run long enough, the dislocation lines that have looped around the fixed point can move through each other. This is unphysical. They have a different sense at that point and would annihilate.

7. Using the fixed node code, start with an uneven distribution of nodal points, with a higher concentration of points near the fixed nodes. Run the simulation and compare with the Problem 6. Do you notice any numerical instabilities? If so, where? Does changing the time step help?

8. Modify the code so that the system remeshes. Vary the maximum length of a segment to see how that effects the final microstructure. For the same parameters, compare the net shape of the loop with a remeshed solution with that from a coarse set of nodal points. How much inaccuracy was introduce with the fixed nodes. We note that Figure 4 was created with code provided in the exercises.

9. Explore the parameter space. For a constant value of `btau`, what happens as the line energy is increased? Is there value for line energy relative to the applied stress beyond which the source will not operate? If so, what is it? Compare with the expected value for the stress needed to operate a Frank-Read source from elementary dislocation theory.

10. Modify the code to enable annihilation of the oppositely aligned segments below the line between the fixed nodes (e.g., as in Figure 4). Introduce criteria for when annihilation will occur and justify those criteria. After annihilation, there should be a closed loop and a curved line between the fixed node. Note that this will require an identification of two types of dislocations, one with the end points fixed and the other a closed loop.

References

1. "Dislocation distributions in two dimensions," A. N. Gulluoglu, D. J. Srolovitz, R. LeSar, P. S. Lomdahl, Scripta Metallurgica 23, 1347-1352 (1989).
2. *Theory of Dislocations* , J. P. Hirth and J. Lothe, (Kreiger Publishing, Malabar, Florida, 1992).
3. *Computer Simulations of Dislocations*, V. V. Bulatov and W. Cai, (Oxford University Press, New York, 2006).
4. "Analysis of dislocation microstructures: Impact of force truncation and slip systems," H. Y. Wang, R. LeSar, and J. M. Rickman, Philosophical Magazine A 78, 1195-1213 (1998).
5. *Introduction to Dislocations*, *4th Edition*, D. Hull and D. J. Bacon (Butterworth Heinemann, Oxford, UK, 2001).