LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-501662

ParaDiS-FEM dislocation dynamics simulation code primer

M. Tang, G. Hommes, S. Aubry , A. Arsenlis

September 28, 2011

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

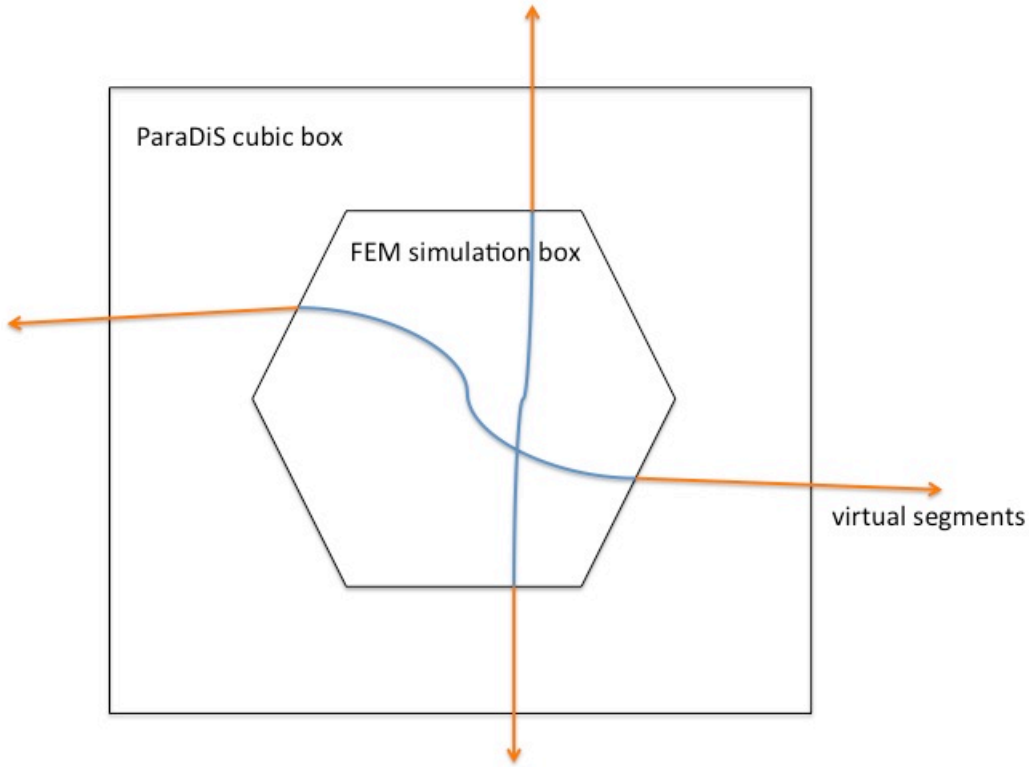
ParaDiS-FEM dislocation dynamics simulation code primer

Meijie Tang*, Gregg Hommes, Sylvie Aubry, Tom Arsenlis
Lawrence Livermore National Laboratory, Livermore, CA 94551

The ParaDiS code is developed to study bulk systems with periodic boundary conditions. When we try to perform discrete dislocation dynamics simulations for finite systems such as thin films or cylinders, the ParaDiS code must be extended. First, dislocations need to be contained inside the finite simulation box; Second, dislocations inside the finite box experience image stresses due to the free surfaces. We have developed in-house FEM subroutines to couple with the ParaDiS code to deal with free surface related issues in the dislocation dynamics simulations. This primer explains how the coupled code was developed, the main changes from the ParaDiS code, and the functions of the new FEM subroutines.

1. General setup and data structures

The general setup for the ParaDiS-FEM coupled simulation is that the ParaDiS cubic box always contains the FEM simulation box. The later describes the actual simulation geometry with dislocations inside. In order to have a kinematically viable dislocation ensemble, dislocation segments intersecting the FEM free surfaces are extended to infinity. These semi-infinite segments are termed 'virtual segments'. They contribute to the dislocation-dislocation elastic interactions only. They do not participate to plastic flow. Fig. 1 is a schematic diagram of a ParaDiS-FEM system.



In the coupled ParaDiS-FEM code, there are two general sets of data structures: the data structures associated with ParaDiS code and the data structures associated with the FEM mesh. The FEM data structures include the FEM element nodal positions, FEM surface connectivity, element surface normal vectors, and information needed to solve the boundary value elasticity problem in the FEM. The ParaDiS data structure essentially is the same as a regular uncoupled ParaDiS code. However, it is necessary for ParaDiS to identify special dislocation nodes that reside at the FEM surfaces. To conveniently handle this, two additional arrays are added to each dislocation node property in the ParaDiS node data structure. These are *node->fem_Surface[2]* and *node->fem_Surface_Norm[3]*. The former is an array of 2 integers. *fem_Surface[0]* labels whether the node is on the surface (>0) or not (=0). If it's on the surface, *fem_Surface[0]* tells which FEM element the dislocation node resides, and *fem_Surface[1]* tells which surface of the FEM element the dislocation node resides. The other array *node->fem_Surface_Norm[3]* tells the surface normal vector.

In order to allow the FEM free surfaces to setup in any arbitrary orientation, the FEM coordinate system can be different from the ParaDiS one. Two vectors *fem_ageom_x* and *fem_ageom_z* are used to specify the coordinate systems of the FEM frame. All the coordinates in the FEM data structure are defined in the FEM frame and all the coordinates in the ParaDiS data structure are defined in the ParaDiS frame. The rotation matrixes between the two frames are built in *FEM_Init* to translate coordinates between them. For example, when the subroutine *node_on_surface* is called from ParaDiS to find out if a dislocation node is on a FEM surface, the dislocation node coordinates are first translated to the FEM coordinate system using the rotation matrix *fem_ageom*.

2. Initialization

Besides the initialization for ParaDiS, a FEM initialization subroutine *FEM_Init* is called to set up the FEM geometry, mesh structure, and boundary conditions. Proper inputs are required to specify the FEM simulation geometry, its mesh resolution, and boundary condition. If these inputs are taken from the ParaDiS side, they need to be passed to the FEM data structure. Currently, we provide all the input parameters using the control file for regular ParaDiS. An example of the added inputs for ParaDiS-FEM is shown in the appendix with detailed descriptions in *FEM_Init*. In addition to the geometry related parameters, inputs such as shear modulus, Poisson ratio, and applied stresses need to be passed to FEM side of subroutines too.

A second important step of initialization is to fit the initial dislocation configuration properly into the FEM simulation box. In general, the ParaDiS code first generates the appropriate initial dislocation configurations in a large cubic box (see Fig. 1). And FEM simulation box is typically contained inside the cubic box. Therefore, the dislocation configuration needs to be cut to fit the FEM box. Dislocation segments out of the FEM simulation box are thrown away. Dislocation segments that cross the FEM box surfaces are cut, and the part of each segment outside the FEM box is thrown away. After cutting, the segments intersect with the FEM free surface are extended to infinity to create the

virtual segments (these only exist for elastic interactions, not for actual plastic deformation). This refitting and cutting of dislocation configuration is done through an added ParaDiS function *AdjustNodePosition*. Inside this function, it calls two FEM subroutines *node_on_surface.f90* and *fem_segment_surface_intersection.f90*. The first FEM subroutine takes the input of a dislocation node coordinate (x, y, z) and outputs the values for *node->fem_Surface* and *node->fem_Surface_Norm*. Also, it outputs a number *num_surfs* , which indicates how many surfaces the node intersects. When a surface node is at a corner, multiple surfaces are involved. A decision needs to be made to decide which element and surface to use for the nodal property. This decision can be made within the FEM subroutine. The second FEM subroutine takes the input of the coordinates (x1, y1, z1; x2, y2, z2) of the two ends of a dislocation segment and outputs the intersection point coordinates (xint1, yint1, zint1) and the intersection node's properties *fem_Surface* and *fem_Surface_Norm*.

After this step is done, the initialization is done and the simulation will then proceed with a dislocation configuration that contain both dislocation nodes inside of the box and nodes on the free surfaces. The following paragraphs describe the important steps during each deformation cycle.

3. Dislocation nodal force calculation

Force calculation is the first important step in each cycle. Due to free surfaces in finite box, additional forces due to the image stressed from the free surfaces need to be superimposed to the nodal force in ParaDiS. The force due to surface image stresses is calculated at the beginning of each force calculation. First, FEM constructs its traction free boundary condition. The traction forces on the FEM surfaces are from the stress fields of all dislocation segments in the simulation box as well as the virtual segments. A FEM subroutine *stress_on_boundary.f90* calculates the stress tensors from all dislocation segments at each FEM element node. This routine contains a call to a function in the ParaDiS code named *AllSegmentStress*, which in turn calculates the superimposed stress tensor from all dislocation segments at the given position. Then, the nodal forces and stiffness matrix are calculated in FEM, and FEM solver will solve the boundary value problem to obtain the image stress tensors at the Gaussian points.

After the FEM solution is obtained, the force calculation routine in ParaDiS will add the force contribution from the FEM image stress field to the segments. Currently, the FEM image stress is added at the middle point of each dislocation segment, thus superimposed to the term *node->sigbRem*. The FEM image stress at the middle point is obtained by a subroutine *fem_point_stress.f90*, which essentially finds the FEM element where the input position resides and interpret the image stress according to the Gaussian points' stress tensors and the given shape functions in the FEM element. By doing it this way, the FEM image stress field is treated as uniform along the dislocation segment. More accurate algorithms can be developed by providing the segment end positions to the FEM and numerical integrating the image stresses along the segment and to obtain the nodal force at each end of the segment. Yoffe image stresses are treated similarly as FEM image stresses. A function *AllYoffeStress* is called to obtain the Yoffe image stress due to

all segments intersecting with the free surfaces. In order to find the intersecting segments efficiently for large number of segments calculations, a list of the intersecting segment is built and updated during each simulation cycle. This is done by the function *BuildSurfaceSeg* and it's called whenever *AdjustNodePosition* is called.

The superimposition of FEM image stress and Yoffe image stress to the nodal forces is done inside the added function *ComputeFEMSegSigbRem* and *ComputeFEM1SegSigbRem.* Both are called from *NodeForce.c.* Then in *LocalSegForces*, the force due to *node->sigbRem* is calculated and accumulated into each node. In addition, the direct segment-segment interaction forces should include the interaction of segments with the virtual segments. This is done in *LocalSegForces* and using the same surface segment list by *BuildSurfaceSeg* for the interaction of a segment in the box with the virtual segments. The far field segment-segment interaction is treated exactly the same way as that in the regular ParaDiS code. In other words, FMM cells are used to calculate the forces on local segments due to remote cells. However, because no PBC in the ParaDiS-FEM, no FMM correction is needed for PBC image segments and one does not need provide any valid fmCorrectionTbl in the input (see the Appendix for input example).

4. Nodal velocity calculation

The velocity of dislocation nodes on the surfaces is constrained so that they only move on the surface, not out of the surface plane. To do this, a function called *AdjustSurfaceNodeVel* is called in every mobility function to adjust the velocity so that the component along either the dislocation line direction or the surface normal direction vanishes.

5. Time integrator

Both explicit and implicit time integrators are allowed for the coupled ParaDiS-FEM code. However, complications exist for the implicit time integrator. Within each iteration of the implicit time integrator, the dislocation nodes move by a trial distance, then forces and velocities are re-evaluated, the iterations are repeated until the final positions are found. During the trial movement, the dislocation nodes have moved. Strictly speaking, we need to re-calculate the FEM traction boundary condition and re-solve for the image stresses. This will be quite computationally demanding since the number of iterations can be quite big. What can be done is to find a way to approximate the image stress fields even with the trial movements of dislocation nodes. For example, we can keep the forces due to the FEM image stresses unchanged when re-evaluate the forces. Better off, we should probably update the image stress on the segments that intersecting the free surfaces using their updated positions. The image stresses on the intersecting segments can be large and likely to require higher accuracy. Currently, we simply keep the image stresses unchanged from the beginning of the cycle for the purpose of re-evaluating the forces.

After the nodal movements at the end of the cycle, a final step is added to finish the time integrator. The function *AdjustNodePosition* is called to update the dislocation node properties *fem_Surface* and *fem_Surface_Norm* because new surface nodes can be created and previous surface nodes may have moved to another FEM element or surface.

6. Topology handling

There is a number of places in the code dealing with surface related topology. The most important topology handling for the ParaDiS-FEM code is to identify and label dislocation nodes on the free surfaces, and cut segments that cross the FEM surface. This is done using the function *AdjustNodePosition* described earlier. Essentially, anytime when there will be a change in dislocation nodal positions either in time integrator, collision handling, multi-node splitting, or remesh, the function needs to be called to update the dislocation nodal properties.

In addition, a very specific topology handling function *SplitSurfaceNodes* is added in ParaDiS. It performs the same function as *SplitMultiNodes* in bulk ParaDiS code, but used to split multi-nodes on the free surfaces. It enlists the possible ways a surface multi-node can be splitted by taking into consideration of the fact that the surface node can have different ways of splitting.

7. Outstanding issues

The current ParaDiS-FEM coupled code lays down a blueprint for code development to study finite systems with free surfaces. Improvements can be made in several ways. Most of the improvements should be done in the FEM side. Here are some suggestions for further development:

- an efficient algorithm to calculate the traction free boundary condition from all dislocation segments in the simulation box at all FEM nodes in parallel and use the FMM table for far away segments
- an efficient algorithm to search for the FEM element # and surface # for any given point in space
- an accurate algorithm to interpret stress fields at any given point in space from the Gaussian point stress values using proper shape functions. And to perform numerical integrations along given segment to obtain nodal forces
- an efficient and accurate algorithm to re-evaluate or estimate the new forces when dislocation nodes have moved in trial distances inside the implicit time integrator

8. Summary of added subroutines/functions for the coupled ParaDiS-FEM code

In this section, a summary is given to list the main added functions/subroutines in the coupled ParaDiS/FEM code, which are not part of the original ParaDiS code. Of course all FEM related subroutines are added. Also, inputs to the subroutines/functions are pointed to indicate what data is passed between the two parts of the coupled code.

The added functions in ParaDiS code and their main functions are below.

- *AdjustNodePosition*: This function calls two FEM subroutines to find out if a dislocation node is inside/outside of the simulation box, or on the box surface. It also cuts segments that cross the simulation box surfaces, and assign values for added nodal properties *node->fem_Surface* and *fem->Surface_Norm.* It also updates the connectivity between dislocation nodes after segment cutting.

- *ComputeFEMSegSigbRem* & *ComputeFEM1SegSigbRem:* These two functions together calculate the stress tensor at a given position (currently the middle point of a segment) and superimpose them in the force contribution for each node through *node->sigbRem*.

- *AllYoffeStress*: This function calculates the Yoffe image stresses due to all segments intersecting with the free surfaces at a given point in space.

- *AllSegmentStress*: This function calculates the stresses due to all dislocation segments at a given FEM node. This function is somewhat special because it's called back from a FEM subroutine and it requires a FEM element node as input.

- *AllImageStress*: This function calculates the stresses due to the Yoffe stress fields at non-intersecting surfaces due to a segment that intersects with a FEM surface. The code structure is the same as in *AllSegmentStress*.

- *BuildSurfaceSeg*: This function checks and finds all segments that intersect with the free surfaces. Build a list of them with appropriate properties. It is updated at the end of *AdjustNodePosition*. It is used for calculating segment-virutual segment interaction forces and *AllYoffeStress.*

The new subroutines in FEM side of the code and their main functions are below.

- *FEM_Init*: This subroutine builds up the FEM mesh and its data structure. The inputs should specify the FEM simulation box shape and size, the mesh resolution, the boundary condition as well as the elastic properties such as the shear modulus and the Poisson ratio. The inputs are usually provided through the ParaDiS input control file. But it does not need to be this way.

- *node_on_surface*: This subroutine takes the input of a dislocation node coordinates and finds out if the node is outside, inside, or on the FEM box surface. The outputs contain two arrays, i.e., node->fem_Surface, and node->fem_Surface_Norm. See text for description.

- *fem_segment_surface_intersection*: This subroutine is called when a dislocation segment crosses the surface. It takes the dislocation segment end points coordinates as inputs and find out the coordinate of the intersection point, and its nodal properties including *fem_Surface* and *fem_Surface_Norm*.

- *stress_on_boundary*: This subroutines calculate the stress tensor at each FEM element node due to all dislocation segments including the virtual segments.

- *fem_boundary_node_force_term*: This subroutine calculates node force at each FEM surface node due to the segment stresses, and assign the boundary condition as specified.

- *fem_analysis*: This subroutine calculates the stiffness matrix, and solve the boundary value problem to obtain the image stress fields at the Gaussian points of each element.

- *fem_point_stress*: This subroutine takes the input of the coordinates of a given point (currently the middle point of a segment) and find out the FEM image stress value at the given point.

9. Appendix: Input example (partial list to show new parameters required to run ParaDiS/FEM code) with detailed description of parameters in *FEM_Init*.

```
## to run ParaDiS/FEM, fmEnabled is required to be set to 1.
fmEnabled = 1
fmMPOrder = 2
fmTaylorOrder = 4
fmCorrectionTbl = "/g/g20/mtang/WORK/Inputs/fm-ctab.m2.t4.dat"

#Boundary conditions
xBoundType = 1
yBoundType = 1
zBoundType = 1

#Simulation box size
xBoundMin = -476
xBoundMax = 476
yBoundMin = -476
yBoundMax = 476
zBoundMin = -119
zBoundMax = 119

### parameters describing FEM geometry/mesh/boundary condition
## Detailed description for each parameter can be found in FEM_Init.
## BC_type describes the boundary condition
BC_type = 5
## mesh_type describes the FEM geometry shape
mesh_type = 1
## maximum number of elements above which an iterative solver is
used for FEM
dirmax = 2000
## describe the number of FEM nodes along each direction. (mesh
resolution)
fem_nx = 3
fem_ny = 3
fem_nz = 3
## describe the FEM simulation box rotation with respect to the DD
frame.
fem_ageom_x = [
 1.000000e+00
 0.000000e+00
 0.000000e+00
 ]
fem_ageom_z = [
 0.000000e+00
 0.000000e+00
 1.000000e+00
 ]
```