**How-To : Set Up Dislocation Dynamics Package**

*Note*: The commands in this tutorial are written for a tutorial system, so there may be slightly differences depending on your computer.

*If starting from Windows OS…*

1. Download and partition system to install Fedora (or other Linux operating system)
   * It is highly recommended that you backup your system prior to going through this process
   * May need to format a USB to be a disk drive to install Fedora – then reboot the computer in BIOS mode
   * <https://getfedora.org/>
   * <https://www.lifewire.com/guide-to-installing-fedora-linux-2202074> (how-to guide)
   * Partitioning roughly 100-200GB should be enough to start with

*If starting from Linux OS or after successfully partitioned system…*

1. Download and install Mercurial (hg) (<https://www.mercurial-scm.org/>)
   * Package will assist with source control management and branch updates
   * sudo dnf install hg
2. Download and install VTK and VTK developer tools (<https://vtk.org/>)
   * Package will provide visualization tools for DD
   * sudo dnf install vtk
   * sudo dnf install vtk-devel
3. Download latest gcc version
   * sudo dnf install gcc-c++
4. Download Model (DD) package using Mercurial (<https://bitbucket.org/model/model/wiki/Home>)
   * Use the terminal to move to the Documents folder (or wherever you want to place the Model code)
   * hg clone https://model@bitbucket.org/model/model Model <- uses hg to pull code and creates the Model folder
   * Then, cd into the created Model folder and enter the following command:
   * hg update DiscreteCrackMechanics <- moves user to the DiscreteCrackMechanics branch
     1. This is the most commonly updated branch by Giacomo
5. Download Eigen package using Mercurial
   * Library to assist with matrix math computation throughout the dislocation dynamics code
   * Use the terminal to move to the Documents folder (or wherever you want to place the Eigen library)
   * hg clone <https://bitbucket.org/eigen/eigen/> Eigen <- uses hg to pull code and creates the Eigen folder
   * OPTIONALLY, users can create symbolic links to the Eigen package with the terminal commands;
     1. sudo ln -s /home/cmcelfresh/Documents/Eigen Eigen
     2. sudo ln -s /home/cmcelfresh/Documents/Eigen/unsupported unsupported
     3. ALTERNATIVELY, you can edit the Makefiles in the Model code (to be discussed later)
6. Download and install FFmpeg package
   * FFmpeg package assists with creating videos from the produced DD figures
   * sudo dnf install ffmpeg
7. Download and install the TetGen package (<http://wias-berlin.de/software/tetgen/>)
   * TetGen assists with generating customizable meshes for DD simulations
   * Download online [http://wiasberlin.de/software/index.jsp?id=TetGen&lang=1#Download](http://wiasberlin.de/software/index.jsp?id=TetGen&lang=1" \l "Download)
   * Follow [instructions](http://wias-berlin.de/software/tetgen/compile.html) on compiling TetGen code to create executable.
     1. Put all downloaded material in its own Tetgen folder
     2. Move into the folder using the command line
     3. make <- this should create a tetgen executable file
8. Download and install Intel Math Kernel Library
   * Find download at <https://software.intel.com/en-us/mkl/choose-download>
   * Select “Intel Parallel Studio XE”
   * Follow instructions for downloading for Linux as a student. You will be required to fill out a brief questionnaire about the usage of MKL for DD.
   * Once submitted it may take a day or so to received email confirmation of approval for the package. Download the package and follow the [Linux installation guide](https://software.intel.com/en-us/download/parallel-studio-xe-2019-install-guide-linux).
   * NOTE: During the installation make note of where the package is installed – its path will be necessary for updating the Model makefiles.

*If starting from UCLA hoffman2 (or other supercomputer)…*

1. Download Model (DD) package using Mercurial (<https://bitbucket.org/model/model/wiki/Home>)
   * Use the terminal to move to the Documents folder (or wherever you want to place the Model code)
   * hg clone https://model@bitbucket.org/model/model Model <- uses hg to pull code and creates the Model folder
   * Then, cd into the created Model folder and enter the following command:
   * hg update DiscreteCrackMechanics <- moves user to the DiscreteCrackMechanics branch
     1. This is the most commonly updated branch by Giacomo
2. Download Eigen package using Mercurial
   * Library to assist with matrix math computation throughout the dislocation dynamics code
   * Use the terminal to move to a folder outside of the newly created Model folder (or wherever you want to place the Eigen library)
   * hg clone <https://bitbucket.org/eigen/eigen/> Eigen <- uses hg to pull code and creates the Eigen folder
   * OPTIONALLY, users can create symbolic links to the Eigen package with the terminal commands;
     1. sudo ln -s /home/cmcelfresh/Documents/Eigen Eigen
     2. sudo ln -s /home/cmcelfresh/Documents/Eigen/unsupported unsupported
     3. ALTERNATIVELY, you can edit the path in the Makefile in the Model code (to be discussed later)
3. Load most recent gcc compiler.
   * Note that you may need to do this each new interactive session on hoffman2, as the default compiler may older than that required by DD.
   * Show and load gcc compilers using the following commands in the interactive terminal:
     1. module available gcc ← lists the available versions of gcc
     2. module load cmake/”most recent version #” ← loads desired version of gcc

*Update the Makefile (for both local and hoffman2/supercomputer use)…*

1. Go into the Makefile in Model/tutorials/DislocationDynamics/Makefile
2. Update the path to the Eigen library
   1. Update the “EIGEN\_INCLUDE = … “ line at the beginning to be your path for Eigen
   2. Example:
      1. EIGEN\_INCLUDE = /home/cmcelfresh/Documents/Eigen/Eigen
3. Update the path to MKL (only if using on local machine)
   1. Update the “MKL\_INCLUDE = … “ line at the beginning to be your path for MKL
   2. Note that this will only be necessary if you installed MKL on your local machine and it was not deposited where DD expected it. This step is not necessary if using DD on hoffman2
   3. Example:
      1. MKL\_INCLUDE=/home/cmcelfresh/intel/mkl/include
   4. Once MKL has been successfully linked to DD, set the usePARDISO value to 1
      1. “usePARDISO = 1”
      2. This greatly increases computational efficiency, particuarly on the FEM-aided simulations.
4. Update the compiler flags for MKL PARDISO (only if using on local Linux machine)
   1. Change line 78 to include your personal path to lib/intel64.
   2. Example:
      1. “MKL\_LIB=/home/cmcelfresh/intel/mkl/lib/intel64”
   3. During compilation of the DD code if the MKL library is not recognized, it may be necessary to directly export path using the two terminal submitted commands that include your personal path to intel/lib/intel64 and mkl/lib/intel64.
   4. Example:
      1. export LD\_LIBRARY\_PATH=/home/cmcelfresh/intel/mkl/lib/intel64:$LD\_LIBRARY\_PATH
      2. export LD\_LIBRARY\_PATH=/home/cmcelfresh/intel/lib/intel64:$LD\_LIBRARY\_PATH

*Updating visualization aid VTK…*

Dislocations Dynamics has been developed in a way that all simulations can be viewed in a frame-by-frame fashion – which provides excellent insight into the mechanisms dominating dislocation interactions. As such, it is highly recommended that the VTK visualization aid to used alongside all simulations!

HOWEVER, VTK is not currently compatible with hoffman2 (as of 2019) , so it it recommended that you run VTK through a hoffman2-mounted folder on your local machine. More on mounting later.

1. Ensure that a recent version of cmake is installed.
   1. If not, the most recent version can be installed on Fedora using:
      1. sudo dnf install cmake
   2. Similarly, the more recent versions of cmake can be loaded on hoffman2 by submitted the command
      1. module available cmake ← lists the available versions of cmake
      2. module load cmake/”most recent version #” ← loads desired version of cmake
2. Go to the tutotrials folder of your locally mounted DD code and re”make” the code to update the header files
   * 1. Go to → Model/tutorials/DislocationDynamics/finiteDomains\_NO\_FEM/uniformLoadController
     2. make
3. Compile DDvtk
   1. Go into the VTK DD folder /home/cmcelfresh/Documents/Model/tools/DDvtk
   2. Run the following commands:
      1. cmake .
      2. Make
   3. This creates a DDvtk executable that can be placed in any of DD tutorial files! DDvtk can be run before the simulation (to see initial microstructure), during the simulation (it continues to update the structure as the folders are populated), and after the simulation for post processing.
   4. Copy the DDvtk file and place it in any of the following tutorial folders:
      1. Model/tutorials/DislocationDynamics/finiteDomains\_NO\_FEM/uniformLoadController
      2. Model/tutorials/DislocationDynamics/finiteDomains\_FEM/cantileverBeam\_strainControl
      3. Model/tutorials/DislocationDynamics/finiteDomains\_FEM/uniaxialPillar\_StressControl
      4. Note that exact paths may change with updates to DD code – a good check is that the DDvtk file must be on the same folder level as your main.cpp

Note: When you are updating the the mesh files (.msh) you must also remake DDvtk.

**How-To : Run Dislocation Dynamics**

Assuming all previous steps have been followed correctly and each piece is installed, running DD can be done with the following commands. As previous, the most recent gcc compiler will be necessary to successfully run DD. Ensure that you are in the proper directory within the tutorial folders, which could be:

* Model/tutorials/DislocationDynamics/finiteDomains\_NO\_FEM/uniformLoadController
* Model/tutorials/DislocationDynamics/finiteDomains\_FEM/cantileverBeam\_strainControl
* Model/tutorials/DislocationDynamics/finiteDomains\_FEM/uniaxialPillar\_StressControl

1. Update header files
   1. make
2. Create and run the microstructureGenerator
   1. make microstructureGenerator
   2. ./microstructureGenerator
      1. generates starting microstructure
3. Remove old evl files (node/line position)
   1. make empty
4. Remove old executable, and make new
   1. make clean
   2. make DDomp
5. Run the simulation!
   1. ./DDomp

**How-To : Update the DD Code when Changes are added to Github**

When changes are posted to the MODEL code on Bitbucket, it is import to update your version to fix bugs or project-specific updates. You can check to see if changes were added by going to the Bitbucket page <https://bitbucket.org/model/model/branches/>

IMPORTANT NOTE: Prior to updating your code it is important that you ***externally backup any files that you do not want to rewrite****.* This applies to Makefile, evl\_0.txt, polycrystal.txt, DD.txt, and any other locally-specific file that you may have.

To update the code through Mercurial the the following commands within the Model folder:

1. hg pull
2. hg update --clean

After the package has been updated, it is recommended that you remake the DDvtk executable.

**How-To : Create Your Own Meshes with Neper**

Download and install Neper and all supporting libraries (gmsh, Netgen, POV-Ray, libScotch). Follow documentation found with the Neper package http://www.neper.info/

1. Execute simple Neper commands to build .tess files.

ex. To build a single-grained 1000x1000x3000 rectangular prism:

neper -T -n 1 -domain “cube(1000,1000,3000)”

1. Then open the mesh in gmsh and save it as a ASCII version 2.2 .msh file.

Than be performed by either:

* + 1. gmsh → then open the .tess or .geo file → export as ASCII version 2.2 msh, or
    2. Neper command line operations such as

neper -M n1-id1.tess (assuming the .tess file previously produced is n1-id1.tess)

1. The location of the .msh file should then be specified in the inputFiles/polycrystal.txt file.

Note: After a new .msh file is generated and specified in polycrystal.txt, the corresponding microstructureGenerator, Ddvtk, and Ddomp executables must ALSO be regenerated.

**How-To : Record Video using FFmpeg**

**How-To : Navigate the MODEL code using doxygen**

**How-To : Change the orientation of the crystal**