5/6/2011

LAWRENCE LIVERMORE NATIONAL LABORATORY

PARADIS USER'S GUIDE
VERSION 2.5.1 - PUBLIC

UCRL-CODE-225691

Author(s): A. Arsenlis, S. Aubry, V. Bulatov, W. Cai, G. Hommes, T. Oppelstrup, T. Pierce, M. Rhee, M.Tang

This work was produced under the auspices of the Lawrence Livermore National Security, LLC (LLNS) under Contract No. DE-AC52-07NA27344 for the U.S. Department of Energy, National Nuclear Security Administration.

Copyright is reserved to Lawrence Livermore national Security LLC for purposes of controlled dissemination, commercialization through formal licensing, or other disposition under terms of contract # DE-AC52-07NA27344, DOE policies, regulations and orders, and U.S. statutes.

Disclaimer

This work was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government, nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the Unites States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Notification of Commercial Use

Commercialization of this product is prohibited without notifying the Department of Energy (DOE) or Lawrence Livermore National Laboratory (LLNL).

Table of Contents

1 Introduction

This guide describes the Parallel Dislocation Simulator (ParaDiS), a Dislocation Dynamics simulation code developed jointly by a team of physicists and computer scientists at Lawrence Livermore National Laboratory. The code was specifically written to perform well on massively parallel computer systems in order to address the large computational expense involved in Dislocation Dynamics (DD) calculations. The ParaDiS approach to DD is to represent the material subjected to deformation by a volume populated with dislocation lines and evolve this population using known equations of dislocation theory. A single step of a simulation consists in general of (1) computing forces on the properly discretized dislocation lines, (2) moving the lines in response to forces, (3) identifying instances in time and space when and where dislocation lines collide (intersect) or dissociate and (4) re-meshing the lines to better represent their evolving geometry.

The guide contains basic information on ParaDiS, describes how to build the code, discusses the input and output file formats as well as descriptions of the control file parameters, and provides information on various utilities for creating tables and initial dislocation configurations.

2 Building ParaDiS

2.1 Directory Structure

The ParaDiS release consists of a file which, when unzipped and untarred creates a primary directory and a structure of subdirectories. For convenience, this primary directory will hereafter be referred to as <ParadisDir>. The directory structure of the release will look like:

<ParadisDir>/

Contains primary 'make' files for controlling build of all executables

<ParadisDir>/bin/

This directory will be created during the 'make' process. All executables will be placed in this directory during compilation

<ParadisDir>/docs/

Contains any additional documentation

<ParadisDir>/include/

Contains all C and C++ header files

<ParadisDir>/inputs/

FMM and non-FMM correction tables, X-display defaults file, gnuplot command file, etc.

<ParadisDir>/src/

All C and C++ source code modules related to the ParaDiS executable

<ParadisDir>/tools

Miscellaneous support scripts

<ParadisDir>/utilities

Source code modules pertaining to various support tools

2.2 Compiling

The current ParaDiS 'make' files support compilation for a number of pre-defined system types that are defined in the file 'makefile.sys'. In order to compile the code on one of these pre-defined system types, simply edit 'makefile.setup' and set the "SYS=..." value to the desired system type. For other system types, add a new system type into 'makefile.sys' following the format used for the existing systems and, as above, set the "SYS=..." value in 'makefile.setup'.

Executing 'gmake' with no options in <ParadisDir> will build the following executables:

<ParadisDir>/bin/paradis

Main parallel application

<ParadisDir>/bin/paradisgen

Utility for creating initial dislocation configurations (See Utilities section for details)

<ParadisDir>/bin/paradisrepart

Utility for generating a new domain decomposition for an existing dislocation configuration (See Utilities section for details)

<ParadisDir>/bin/paradisconvert

Utility for converting older format ParaDiS control parameter files to the current format (See Utilities section for details)

<ParadisDir>/bin/ctablegen

Utility for creating image correction tables needed when FMM code is enabled. (See Utilities section for details)

<ParadisDir>/bin/ctablegenp

Parallel version of the ctablegen utility. See above.

<ParadisDir>/bin/stresstablegen

Utility for creating tables used in calculating far-field stress if the FMM code is not enabled. (See Utilities section for details)

<ParadisDir>/bin/calcdensity

Utility for generating a density field grid and writing it out.

2.3 Make files

Compilation of the code is done via 'gmake' and depends on the following 'make' files located in the <ParadisDir> directory. (See the individual make files for details)

makefile

> This is the primary make file controlling the build of the parallel executable and associated utilities.

makefile.srcs

> Defines all the source modules [other than the source containing main()] that are compiled and linked into the ParaDiS executable.

makefile.sys

> This file contains, for each supported system type, a set of 'make' macros, definitions, compiler selection, library, and include file paths, etc.

makefile.setup

> Contains numerous 'make' settings and flags that are not system specific, including settings for system type, optimization level, debugging flags, etc.

3 Executing ParaDiS

3.1 Command Line

The ParaDiS command line format is:

paradis [-r <numCycles> [-n <numThreads>] [-b] [-d <dataFile>] <controlFile>

where:

<ctrlFile>

Specifies the name of the ParaDiS control parameter file

-b

Indicates the nodal data portion of the ParaDiS restart file is in a binary (HDF5) format.

-d <dataFile>

Specifies the base name of the file(s) containing the nodal data for the simulation run. If this file name is not supplied, the code looks for a data file named the same as the control parameter file with the suffix (if any) replaced with ".data".

-n <numThreads>

Specifies the maximum number of threads to use within a process. This option applies only if the executable has been compiled with OpenMP support. (See definition for OPENMP_MODE in the file 'makefile.setup')

-r <numCycles>

Causes the code to execute a series of <numCycles> cycles during which no force calculations or dislocation movement will occur. These cycles will be used strictly for load-balancing purposes and will be done before the normal cycles.

3.2 Examples

For example, assuming you had a control parameter file ‘testrun.ctrl’ and the two different nodal data files ‘testrun.data’ and ‘testrun2.data’, all of the following command line formats are valid. The first two commands are equivalent, the third uses the same control parameter file as the first two, but explicitly selects a different nodal data file.

mpirun –np 8 bin/paradis testrun.ctrl

mpirun –np 8 bin/paradis –d testrun.data testrun.data

mpirun –np 8 bin/paradis –d testrun2.data testrun.data

NOTE: The number of processors specified to mpirun (or other parallel job initiator applicable to the executing system) must match the number of domains specified in the control parameter file (i.e. np = numXdoms * numYdoms * numZdoms).

Several small sample problems are included with the ParaDiS release. See the <ParadisDir>/tests directory for a list of available samples and see the comments at the beginning of each control file for details on the sample. These test require the following table for calculating the far-field forces from remote dislocation:

<ParadisDir>/inputs/fm-ctab.Ta.600K.0GPa.m2.t5.dat

This file *may* have been provided with the source code release. If not, see the README file in the <ParadisDir>/tests directory for instructions on generating this file.

Among the sample simulations provided are:

<ParadisDir>/tests/frank_read_src.ctrl

<ParadisDir>/tests/frank_read_src.data

<ParadisDir>/tests/form_binaryjunc.ctrl

<ParadisDir>/tests/form_binaryjunc.data

<ParadisDir>/tests/fmm_8cpu.ctrl

<ParadisDir>/tests/fmm_8cpu.data

The ‘form_binaryjunc*’ files contain a small configuration that will demonstrate the formation of a binary junction from two dislocation lines, while the ‘frank_read_src*’ files contain a configuration that demonstrates the behavior of a

frank-read source. Both of these samples are single cpu samples. To execute them, from the main <ParadisDir> directory execute:

mpirun –np 1 ./bin/paradis ./tests/form_binaryjunc.ctrl

or

mpirun –np 1 ./bin/paradis ./tests/frank_read_src.ctrl

The 'fmm_8cpu*' files are just a general demonstration of the behavior of a small number of screw dislocations. This is an 8 processor simulation which can be executed by:

mpirun –np 8 ./bin/paradis ./tests/fmm_8cpu.ctrl

Any output from these runs will be placed into corresponding sub-directories under the <ParadisDir>tests directory.

Additional samples may be added to this directory over time. See the <ParadisDir>/tests directory for a complete list of all available samples, and see the comments at the top of each control parameter file for a brief description of what the sample is demonstrating.

4 Input Files

A ParaDiS simulation may be started from scratch, or may be restarted from a previously terminated simulation if the previous simulation had periodically created restart files. In either case, the initial or restart data consists of two files, a control parameter file and a nodal data file. The formats of these files are discussed below.

4.1 Control Parameter File

The control parameter file consists of data of the forms:

identifier = value

identifier = [value_list]

identifier = string

where:

identifier

Is the name of a control parameter

value

Is a single numeric value associated with the parameter specified by <identifier>

value_list

Is a list of numeric values to be associated with the array specified by <identifier>. The values in this list must be delimited by white-space or line-feeds.

string

Specifies a character string enclosed within either single or double quotes. The string may not contain a line-feed character.

The identifier names are case-insensitive and may be specified as any mixture of upper and lower case. If the code encounters an identifier it does not rrecognize,

the identifier and associated value(s) will be ignore and a warning message displayed.

Any blank lines in the control parameter file will be ignored, and any '#' not contained within quotes will be treated as the beginning of a comment and results in the remainder of the current line to be ignored.

See the Appendix for a complete list of the recognized control parameters as well as a brief description of each.

4.2 Nodal Data File

4.2.1 Data File Format

The nodal data is contained in 1 or more file segments in which the information is broken into three sections (described below). The first section is the data file parameters, the second is the domain decomposition, and the third the nodal data. The first two sections are included only in the first file segment.

4.2.2 Data File Segments

Given a control parameter file 'rs0010', the associated nodal data file(s) would be named:

rs0010.TYPE[.SEQ]

where:

SEQ

Is a sequence number appended only when the nodal data file was written in parallel (i.e. the <numIOGroups> control parameter and total number of domains were both greater than 1). <SEQ> will range from zero to <numIOGroups> - 1.

TYPE

Is 'data' for regular text restart files or 'hdf' for HDF5 format restart files.

For example, if the <numIOGroups> control parameter was set to 4 for a 16 processor simulation in which dumped a text format restart file at termination, the following restart file set would be generated and could be used as input to continue the simulation at a later time:

<outputDir>/restart/restart.cn

<outputDir>/restart/restart.data.0

<outputDir>/restart/restart.data.1

<outputDir>/restart/restart.data.2

<outputDir>/restart/restart.data.3

4.2.3 Data File Parameters

The data file parameters make up the first section of the nodal data file and are specified and parsed in the same manner as described above for the control file parameters. These parameters must precede the other two types of information in the nodal data file. Note: The values of these parameters are updated within the ParaDiS code as necessary and should not be changed by the user.

The recognized data file parameters are:

dataFileVersion

numFileSegments

minCoordinates

maxCoordinates

nodeCount

dataDecompType

dataDecompGeometry

4.2.4 Domain Decomposition

TBD

4.2.5 Nodal Data

The raw nodal data comprises the third section of the nodal data file. For each node, there will be a single line of node specific data followed by several lines of segment specific data for each segment associated with the node. The nodal data consists of:

node_tag x_coord y_coord z_coord num_segments constraint

The segment specific data is two lines containing the following:

neighbor_tag burg_x burg_y burg_z

norm_x norm_y norm_z

where:

*_tag

Is a comma delimited pair of integers uniquely identifying the node. The first number is the ID of the domain owning the node, the second is the index number of the node within the domain.

x_coord, y_coord, z_coord

Coordinates of the node

norm_x, norm_y, norm_z

Components of the glide plane normal for the segment

num_segments

Number of segments associated with the node

constraint

Integer value indicating any constraints placed on the node (i.e. a constraint of 7 implies a node is pinned in place and unmovable).

5 Cell and Domain Structure

The basic ParaDiS code is used to simulate cubic systems which are simultaneously partitioned into a uniform mesh of cubic ‘cells’ and spatially decomposed into processor ‘domains’.

The cellular structure is defined by the <nXcells>, <nYcells> and <nZcells> control parameters and is used to determine the cut-off distance between direct segment to segment dislocation interactions and remote (or far-field) interactions. In particular, for a given dislocation, the interactions between the segment and any other segments in the same cell or any of the immediately neighboring 26 cells are calculated directly. Interactions with all segments beyond that range are calculated via a hierarchical Fast Multipole Method, or by lumping all segments in the remote cell into a ‘super-dislocation’ where the group of remote dislocations are represented as an expansion of dislocation multipoles. (See section on Far-Field Dislocation Interactions for details)

The type of spatial decomposition used for the simulation is selected by the <decompType> control parameter, along with the <nXdoms>, <nYdoms> and <nZdoms> parameters defined the number of processing domains in each dimension. Each ‘domain’ is then assigned to a single processor within the simulation. (See section on Domain Decomposition for details on the supported types of spatial decomposition)

6 Domain Decomposition

6.1 Selecting a Domain Decomposition Method

ParaDiS simulations are spatially decomposed into a number of domains equal to the number of processors on which the simulation is being executed. The code currently supports two types of domain decomposition and uses the <decompType> control parameter to select between them. Type 1 uses a Recursive Sectioning algorithm while type 2 (the default if none is explicitly requested) uses a Recursive Bisectioning algorithm. (See descriptions of decomposition algorithms below.)

Since the ParaDiS simulations tend to grow in size and are spatially heterogeneous, it is preferable to dynamically recalculate the domain decomposition at intervals during the simulation in order to rebalance the workload more efficiently. The frequency with which the decomposition will be recomputed is specified by the <DLBfreq> control parameter.

The domain decomposition will be included in the restart files. If the simulation is restarted and the <decompType> control parameter selects a decomposition type other than the type that was used when the restart file was generated, the old domain decomposition from the restart file will be ignored and a new domain decomposition of the selected type will be used.

Additionally, if a simulation is restarted with a different number of domains or a different domain geometry than that which was used when the restart file was created, the decomposition from the restart file will again be ignored and a new decomposition will be generated. Note: When restarting a large simulation (i.e. many thousands of processors) in such a way the old domain decomposition must be discarded, it may take some time for the simulation to re-converge on an optimal domain decomposition. (The type 1 decomposition is primarily susceptible to this.) There are two ways to mitigate this effect. The first is to use the ‘paradisrepart’ utility, the second through the use of the ‘-r’ command line option to ParaDiS. See the “Utilities” and “Executing ParaDiS” sections of the manual for more information pertaining to these capabilities.

6.2 Recursive Sectioning

The Recursive Sectioning algorithm performs a domain decomposition over a 3-timestep period. During the first timestep, the entire problem space is sectioned along the X dimension into <nXdoms> slabs such that the computational cost of each slab is roughly equivalent. The next timestep, each slab is sectioned along the Y dimension into <nYdoms> columns such that the computational cost of each column within a slab is roughly equivalent. On the third timestep, every column is sectioned along the Z dimension into <nZdoms> chunks such that the computational cost of each chunk is a column is roughly equivalent.

As stated above, the frequency with which the domain boundaries will be recalculated is controlled by the <DLBfreq> control parameter. If this parameter is not explicitly provided, the default frequency for this type of domain decomposition is every third timestep. Note: Due to the fact that this algorithm requires 3 timestep to complete a new decomposition, the <DLBfreq> value must be no less than 3.

6.3 Recursive Bisectioning

The Recursive Bisectioning algorithm begins with the entire problem space and bisects the space into in the X, Y and/or Z dimensions into octants, quarters or halves (depending on the number of domains specified per dimension) such that the per-domain computational cost of each sub-partition is roughly the same. The decomposition is then recursively applied to each of the sub-partitions until no further decomposition is necessary.

As stated above, the frequency with which the domain boundaries will be recalculated is controlled by the <DLBfreq> control parameter. If this parameter is not explicitly provided, the default frequency for this type of domain decomposition is every timestep.

7 Dislocation Mobility

7.1 Selecting a Mobility Module

One of the crucial aspects of a ParaDiS simulation is the selection of the set of rules governing the material specific physics motion of the dislocations through the material including such aspects as glide, climb and cross-slip with respect to crystallographic constraints. ParaDiS provides multiple sets of rules (or 'mobility laws'), each implemented in a separate module that may be selected via the <mobilityLaw> control file parameter.

7.2 Available Mobility Modules

The currently supported mobility modules are:

- "BCC_0"
- "BCC_0b"
- "BCC_glide"
- "FCC_0"
- "FCC_0b"
- "FCC_climb"

For a list of the control file parameters related to the mobility of dislocations, see the "Material and Mobility Parameters" section of the Appendix detailing the control file parameters. Note: not all mobility parameters are applicable to all mobility functions. See the descriptions of the individual mobility functions below for information on which parameters apply.

Currently, the default values of mobility related parameters correspond to Tantalum at a temperature of 600K and a pressure of 0GPa.

7.2.1 BCC_0 Mobility

In BCC metals, screw dislocations do not dissociate into partial dislocations the same way they do in FCC metals, therefore, for BCC crystals we do not assign glide plane normal values to screw dislocations.

Instead, screw dislocations are given the same mobility in all directions perpendicular to the line. This isotropic mobility for screws mimics the 'pencil-glide' behavior of dislocations observed in BCC metals at elevated temperatures. At the same time, the drag coefficient for non-screw segments will remain anisotropic with respect to glide and climb.

Other than the general material specific parameters in the control file, dislocation motion with the BCC_0 mobility is primarily affected by the following control parameters:

MobClimb

MobEdge

MobScrew

For further details on this mobility module, refer to the paper:

<ParadisDir>/docs/ParaDiSAlgorithm.pdf

Optional inertial terms have been included into this mobility module to account for dislocation mass. These inertial terms can be enabled by setting the <includeInertia> flag to 1 and defining <massDensity> in the control parameter file. By default, these inertial terms are not included.

7.2.2 BCC_0b Mobility

The BCC_0b mobility module is nearly a duplicate of BCC_0 with the exception that the movement of discretization nodes along the dislocation line has been dampened.

Other than the general material specific parameters in the control file, dislocation motion with the BCC_0 mobility is primarily affected by the following control parameters:

MobClimb

MobEdge

MobScrew

Optional inertial terms have been included into this mobility module to account for dislocation mass. These inertial terms can be enabled by setting the <includeInertia> flag to 1 and defining <massDensity> in the control parameter file. By default, these inertial terms are not included.

7.2.3 BCC_glide Mobility

The BCC_glide mobility module is based heavily on the BCC_0 module with the addition that dislocation motion is confined to the dislocation's glide plane. Use of this module automatically enables two features. First is the mechanism which allows BCC screw dislocations to cross-slip to a different glide plane. Secondly it enables the <enforceGlidePlanes> control parameter toggle which prevents the remesh functions from removing discretization nodes attached to two segments on different glide planes.

Other than the general material specific parameters in the control file, dislocation motion with the BCC_0 mobility is primarily affected by the following control parameters:

MobEdge

MobScrew

7.2.4 FCC_0 Mobility

The FCC_0 mobility module attempts to simulate easy glide in FCC materials. The glide plane is limited to one of the [111] planes. Also, no crystallographic information is used in the dislocation core reactions and hence junction formation can take place even slightly off the zone axis.

Use of this module automatically enables the <enforceGlidePlanes> control parameter toggle which prevents the remesh functions from removing discretization nodes with segments on different glide planes. Additionally, a mechanism is provided to allow dislocations to cross-slip to new glide planes. This cross-slip mechanism is enabled/disabled via the

<enableCrossSlip> control parameter, and by default is enabled with this mobility module.

Other than the general material specific parameters in the control file, dislocation motion with this mobility module is primarily affected by the following control parameters:

MobScrew

MobEdge

7.2.5 FCC_0b Mobility

This module is an alternate version of a generic FCC mobility. It is very similar in function and structure to the BCC_glide module although glide planes rather than burgers vectors are used to identify junctions. Additionally this module attempts to identify nodes that both have short segments and have reversed the direction of their velocity. When found, the velocity of these nodes is artificially slowed for a single timestep in an effort to dampen motion in oscillating/flickering nodes.

Use of this module automatically enables the <enforceGlidePlanes> control parameter toggle which prevents the remesh functions from removing discretization nodes with segments on different glide planes. Additionally, a mechanism is provided to allow dislocations to cross-slip to new glide planes. This cross-slip mechanism is enabled/disabled via the <enableCrossSlip> control parameter, and by default is enabled with this mobility module.

Other than the general material specific parameters in the control file, dislocation motion with this mobility module is primarily affected by the following control parameters:

MobScrew

MobEdge

7.2.6 FCC_climb Mobility

This module is yet another alternate version of a generic FCC mobility. It is very similar in function and structure to the BCC_glide module although glide planes rather than burgers vectors are used to identify junctions.

Additionally, it permits dislocations a small amount of climb which allows them to drift out of their original glide planes.

Use of this module automatically enables the <enforceGlidePlanes> control parameter toggle, but also enables an internal flag that allows some ‘fuzziness’ in the glide planes. This gives the remesh functions some flexibility in rediscretizing nodes with segments on different glide planes and can potentially improve the overall timestep attainable in the simulations. Additionally, a mechanism is provided to allow dislocations to cross-slip to new glide planes. This cross-slip mechanism is enabled/disabled via the <enableCrossSlip> control parameter, and by default is enabled with this mobility module.

Other than the general material specific parameters in the control file, dislocation motion with this mobility module is primarily affected by the following control parameters:

MobScrew

MobEdge

8 Material Properties

8.1 Specifying Material Properties

All material specific properties are set via the control file parameters. If not specified, defaults will correspond to Tantalum at a temperature of 300K and pressure of 0GPa.

See the “Material and Mobility Parameters” section of the Appendix for a list of the material related control parameters.

9 Forces on Dislocation Segments

As mentioned earlier, a cellular grid is imposed on ParaDiS simulations and is used to determine the cut-off distance at which the simulation switches from direct segment to segment (i.e. local) interactions to remote (or far-field) interactions. In particular, for a given dislocation segment, the interaction between the segment and any other segments in the same cell or any of the immediately neighboring 26 cells are calculated directly. Interactions with all segments outside that range are calculated by all segments in the remote cell into a 'super-dislocation' where the group of remote dislocations is represented as an expansion of dislocation multipoles.

9.1 Local Forces

The 'local' component of force on any given dislocation segment is composed primarily of the forces from direct interactions with every other segment within the same or immediately neighboring cell, Peach-Koehler forces from externally applied stress, and self-force form the segment itself.

Additionally, ParaDiS can calculate the osmotic force associated with a super saturation of vacancies in the crystal. The force is in the climb direction of the dislocation if the dislocation has any edge character, and the force vanishes for screw dislocations. The model follows the framework proposed by Mordehai et al. When this capability is activated, there is a balance between the climb of dislocations and the vacancy concentration in the simulation. This component of the force is optional and is not enabled by default. It may be enabled by setting the control file parameters <vacancyConc> and <vacancyConcEquilibrium> where <vcacancyConc> is the number density of vacancies per atomic volume, and <vacancyConcEquilibrium> is the number density of vacancies in thermal equilibrium per atomic volume. The units of these parameters are tied to the other units of the simulation, in particular Boltzmann's constant (Joules/Kelvin) and atomic volume (burgMag^3).

9.2 Far-Field Dislocation Interactions

9.2.1 Fast Multipole for Far-Field Interactions (FMM)

The FMM algorithm is enabled within the code via the <fmEnabled> toggle in the control parameter file. The order of the multipole and taylor expansions used by the FMM algorithm are set via <fmMPOrder> and <fmtaylorOrder> control parameters respectively. As a general rule, the taylor expansion order should be approximately twice that of the multipole expansion order. One additional control parameter <fmCorrectionTbl> specifies the name of the file containing a table used by the FMM code to adjust the calculated stress to account for the multiple periodic images in the simulation.

The FMM is based on the cellular grid defined by the <nXcells>, <nYcells> and <nZcells> control parameters. There are, however, two restrictions associated with the current implementation of the FMM.

1. The number of cell must be the same in all dimensions
2. Each cell must represent a cubic (or very nearly so) portion of the problem space. This is needed because the algorithm makes use of certain symmetries in forces, but these symmetries disappear if non-cubic cells are used.

It is important to note that the file indicated by <fmCorrectionTbl> must be built with the same multipole expansion order, taylor expansion order and poisson value (<pois> control parameter) specified for the simulation. An appropriate correction table may be generated via the ‘ctablegen’ utility (see section on Utilities for details on the use of ‘ctablegen’), however a correction table matching the code’s default control parameter settings *may* have been provided in the code release as:

<ParadisDir>/inputs/fm-ctab.Ta.600K.0GPa.m2.t5.dat

The use of a Fast Multipole Method for calculating the far-field forces is based on formulae for generating and evaluating multipole expansions and a few translation theorems. A very general description can be found below, but for details refer to the following paper included with the source release.

<ParadisDir>/docs/ParaDiSAlgorithm.pdf

The general FMM algorithm consists of the following steps:

1. Construct multipole moments: each domain (processor) calculates the contributions of its dislocation segments to the multipole moments of the FMM sub-cells to which those segments belong, and communicates those contributions to the domain that owns the FMM sub-cell. Each domain that owns an FMM sub-cell sums all the contributions for that sub-cell together.

2. Upward pass: starting at the lowest layer of the FMM hierarchy, each domain collects and sums the contributions to the multipole moments for each of the FMM sub-cells it owns from the eight child sub-cells, calculates the upward pass translation of its multipole moments and communicates the result to the domain owning the sub-cells parent, until the top of the hierarchy is reached.
3. Transverse translations: the multipole moments from 189 cells that are outside the nearest neighbor distance of the target cell but inside the nearest neighbor distance of its parent are collected by the domain owning the target cell, and their contributions to the taylor series expansion of the stress field in the target cell are calculated.
4. PBC correction: the domain that owns the FMM cell at the highest level of the FMM hierarchy calculates the taylor series expansion of the stress state due to the periodic images of the system.
5. Downward pass: starting with the highest level of the FMM hierarchy, each domain that owns an FMM cell sums the contributions from its parent to its taylor series expansion of the stress from step 3, then calculates the downward pass translation of the stress for each one of its child cells and sends the results to the domain owning the child sub-cells until the bottom of the hierarchy is reached.
6. Each domain that owns a sub-cell at the lowest level of the FMM hierarchy communicates the taylor series expansions of the stress filed to the domains intersecting the sub-cell.

9.2.2 Non-FMM Far-Field Interactions

Although we recommend the use of FMM for far-field interactions, the FMM in ParaDiS can be disabled by setting the control parameter <fmEnabled> to zero. When FMM is disabled, ParaDiS requires additional tables to factor in the far-field stresses from distant cells and periodic images of the problem space. The names of the files containing the tables are specified via the <Rijmfile> and <RijmPBCfile> control parameters. These tables can be generated using the 'stresstablegen' utility (see the 'Utilities' section for more details), however, copies of these tables *may* have been provided in the code release as:

<ParadisDir>/inputs/Rijm.cube.out

<ParadisDir>/inputs/RijmPBC.cube.out

This method essentially lumps all dislocations in a cell into a ‘super-dislocation’ where the cell’s dislocations are represented as an expansion of the dislocation multipoles. The remote stress for a segment in a given cell is then calculated from two components. The first is the sum of the stresses resulting from expansions from all remote cells (i.e. neither the current cell nor any of its 26 immediate neighbors) and the stress from all periodic images of those remote cells. The second component consists of the stress only from the periodic images of the local cells (i.e. the current cell and its immediate neighbors).

This method can be faster for smaller simulations, but unlike the FMM, its performance does not scale well as the size of the simulation increases in both dislocation density and number of cells.

10 Discretization (Remesh)

The nature of DD simulations is such that the length of dislocation lines can change dramatically during the course of a simulation. Hence, rediscretization of the dislocations is an absolutely necessary component of ParaDiS simulations. The goal of the rediscretization is to optimize the numerical description of the continuous dislocation line geometry so that a given level of accuracy is achieved with the fewest degrees of freedom. For regions of high curvature, an optimal distribution of nodes will place nodes more closely together than in regions of lower curvature.

The level of accuracy is tied to the control parameters <maxSeg> and<minSeg> which define the maximum and minimum desired discretization segment lengths (in units of b) respectively. The smaller the maximum segment length specified, the higher the accuracy. (To clarify, a segment between two physical nodes, or connected to segments on differing glide planes is not considered a 'discretization' node and thus may be shorter than <minSeg>.)

A number of rediscretization methods have been developed and tested, although only versions 2 and 3 are currently supported. The rediscretization method to be used is selected via the <remeshRule> control parameter.

10.1 Remesh Method 2

This method achieves rediscretization through two types of operations; mesh coarsening and mesh refinement (deletion and insertion of discretization nodes respectively).

This involves defining both minimum and maximum discretization areas (A_min and A_max) based on the simulation minimum and maximum segment lengths. Then, for each discretization node (i.e. node with less than three associated segments on identical glide planes), the discretization area (A_node) is calculated. This area is defined as the triangle with vertices at the node and its two neighbors. When the discretization area associated with the node is less than A_min, the node is removed (coarsened out). Conversely, if the area A_node > A_max, the local discretization is refined by bisecting one or both dislocation segments attached to the node.

For additional information, refer to the following paper included with the ParaDiS source release:

<ParadisDir>/docs/ParaDiSAlgorithm.pdf

10.2 Remesh Method 3

This rediscretization method is identical to method two with the exception that during mesh refinement, the inserted nodes are not placed at the exact midpoint of the segment being bisected. Instead, the algorithm treats the three initial nodes as if they were on an arc and places the new node at the center of the arc.

The rationale behind this is that in simulations using long segments and/or high stress, a new node added at the midpoint of a segment in a region of high curvature will not be optimally placed. This new node may immediately accelerate quickly toward its optimal position then decelerate as it nears that position. This behavior can severely impact the simulation timestep. By placing the new node on an arc, the initial stress on the new node will hopefully not be as high, keeping the motion of the node similar to that of the surrounding nodes, and hence no detrimental effects on the timestep.

11 Simulation Timestepping

The ParaDiS code currently provides multiple algorithms to control simulation timestepping. The <timestepIntegrator> control parameter is used to select the desired algorithm at execution time. The two currently supported timestep integration methods are:

"trapezoid"

"forward-euler"

The default integrator is the "trapezoid" integrator. A brief description of each of the methods is given below.

The timestepping algorithms are affected by a number of control file parameters. For a list and description of these parameters, see the "Simulation and Timestepping Controls" section of the Control Parameters table in the Appendix.

11.1 Trapezoid Timestep Integrator

This implicit integrator is a mix of the Forward-Euler and Backward-Euler methods. This mechanism is unconditionally stable, but requires an iterative process that may involve multiple nodal force and velocity calculations. The additional expense of the multiple calculations, however, is usually offset by the gains from the larger timesteps attained in comparison to such methods as the Forward-Euler method.

The algorithm uses the current nodal velocities to reposition each node then recalculates forces and velocities for the nodes at their new positions. A positioning 'error' is calculated for each node based on the current and previous node forces and velocities. If the positioning error of any node exceeds the maximum allowed positioning error as defined by the <rTol> control parameter, the timestep will be decremented by the factor specified in <dtDecrementFactor> and the process starts again. On the other hand, if the positioning error of all nodes is within the tolerance, the timestep is accepted and the current timestep is multiplied by the factor given in <dtIncrementFactor> to be used as the initial timestep to attempt the next time cycle.

11.2 Forward-Euler Timestep Integrator

This timestep integrator is relatively simple and inexpensive in that it requires only a single calculation of force and velocity per cycle. Unfortunately, the algorithm is subject to the Courant condition for numerical stability and is limited to relatively small timestep. The size of the timestep is controlled by the ratio between the lengths of the shortest segment and the velocity of the fastest moving node. Additionally, the <rmax> control parameter defines the maximum distance any node is permitted to move during a single timestep which further limits the timestep length. This <rmax> distance must be set such that no dislocation node crosses multiple simulation cells in a single timestep.

Note: This timestep integrator is NOT recommended

12 Visualization

12.1 X-window Display

ParaDiS provides a simple X-window display capability for visualization and debugging of small scale simulations as well as obtaining certain types of nodal data via the display window.

Unlike some of the other visualization capabilities, this one must be enable/disabled via the compile-time flag "XLIB_MODE" in the file <ParadisDir>/makefile.setup. The X-window support is enabled by setting

XLIB_MODE = ON

And disabled by

XLIB_MODE = OFF

The default behavior is to have the X-window support enabled.

The <winDefaultsFile> control parameter can be used to specify default visualization options and attributes such as view perspective, colors and so on. Unless otherwise specified, this control parameter will point to the following defaults file provided with the source release:

<ParadisDir>/inputs/paradis.xdefaults

Once the X-window display is initiated, the view can be controlled through the following single-key commands:

Key	Command Description
<Home>	Restores the image to the default view
<Esc>	Terminates the X-window display without terminating the simulation
a	Enable/disable aspect ratio changes. When enabled, the Arrow keys alter the aspect ratio
c	Enable/disable slice view of the image. When enabled, Up/Down Arrows control slice position while Left/Right Arrows

	control the slice thickness.
f	Turn display frame on/off
p	Pause/restart simulation
r	Enable image rotation. When enabled, rotation can be control via the mouse or Arrow keys
s	Enable display scaling. When enabled, the Arrow keys control scaling size
T	Enable image translation. When enabled, the Arrow keys control translation direction
<F10>	Generates a postscript file image of the display window. Output file will be named <outputDir>/YshotNNNN where 'NNNN' is a sequence number incremented each time a dump of the display window is generated and <outputDir> is the director specified by the <dirname> control file parameter

Additionally, clicking the mouse on a nodal point in the X-window image will cause the following information to be written to the terminal device (not the X-window display):

- Relative position (x,y) on the X-window display
- Node ID (domainID, nodeIndex)
- Number of segments associated with the node
- Simulation coordinates (x, y, z) of the node

12.2 Gnuplot

ParaDiS is capable of producing output file formatted for use with the gnuplot visualization package. This capability is enabled via the <gnuplot> toggle in the control parameter file. If enabled the code will periodically create a set of gnuplot output file in the directory <outputDir>/gnuplot where <outputDir> is the directory specified by the <dirname> control parameter. The frequency with which the gnuplot files will be created is controlled by the settings of the <gnuplotfreq> and <gnuplotdt> control parameters. The naming convention used for these output files is:

<outputDir>/gnuplot/box.in

<outputDir>/gnuplot/0tNNNN[.SEQ]

where

NNNN

Is a sequence number beginning at ‘0001’ and incremented each time a gnuplot file set is written.

SEQ

Is a sequence number included only when gnuplot files are being written in parallel (i.e. the <numIOGroups> control parameter and number of processors are both greater than 1). <SEQ> will range from zero to <numIOGroups>-1.

In addition, at program termination, an extra set of gnuplot files will be created. This set will be generated regardless of the specified dump frequency in order to provide an image of the final configuration. This set will be named with the slightly different naming convention:

<outputDir>/gnuplot/gnuplot.final[.SEQ]

The ‘box.in’ data file will contain data for plotting the boundaries of the simulation space, while the ‘0t*’ files contain the coordinates pairs defining each unique dislocation segment.

Note: when gnuplot data is being written in parallel (i.e. <numIOGroups> > 1), the dislocation segment data is spread over the file segments. These file segments can be combined into a single file for display in gnuplot using the <ParadisDir>/tools/stitch tool. (See “Tools” section for details on ‘stitch’.)

To aid in visualizing the dislocation configuration, a file containing commands to set some useful gnuplot options has been provided. This file is located at:

<ParadisDir>/inputs/gnuplot.defaults

For example, to view the dislocation structure contained in the gnuplot file 0t0001 you could start gnuplot interactively and execute the commands:

gnuplot> load ‘<ParadisDir>/inputs/gnuplot.defaults’

gnuplot> splot ‘box.in’ with lines, ‘0t0001’ w lines

A second gnuplot command file has been provided as an example of displaying a sequence of gnuplot files. This sample is located at:

<ParadisDir>/inputs/gnuplot.movie

This file has the necessary commands to plot the sequence of files beginning with 0t0001 and ending with 0t1000. After starting gnuplot, simply load this file with:

gnuplot> load '<ParadisDir>/inputs/gnuplot.movie'

This will initialize the plot of the first file, and thereafter simply hitting <Return> will cause gnuplot to move to the next file in the sequence.

For more details on using gnuplot, refer to the gnuplot manual, or enter 'help' from the gnuplot interactive prompt.

12.3 Tecplot

ParaDiS is capable of producing output file formatted for use with the tecplot visualization package. This capability is enabled via the <tecplot> toggle in the control parameter file. If enabled the code will periodically create a set of tecplot output file in the directory <outputDir>/tecplot where <outputDir> is the directory specified by the <dirname> control parameter. The frequency with which the tecplot files will be created is controlled by the settings of the <tecplotfreq> and <tecplotdt> control parameters. The naming convention used for these output files is:

<outputDir>/tecplot/tecdataNNNN[.SEQ]

where

NNNN

Is a sequence number beginning at '0001' and incremented each time a tecplot file set is written.

SEQ

Is a sequence number included only when tecplot files are being written in parallel (i.e. the <numIOGroups> control parameter and number of processors are both greater than 1). <SEQ> will range from zero to <numIOGroups>-1.

In addition, at program termination, an extra set of tecplot files will be created. This set will be generated regardless of the specified dump frequency in order to provide an image of the final configuration. This set will be named with the slightly different naming convention:

<outputDir>/tecplot/tecdata.final[.SEQ]

The generated tecplot output has specific headers to assign variables and time frames (zones) and should look something like this:

```
...

variables = X,Y,Z,V1,V2,V3,V4,V5,V6,V7,V8

zone i = 54  F=POINT

-4000.0 500.0 6000.0 133.1 612.4 -612.3 0.0000 0.0 0.0 1 3

-3866.9 1112.4 5387.7 -133.1 -612.4 -612.3 0.0000 0.0 0.0 1 3

471.0 -5132.6 11632.3 -1712.7 -48.8 48.8 0.0090 -0.0918 0.0918 2 3

...
```

The first line defines the variables to reconstruct the dislocation configuration. X, Y and Z specify the nodal coordinates, and V1, V2 and V3 define arm vectors to the neighboring nodes. These six variables are used to represent the dislocation segments as combinations of points and vectors in tecplot. The V4, V5 and V6 values represent the nodal velocity vector, V7 indicates the number of segments associated with the node, and V8 indicates a burgers vector type.

Note: when tecplot data is being written in parallel (i.e. <numIOGroups> > 1), the dislocation segment data is spread over the file segments. These file segments can be combined into a single file for display in tecplot using the <ParadisDir>/tools/stitch tool. (See the "Tools" section for details on 'stitch'.)

12.4 Povray

If the <povray> toggle is enabled in the control parameter file, ParaDiS will periodically create files containing dislocation segment data for use with the POVRAY (Persistence of Vision™ Ray Tracer) tool. The frequency with which the povray files will be created is controlled by the settings of the <povrayfreq> and <povraydt> control parameters. The naming convention used for these output files is:

<outputDir>/povray/povframeNNNN[.SEQ]

where

NNNN

Is a sequence number beginning at '0001' and incremented each time a povray file set is written.

SEQ

Is a sequence number included only when povray files are being written in parallel (i.e. the <numIOGroups> control parameter and number of processors are both greater than 1). <SEQ> will range from zero to <numIOGroups>-1.

In addition, at program termination, an extra set of povray files will be created. This set will be generated regardless of the specified dump frequency in order to provide an image of the final configuration. This set will be named with the slightly different naming convention:

<outputDir>/povray/povray.final[.SEQ]

Note: these output files contain the main data to be processed by povray, but they must be post-processed via the 'genPovrayFrames' tool (see section on Tools) which will create the final povray input file containing the segment data embedded within a proper framework of povray settings and commands. However, when the fragment data is being written in parallel (i.e. <numIOGroups> > 1) the data is spread over the file segments. These file segments must be combined into a single file for use with povray using the <ParadisDir>/tools/stitch tool. (See the "Tools" section for details on 'stitch'.)

For details on using povray, refer to the povray manual.

12.5 Postscript

If the <psfile> toggle is enabled in the control parameter file, ParaDiS will periodically generate a postscript file containing an image of the current state of the simulation system. The frequency with which the postscript files will be created is controlled by the settings of the <psfilefreq> and <psfiledt> control parameters. The naming convention used for these output files is:

<outputDir>/YshotNNNN.ps

where

NNNN

Is a sequence number beginning with zero and incremented each time another postscript file is written.

12.6 Dislocation Line Fragment Files (for VisIt)

If the <fragfile> toggle is enabled in the control parameter file, ParaDiS will periodically create files containing dislocation line fragment data which can be post-processed for use with the VisIt visualization tool. The frequency with which the fragment files will be created is controlled by the settings of the <fragfreq> and <fragdt> control parameters. The naming convention used for these output files is:

<outputDir>/visit/fragmentsNNNN[.SEQ]

where

NNNN

Is a sequence number beginning at '0001' and incremented each time a fragment file set is written.

SEQ

Is a sequence number included only when fragment files are being written in parallel (i.e. the <numIOGroups> control parameter and number of processors are both greater than 1). <SEQ> will range from zero to <numIOGroups>-1.

In addition, at program termination, an extra set of fragment files will be created. This set will be generated regardless of the specified dump frequency in order to provide an image of the final configuration. This set will be named with the slightly different naming convention:

<outputDir>/visit/fragments.final[.SEQ]

Note: when the fragment data is being written in parallel (i.e. <numIOGroups> > 1), the data is spread over the file segments. These file segments can be combined into a single file for use with VisIt using the <ParadisDir>/tools/stitch tool. (See the "Tools" section for details on 'stitch'.)

See comments in WriteFragments.c for a description of the format of these files.

For details on using VisIt, refer to the VisIt user's manual.

12.7 Node and Segment Files (for VisIt)

If the <writeVisit> toggle is enabled in the control parameter file, ParaDiS will periodically create files containing node and/or segment data formatted for use with the VisIt visualization tool. The <writeVisitNodes> and <writeVisitSegments> toggles are used to select which types of output files are generated. Additionally the <writeVisitNodesAsText> and <writeVisitSegmentsAsText> toggles can be used to selected between text and binary formats for those files (the preferred and default format is the binary format). The frequency with which these files will be created is controlled by the settings of the <writeVisitFreq> and <writeVisitDT> control parameters. The naming convention used for these output files is:

<outputDir>/visit/visitNNNN.meta

<outputDir>/visit/visitNNNN.node[.SEQ]

<outputDir>/visit/visitNNNN.seg[.SEQ]

where

NNNN

Is a sequence number beginning at '0001' and incremented each time a visit file set is written.

SEQ

Is a sequence number included only when visit files are being written in parallel (i.e. the <numIOGroups> control parameter and number of processors are both greater than 1). <SEQ> will range from zero to <numIOGroups>-1.

The ".meta" file is a text file containing information detailing which types of data (node and/or segment) has been written, the format of the data in the node/segment files, and a list of the associated node/segment files.

In addition, at program termination, an extra set of visit files will be created. This set will be generated regardless of the specified dump frequency in order to provide an image of the final configuration. This set will be named with the slightly different naming convention:

<outputDir>/visit/visit.final.meta

<outputDir>/visit/visit.final.node[.SEQ]

<outputDir>/visit/visist.final.seg[.SEQ]

See comments in the WriteVisit.c source file for a description of the format of these files.

13 Output

13.1 Restart Files

Periodic creation of ParaDiS restart files is enabled via the <savecn> toggle in the control file. The frequency with which restart files are written is determined by the settings of the <savecnfreq> and <savecndt> control parameters. When enabled, the periodic restart files will be placed in the directory <outputDir>/restart where <outputDir> is the directory specified by the <dirname> control file parameter.

The nodal data portion of the restart file pair can be written as plain text (the default) or in a binary HDF5 format. Creation of HDF5 files is enabled through the <writeBinRestart> control parameter and setting the HDF_MODE value in <ParadisDir>/makefile.setup to "ON".

Note: use of HDF5 requires the local site to have a version of HDF5 compiled and installed, and the location of the related library and include files specified via the HDF_LIBDIR*, HDF_LIB* and HDF_INC* values in <ParadisDir>/makefile.sys.

The naming convention used for the restart file pairs is:

<outputDir>/restart/rsNNNN # control parameter file

<outputDir>/restart/rsNNNN.data.TYPE[.SEQ] # nodal data file

where

NNNN

Is a sequence number beginning at '0001' and incremented each time a restart file set is written.

SEQ

Is a sequence number included only when tecplot files are being written in parallel (i.e. the <numIOGroups> control parameter and number of processors are both greater than 1). <SEQ> will range from zero to <numIOGroups>-1.

TYPE

Is “data” for regular text restart files or “hdf” for hdf format restart files.

Each time a restart file set is written, the name of the control parameter file will also be written into the file:

<outputDir>/latest_restart

Additionally, when creation of restart files is enabled, an extra restart file pair named ‘restart.cn’ and ‘restart.data’ will be written automatically at program termination even if the cycle is not a multiple of the <savecnfreq> control parameter.

13.2 Property Outputs

13.2.1 Enabling Property Outputs

ParaDiS is capable of producing files containing various properties of the simulation. This capability is enabled via the <saveprop> toggle in the control file. The frequency with which these properties data is written is determined by the settings of the <savepropfreq> and <savepropdt> control parameters.

When enabled, the code will periodically append information to one or more of the properties files specified below. The files will be located in the <outputDir>/properties directory, where <outputDir> is the directory specified by the <dirname> control file parameter.

13.2.2 Density File

This file can be located at:

<outputDir>/properties/density

Each time the code writes out properties data a single line will be appended to this file, where the contents of each line are described in the table below.

Column	Description
1	Plastic strain

Column	Description
2	Strain
3	Dislocation density
4	Deleted dislocation density (lost through annihilation, junction formation, etc)
5	Average dislocation velocity (this will be zero unless the “VEL_STATISTICS” pre-processor macro was defined during compilation)
6	Std. deviation of dislocation velocities (this will be zero unless the “VEL_STATISTICS” pre-processor macro was defined during compilation)
7	Density file version number
The remaining columns of data contain the dislocation density for specific groupings of burgers vectors. These groupings differ in number and content for specific material types.	
BCC	
8	Burgers vectors [1 1 1] [-1 -1 -1]
9	Burgers vectors [-1 1 1] [1 -1 -1]
10	Burgers vectors [1 -1 1] [-1 1 -1]
11	Burgers vectors [1 1 -1] [- 1-1 1]
12	Burgers vectors [1 0 0] [-1 0 0] [0 1 0] [0 -1 0] [0 0 1] [0 0 -1]
FCC	
8	Burgers vectors [1 1 0] [-1 -1 0]
9	Burgers vectors [-1 1 0] [1 -1 0]
10	Burgers vectors [1 0 1] [-1 0 -1]
11	Burgers vectors [-1 0 1] [1 0 -1]
12	Burgers vectors [0 1 1] [0 -1 -1]
13	Burgers vectors [0 -1 1] [0 1 -1]
14	All other burgers vectors

13.2.3 Time/Plastic Strain File

This file can be located at:

<outputDir>/properties/time_Plastic_strain

Each time the code writes out properties data a single line will be appended to this file. Each line will contain two items of data. The first item is the elapsed simulation time and the second is the plastic strain.

13.2.4 Stress/Plastic Strain File

This file will only be generated if the <loadType> control parameter is of type 1 (constant strain rate) or type 4 (cyclic loading). The file will be located at:

<outputDir>/properties/stress_Plastic_strain

Each time the code writes out properties data a single line will be appended to this file. Each line will contain two items of data. The first item is the plastic strain and the second is the stress.

13.2.5 Stress/Total Strain File

This file will only be generated if the <loadType> control parameter is of type 1 (constant strain rate) or type 4 (cyclic loading). The file will be located at:

<outputDir>/properties/stress_Total_strain

The content format of this file is dependent on the loading condition and is described in the table below.

Column	Description
<loadType> = 1 (constant strain rate)	
1	Strain
2	Stress
<loadType> = 4 (cyclic loading)	
1	Net accumulated strain
2	Stress
3	Elapsed simulation time
4	Number of loading cycles

13.2.6 Alleps File

This file can be located at:

<outputDir>/properties/alleps

Each time the code writes out properties data a single line will be appended to this file, where the contents of each line are described in the table below.

Column	Description
1	Simulation timestep number
2	Elapsed simulation time
3 – 8	Plastic strain components (from plastic strain tensor matrix, elements [0][0], [1][1], [2][2], [1][2], [0][1], and [0][2])
9	Dislocation density

13.2.7 Epsdot File

This file can be located at:

<outputDir>/properties/epsdot

Each time the code writes out properties data a single line will be appended to this file. Each line will contain two items of data. The first item is the elapsed simulation time and the second is the plastic strain rate.

13.2.8 Density Delta File

This file contains incremental changes in dislocation density on a per-burgers vector basis with density gains and losses independent of each other. Currently this is only created when one of the BCC mobility modules is in use. The file will be located at:

<outputDir>/properties/density_delta

Each time the code writes out properties data a single line will be appended to this file, where the contents of each line are described in the table below.

Column	Description
1	Strain
2	Sum of columns 4 thru 10 (total incremental gain)
3	Sum of columns 11 thru 17 (total incremental loss)
Density gains	
4	Burgers vectors [-1 1 1] [1 -1 -1]
5	Burgers vectors [1 -1 1] [-1 1 -1]
6	Burgers vectors [1 1 -1] [-1 -1 1]
7	Burgers vectors [1 1 1] [-1 -1 -1]
8	Burgers vectors [1 0 0] [-1 0 0]
9	Burgers vectors [0 1 0] [0 -1 0]
10	Burgers vectors [0 0 1] [0 0 -1]
Density losses	
11	Burgers vectors [-1 1 1] [1 -1 -1]
12	Burgers vectors [1 -1 1] [-1 1 -1]
13	Burgers vectors [1 1 -1] [-1 -1 1]
14	Burgers vectors [1 1 1] [-1 -1 -1]
15	Burgers vectors [1 0 0] [-1 0 0]
16	Burgers vectors [0 1 0] [0 -1 0]
17	Burgers vectors [0 0 1] [0 0 -1]

13.3 Flux Decomposition Files

ParaDiS is capable of producing files containing flux decomposition information for the simulation. The flux is in units of density*velocity or 1/(m*s). This capability is enabled via the <fluxfile> toggle in the control file. The frequency with which the flux data is written is determined by the settings of the <fluxfreq> and <fluxdt> control parameters.

When enabled, the code will periodically append information to the appropriate flux data files. The data files will be located in the <outputDir>/properties directory, where <outputDir> is the directory specified by the <dirname> control file parameter.

The flux decomposition is specific to the material type, and the flux files for the supported types are described below.

13.3.1 BCC Flux Decomposition

The BCC flux decomposition data consists of two sets of files. Each set contains 4 files, one file per burgers vector. The files Ltot_b1 thru Ltot_b4 contain statistics for burgers vector types as indicated below

- Ltot_b1 Burgers vector ½ [1 1 1]
- Ltot_b2 Burgers vector ½ [-1 1 1]
- Ltot_b3 Burgers vector ½ [1 -1 1]
- Ltot_b4 Burgers vector ½ [1 1 -1]

The format of the contents of each of these files is the same, and consists of lines containing data as specified in following table.

Column	Description
1	Plastic strain
2	Strain
3	Screw density
4	Edge density 1
5	Edge density 2
6	Edge density 3
7	Sum of edge densities (columns 4 – 6)
8	Total system edge density (from all Ltot* files)
9	Total system screw density (from all Ltot* files)

The second set of files are fluxtot_b1 through fluxtot_b4, corresponding to the same burgers vectors as specified above for the Ltot_* files. The format for each of these files are similar, however, the flux from edge components on the 3 planes (columns 4 thru 6), and the flux from screw components on the 3 planes (columns 7 thru 9) differ between the files as seen in the table below.

Column	Description

Column	Description
1	Plastic strain
2	Strain
3	Flux due to climb
For burgers vector b1 = ½ [1 1 1]	
4	(0 1 -1), [-2 1 1]
5	(-1 0 1), [1 -2 1]
6	(1 -1 0), [1 1 -2
7	(0 1 -1), [-2 1 1]
8	(-1 0 1), [1 -2 1]
9	(1 -1 0), [1 1 -2
For burgers vector b2 = ½ [-1 1 1]	
4	(0 1 -1). [2 1 1]
5	(1 0 1), [1 2 -1]
6	(0 1 -1), [1 -1 2]
7	(0 1 -1). [2 1 1]
8	(1 0 1), [1 2 -1]
9	(0 1 -1), [1 -1 2]
For burgers vector b3 = ½ [1 -1 1]	
4	(0 1 1), [2 1 -1]
5	(1 0 -1), [1 2 1]
6	(1 1 0), [-1 1 2]
7	(0 1 1), [2 1 -1]
8	(1 0 -1), [1 2 1]
9	(1 1 0), [-1 1 2]
For burgers vector b4 = ½ [1 1 -1]	
4	(0 1 1), [2 -1 1]
5	(1 0 1), [-1 2 1]
6	(1 -1 0), [1 1 2]
7	(0 1 1), [2 -1 1]
8	(1 0 1), [-1 2 1]
9	(1 -1 0), [1 1 2]

13.3.2 FCC Flux Decomposition

The FCC flux decomposition data consists of two sets of files. Each set contains 6 files, one file per burgers vector. The files Ltot_b1 thru Ltot_b6 contain statistics for burgers vector types as indicated below

- Ltot_b1 Burgers vector ½ [1 1 0]
- Ltot_b2 Burgers vector ½ [1 -1 0]
- Ltot_b3 Burgers vector ½ [1 0 1]
- Ltot_b4 Burgers vector ½ [1 0 -1]
- Ltot_b5 Burgers vector ½ [0 1 1]
- Ltot_b6 Burgers vector ½ [0 1 -1]

The format of the contents of each of these files is the same, and consists of lines containing data as specified in following table.

Column	Description
1	Plastic strain
2	Strain
3	Screw density
4	Edge density 1
5	Edge density 2
6	Edge density 3
7	Sum of edge densities (columns 4 – 6)
8	Total system edge density (from all Ltot* files)
9	Total system screw density (from all Ltot* files)

The second set of files are fluxtot_b1 through fluxtot_b6, corresponding to the same burgers vectors as specified above for the Ltot_* files. The format for each of these files are similar, however, the flux from edge components on the 3 planes (columns 4 thru 6), and the flux from screw components on the 3 planes (columns 7 thru 9) differ between the files as seen in the table below.

Column	Description
1	Plastic strain
2	Strain
3	Flux due to climb
For burgers vector b1 = ½ [1 1 0]	
4	(1 -1 1), [-1 1 2]
5	(-1 1 1), [-1 1 -2]
6	(0 0 1), [1 1 0]

Column	Description
7	(1 -1 1), [-1 1 2]
8	(-1 1 1), [-1 1 -2]
9	(0 0 1), [1 1 0]
For burgers vector b2 = ½ [1 -1 0]	
4	(1 1 1), [1 1 -2]
5	(1 1 -1), [-1 -1 -2]
6	(0 0 1), [1 1 0]
7	(1 1 1), [1 1 -2]
8	(1 1 -1), [-1 -1 -2]
9	(0 0 1), [1 1 0]
For burgers vector b3 = ½ [1 0 1]]	
4	(1 1 -1), [1 -2 -1]
5	(-1 1 1), [1 2 -1]
6	(0 1 0), [1 0 -1]
7	(1 1 -1), [1 -2 -1]
8	(-1 1 1), [1 2 -1]
9	(0 1 0), [1 0 -1]
For burgers vector b4 = ½ [1 0 -1]	
4	(1 1 1), [-1 2 -1]
5	(1 -1 1), [1 2 1]
6	(0 1 0), [-1 0 -1]
7	(1 1 1), [-1 2 -1]
8	(1 -1 1), [1 2 1]
9	(0 1 0), [-1 0 -1]
For burgers vector b5 = ½ [0 1 1]	
4	(1 1 -1), [2 -1 1]
5	(1 -1 1), [-2 -1 1]
6	(1 0 0), [0 1 1]
7	(1 1 -1), [2 -1 1]
8	(1 -1 1), [-2 -1 1]
9	(1 0 0), [0 1 1]
For burgers vector b6 = ½ [0 1 -1]	
4	(1 1 1), [-2 1 1]
5	(-1 1 1), [-2 -1 -1]
6	(1 0 0), [0 1 1]
7	(1 1 1), [-2 1 1]
8	(-1 1 1), [-2 -1 -1]

Column	Description
9	(1 0 0), [0 1 1]

13.4 Velocity Files

If the <velfile> toggle is set in the control file, ParaDiS will periodically create a set of files containing velocity information about each unique dislocation node in the simulation. The frequency with which the velocity data is written is determined by the settings of the <velfilefreq> and <velfiledt> control parameters.

The velocity data files will be located in the <outputDir>/velocity directory, where <outputDir> is the directory specified by the <dirname> control file parameter, and will be named with the convention:

<outputDir>/velocity/velNNNN[.SEQ]

where

NNNN

Is a sequence number beginning at '0001' and incremented each time a velocity file set is written.

SEQ

Is a sequence number included only when velocity files are being written in parallel (i.e. the <numIOGroups> control parameter and number of processors are both greater than 1). <SEQ> will range from zero to <numIOGroups>-1.

In addition, at program termination, an extra set of velocity files will be created. This set will be generated regardless of the specified dump frequency in order to provide an image of the final simulation configuration. This set of files will be named with the slightly different naming convention:

<outputDir>/velocity/vel.final[.SEQ]

Note: When velocity data is being written is parallel, the data is spread over multiple file segments. These files can be combined into a single file via the <ParadisDir>/tools/stitch tool. See section on "Tools" for details on 'stitch'.

The contents of the velocity file consist of 5 columns of data for each dislocation node. The first three columns are the velocity components (x, y and z), followed by an integer that is 1 if it is contributing to the strain rate, and -1 if it is moving in the opposite direction. The last column is the node ID in the form "(domainID,nodeIndex)".

13.5 Force Files

If the <writeForce> toggle is set in the control file, ParaDiS will periodically create a set of files containing force information about each unique dislocation node in the simulation. The frequency with which the force data is written is determined by the settings of the <writeForceFreq> and <writeForceDT> control parameters.

The force data files will be located in the <outputDir>/force directory, where <outputDir> is the directory specified by the <dirname> control file parameter, and will be named with the convention:

<outputDir>/force/forceNNNN[.SEQ]

where

NNNN

Is a sequence number beginning at '0001' and incremented each time a force file set is written.

SEQ

Is a sequence number included only when force files are being written in parallel (i.e. the <numIOGroups> control parameter and number of processors are both greater than 1). <SEQ> will range from zero to <numIOGroups>-1.

In addition, at program termination, an extra set of force files will be created. This set will be generated regardless of the specified dump frequency in order to provide an image of the final simulation configuration. This set of files will be named with the slightly different naming convention:

<outputDir>/force/force.final[.SEQ]

Note: When force data is being written is parallel, the data is spread over multiple file segments. These files can be combined into a single file via the <ParadisDir>/tools/stitch tool. See section on "Tools" for details on 'stitch'.

The contents of the force file consist of 4 columns of data for each dislocation node. The first three columns are the force components (x, y and z), and the final column is the node ID in the form "(domainID,nodeIndex)".

13.6 Segment Files

If the <armfile> toggle is set in the control file, ParaDiS will periodically create a set of files containing information about each unique dislocation segment in the simulation. The frequency with which the segment data is written is determined by the settings of the <armfilefreq> and <armfiledt> control parameters.

The segment data files will be located in the <outputDir>/armdata directory, where <outputDir> is the directory specified by the <dirname> control file parameter, and will be named with the convention:

<outputDir>/armdata/armNNNN[.SEQ]

where

NNNN

Is a sequence number beginning at '0001' and incremented each time a segment file set is written.

SEQ

Is a sequence number included only when segment files are being written in parallel (i.e. the <numIOGroups> control parameter and number of processors are both greater than 1). <SEQ> will range from zero to <numIOGroups>-1.

In addition, at program termination, an extra set of segment files will be created. This set will be generated regardless of the specified dump frequency in order to provide an image of the final simulation configuration. This set of files will be named with the slightly different naming convention:

<outputDir>/armdata/arm.final[.SEQ]

Note: When segment data is being written is parallel, the data is spread over multiple file segments. These files can be combined into a single file via the <ParadisDir>/tools/stitch tool. See section on "Tools" for details on 'stitch'.

The contents of the segment files consist of 10 columns of data (1 line per segment) as described below:

Column	Description
1 thru 3	Burgers vector components (x, y, z)
4 thru 6	Line direction vector (x, y, z)
7	Segment length (units of b)
8 thru 10	Coordinates of the node owning the segment (x, y, z)

13.7 Density Field File

If all components of the <savedensityspec> control parameter are set to positive values, ParaDiS will (at program termination) create a file containing a 3D dislocation density field file formatted for use with the VASP Data Viewer (vaspview). The three components of the <savedensityspec> parameter specify the granularity of the density field in the X, Y and Z dimensions respectively. The density field will be written to the file:

<outputDir>/densityfield.out

Warning: this will overwrite any existing density field file of the same name!

The VASP Data Viewer is publicly available on the web for non-commercial use. For details on the product, download the VASP Data Viewer from the web and refer to the accompanying documentation.

14 Utilities

14.1 Creating Initial Dislocations with Paradisgen

The ‘paradisgen’ utility is designed to generate a set of initial dislocations suitable for a ParaDiS simulation. The command line options for paradisgen control the type of dislocations, size of the simulation box, and so on. Execute ‘paradisgen –help’ for a list and brief description of the available command line options to the utility.

14.2 Recomputing Domain Boundaries with Paradisrepart

The ‘paradisrepart’ utility provides a mechanism by which to replace the domain decomposition/partitioning in an existing nodal data file with a new domain decomposition. The utility will read the nodal data, domain and cell geometry information from a restart file and attempt to repartition the domains such that the computational cost for each domain will be roughly equivalent.

This utility is primarily used when it is necessary to alter the domain geometry or domain count for a simulation in order to continue. In such a situation, the existing domain decomposition would be thrown away and the ParaDiS code would generate an initial uniform domain decomposition and then over time converge on a more optimal decomposition. Using this utility instead to generate a new decomposition provides a more reasonable starting point, allowing the ParaDiS simulation to converge on an optimal decomposition much more quickly.

Note: The addition of the Recursive Bisectioning decomposition method makes this utility fairly obsolete because that decomposition method is capable of converging on an optimal decomposition much more quickly than the Recursive Sectioning algorithm that existed when this utility was first developed.

Execute ‘paradisrepart –help’ a list and brief description of the available command line options to the utility.

14.3 Converting Restart Files with Paradisconvert

The ‘paradisconvert’ utility provides a mechanism for converting older no longer supported format ParaDiS control and data files (and restart files) to the current file formats. This utility does recognize and handle segmented data files.

In most cases this utility will not be required since ParaDiS is still able to recognize and handle most of the older file formats. The only format that is currently no longer supported is the truly ancient format associated with the earliest incarnations of the code in which the control file parameters and nodal data were combined in a single file. For those restart files, paradisconvert will be needed.

The command line format for paradisconvert is:

paradisconvert <controlFile> [dataFile]

where

controlFile

specifies the name of the control parameter file to be updated to the current format.

dataFile

if the specified control file is the original ancient format combining control parameters and nodal data in the same file, this argument is ignored. Otherwise, this specifies the base name of the nodal data file(s) to be converted. If this argument is not provided, it will default to the same name as <controlFile> with any file name suffix removed and replaced with the “.data” suffix.

On success, the utility will rename the original files by appending a “.bkup” suffix to the names, and create new control and data files with the specified names. Note: for control parameter files which include nodal data, a new data file will be generated under the same name as <controlFile> with any file name suffix removed and replaced with the “.data” suffix.

14.3.1 Examples

To convert the original format file ‘restart.cn’ containing both control parameters and nodal data, execute:

paradisconvert restart.cn

The utility will generate the following output files:

restart.cn.bkup

restart.cn

restart.data

To convert a newer control parameter and data file pair with the names ‘rs0100’ and ‘rs0100.data’ respectively, execute either of the equivalent command lines:

paradisconvert rs0100

paradisconvert rs0100 rs0100.data

In both cases the utility will generate the output files:

rs0100.bkup

rs0100.data.bkup

and

rs0100

rs0100.data

To convert a new control file ‘rs0100’ and a set of segmented data files with the names ‘rs0100.data.0’, ‘rs0100.data.1’,… ‘rs0100.data.7’, execute any of the following equivalent commands:

paradisconvert rs0100

paradisconvert rs0100 rs0100.data

paradisconvert rs0100 rs0100.data.0

In all of the above cases, the following output files will be generated:

rs0100.bkup

rs0100.data.0.bkup

rs0100.data.1.bkup

…

rs0100.data.7.bkup

and

rs0100

rs0100.data.0

rs0100.data.1

…

rs0100.data.7

14.4 Creating an FMM Image Correction Table with Ctablegen

The ‘ctablegen’ utility is used to create an image correction table needed when the FMM (Fast Multipole method) has been enabled by setting the <fmEnabled> toggle in the control parameter file. Since the data in this table is dependent on the poisson ratio, shear modulus and the orders for the multipole and taylor expansions, the file must be created for the particular values of these items used in the simulation.

Given the current defaults used by ParaDiS (as set in Param.c):

poisson ratio	3.327533e-01
shear modulus	6.488424e+10
multipole order	2
taylor expansion order	5

To create the FMM image correction table, you could execute ctablegen from the main <ParadisDir> directory using the command line:

bin/ctablegen –nu 3.327533e-01 –mu 6.488424e+10 –mporder 2 –torder 5 –outfile inputs/fm-ctab.Ta.600K.0GPa.m2.t5.data

Note: the generation of this file can take a significant amount of time, therefore a parallel version of the utility (called ctablegenp) has also been provided. To execute the table generator in parallel on 8 processors using mpirun ad a parallel program initiator, one could execute:

mpirun –np 8 bin/ctablegen –nu 3.327533e-01 –mu 6.488424e+10 –mporder 2 –torder 5 –outfile inputs/fm-ctab.Ta.600K.0GPa.m2.t5.data

By default, the image correction table is created assuming periodic boundary conditions are imposed on the simulation space in all dimensions. However, a

mixture of periodic boundaries and free surfaces can be selected via the '-pbc' command line option to the utility. Execute 'ctablegen –help' for details.

14.5 Creating Far-Field Stress Tables with Stresstablegen

The 'stresstablegen' utility is used to create the tables needed for calculating stress from distant cells and periodic images of the system when the FMM is disabled (i.e. <fmEnabled> is zero). There are two required tables, one which factors in stress from periodic images only, and the other which factors in stress from both the primary and periodic images.

As an example, you could create the needed stress tables by executing the stresstablegen utility twice from the main <ParadisDir> directory using the following command lines:

bin/stresstablegen –nopbc –outfile inputs/Rijm.cube.out

and

bin/stresstablegen –pbc –outfile inputs/RijmPBC.cube.out

At run time, the locations of these files are specified by the <Rijmfile> and <RijmPBCfile> control parameters.

For details on using the stresstablegen utility, execute 'bin/stresstablegen –help'

14.6 Generating Density Fields via Calcdensity

The 'calcdensity' utility is designed to read the nodal data information from a restart file and calculate the corresponding dislocation density at the center of a uniform set of cells where the number of cells per dimension is specified by the user. This density grid is then written to a file which can be visualized via an external utility.

The output file created by this utility contains data lines of the format:

<xCoord> <yCoord> <zCoord> <density>

Where the <*Coord> values on each line correspond to the coordinates of the center of a density grid cell, and the <density> value is the total portion of the simulation's dislocation density contained within that cell. Anything followed by a

‘#’ on any line in the file is considered to be a comment, and blank lines are considered white space.

As an example, if you had a restart file pair called ‘rs0100’ and ‘rs0100.data’ you could create a 30 X 30 X 30 density grid file ‘rs0100.dens’ with any of the following commands:

calcdensity –g 30 rs0100

calcdensity –g 30 –d rs0100.data rs0100

calcdensity –g 30 –d rs0100.data rs0100 –o rs0100.dens

If you want to specify a non-uniform density grid of dimensions 20 X 30 X 40, you could use:

calcdensity –g 20,30,40 rs0100

For full details on the available command line options for this utility, execute ‘calcdensity –h’.

15 Tools

15.1 genPovrayFrames

The genPovrayFrames tool is provided to post-process the povray data generated by ParaDiS when the <povray> control parameter toggle has been set. This tool will create an '*.pov' file containing the ParaDiS generated data embedded in a proper framework of povray settings and commands. The tool is located in the <ParadisDir>/tools directory. For details on the use of this tool, see the comments at the beginning of the script or execute

<ParadisDir>/tools/genPovrayFrames -help

15.2 gnuplot2povray

The 'gnuplot2povray' tool provides a mechanism by which to convert existing gnuplot files created via ParaDiS into a '*.pov' file containing the converted data embedded in a proper framework of povray settings and commands. The tool is located in the <ParadisDir>/tools directory. For details on the use of this tool, see the comments at the beginning of the script or execute

<ParadisDir>/tools/gnuplot2povray -help

15.3 stitch

When ParaDiS is configured to enable parallel I/O (i.e. the <numIOGroups> control parameter and number of processors are greater than 1), each of the following types of output files will be generated as a set of files rather than a single output file.

- Gnuplot files
- Tecplot files
- Segment/arm data files
- Povray files
- Velocity data files
- Pole figure files

- Force data files

Each of the file ‘segments’ will contain a portion of the full data and will have a sequence number appended to the file name.

Most utilities for processing these types of output files, however, expect the data in a single file, so the ‘stitch’ tool has been provided in order to recombine these data file segments into a single usable file as a post-processing step. The tool can be found in the source code release as:

<ParadisDir>/tools/stitch

The command line for stitch is as follows:

stitch [-h] [-d dir | -f file]

where

-h

Prints the usage information to stdout.

-d <dir>

Specifies a directory the utility will scan for segmented output files that need to be ‘stitched’ together.

-f <file>

Specifies a base file name. The tool will scan for the corresponding file segments (files named <file>.N where N ranges from zero on up) and stitch the segments into a single file with the base file name.

Note: if neither a directory or file name are provided on the command line, the tool will behave as though the user specified the current directory via the –d option and will perform as stated above.

16 Appendix

16.1 Control File Parameters

The following table contains a list of the valid ParaDiS control file parameters along with a brief description of each. The parameters have been grouped into the following categories, and unless otherwise specified, units are in SI with lengths normalized by burgers vector magnitude:

- Simulation cell an processor setup
- Simulation time and timestepping controls
- Discretization controls
- FMM controls
- Tables for non-FMM far-field forces
- Loading condition parameters
- Material and mobility parameters
- Velocity statistics and controls
- I/O controls and parameters
- Miscellaneous parameters

Simulation cell and processor setup		
numXdoms, numYdoms, numZdoms	Integer	Defines the number of computational domains into which the problem space is partitioned in the corresponding dimensions.
numXcells, numYcells, numZcells	Integer	Defines the number of cells in the corresponding dimension of the problem space. Cells are independent of the domain geometry and are used to determine boundaries at which far-field forces are computed rather than direct segment to segment forces.

taskMappingMode	Integer	Defines the behavior when the user-supplied spatial decomposition does not match the physical 3D geometry of the hardware partition on which the simulation is executing. This is only valid on BG/P type systems. Valid values are: 0 – domain decomposition will be reset to be consistent with the hardware geometry if possible 1 – use the user-supplied domain decomposition. This is the default 2- abort if the domain decomposition does not directly map onto the physical hardware partition
xBoundType, yBoundType, zBoundType	Integer	Defines the type of problem space boundaries in the corresponding dimension. Currently supported types are 0 and 1 for periodic and free surfaces respectively.
xBoundMax, yBoundMax, zBoundMax	Double	If periodic boundaries are not enabled, defines the upper limit on coordinates of any dislocation nodes in the corresponding dimension. Value must be <= the respective maximum problem space coordinate specified in the nodal data file.
xBoundMin, yBoundMin, zBoundMin	Double	If periodic boundaries are not enabled, defines the lower limit on coordinates of any dislocation nodes in the corresponding dimension. Value must be >= the respective minimum problem space coordinate specified in the nodal data file.
decompType	Integer	Specifies the type of domain decomposition to be used. A value of 1 selects the Recursive Sectioning (RS) algorithm, and a value of 2 selects the Recursive Bisectioning (RB) decomposition algorithm. The default is to use the RB decomposition.
DLBFreq	Integer	Indicates the frequency (in cycles) at which the Dynamic Load-Balancing is to be attempted. A value of zero turns off load-balancing. The default is 3.
Simulation time and timestepping controls		

cycleStart	Integer	Starting cycle number for the simulation.
maxStep	Integer	Indicates the number of timesteps to execute before terminating.
timeNow	Double	Current simulation time (in seconds).
timeStart	Double	Initial simulation time (in seconds).
timestepIntegrator	String	Selects a timestep integration method. Valid methods are: "trapezoid" "forward-euler" The default is "trapezoid.
deltaTT	Double	Indicates the duration of the previous timestep in units of seconds.
maxDT	Double	Specifies the maximum timestep duration permitted. Default is 1.0e-07.
nextDT	Double	Specifies the timestep duration to attempt on the next simulation cycle. The timestep integrator will adjust this value dynamically. The default value is <maxDT>.
dtIncrementFactor	Double	Maximum factor by which <deltaTT> is multiplied when increasing the timestep duration. Must be at least 1.0. Default is 1.2
dtvariableAdjustment	Integer	Toggles ability to vary the increment by which the timestep is adjusted when the current timestep is determined to be too small. This permits the timestep to be adjusted to a value between 1.0 and <dtIncrementFactor> * <delatTT>. Default is zero. Used only with the "trapezoid" timestep integrator.
rTol	Double	Maximum position error (in units of b) tolerated in the timestep integration. Only applies to the "trapezoid" timestep integrator. Default is 0.25 * <rc>.
rmax	Double	Maximum distance (in units of b) a node is permitted to move in a single timestep. Only applies to the "forward-euler" timestep integrator. Default is 0.5 * <minSeg>.
Discretization controls		

maxSeg	Double	Sets the maximum permitted length (in units of b) of a dislocation segment. Primarily used for determining when segments are to be rediscretized during remesh operations. This value must be < 9/10 the cell size of a cell. There is no default value and must be specified in the control file.
minSeg	double	Sets the minimum permitted length (in units of b) of a discretization segment. Primarily used for determining when nodes are to be removed during remesh operations. Default is sqrt(remeshAreaMin*(4/sqrt(3)) where remeshAreaMin = 2*rTol*maxSeg.
remeshRule	Integer	Indicates the version of remesh rules governing the rediscretization of dislocations. Currently supported versions are 2 and 3. (See section on Rediscretization for details on remesh versions). Default value is 2.
splitMultiNodeFreq	Integer	Indicates the frequency with which the code attempts to split multi-nodes (or surface nodes). Splits will be attempted each cycle that is a multiple of this value. Default and minimum values are 1.
collisionMethod	Integer	Selects the method used for determining if dislocation segments should collide. A value of 1 selects the original method based purely on proximity criteria. A value of 2 selects a new predictive method which determines if and when segments will intersect and uses this data as collision criteria instead. The default is 2.
enforceGlidePlanes	Integer	Toggle enabling code to restrict dislocation motion to the segments' glide planes, and to alter behavior of remesh operations for non-discretization nodes. Default is 0 (off) however the toggle will be force on if the selected <mobilityLaw> constrains motion to glide planes.
FMM controls		

fmEnabled	Integer	Toggle controlling use of a fast Multipole Method (FMM) for computing force contributions from remote dislocation segments. Any value other than zero enables FMM. Default is zero.
fmCorrectionTbl	String	Name of the image correction table used for the FMM. This table must correspond to the specified <fmMPOrder>, <fmtaylorOrder>, and <pois> control parameters. See the 'ctablegen' utility for information on creating this table. This string is ignored if <fmEnabled> is zero.
fmMPOrder	Integer	Defines the multipole expansion order used by the FMM. This value is ignored if <fmEnabled> is zero. Default value is 2.
fmTaylorOrder	Integer	Defines the order of the taylor expansions used by the FMM. This value is ignored if <fmEnabled> is zero. Default value is 5.
Tables for non-FMM far-field forces		
Rijmfile	String	Name of the file containing the RIJM table to be used for far-field stress calculations. This parameter is ignored if the <fmEnabled> parameter is non-zero. Default is "inputs/Rijm.cube.out".
RijmPBCfile	String	Name of the file containing the RIJM table to be used for far-field stress calculations with periodic boundary conditions. This parameter is ignored if the <fmEnabled> parameter is non-zero. Default is "inputs/RijmPBC.cube.out".
Loading condition parameters		
TempK	Double	Simulation temperature (in degrees Kelvin)

loadType	Integer	Defines the type of load applied to the system. Valid types are: 0: Creep test 1: Constant strain rate 2: Jump test for bulk simulation 3: Jump test for junction unzipping 4: Cyclic loading condition 5: Conditional cyclic loading Note: types 4 and 5 are not yet fully supported.
appliedStress	Double[6]	Initial external stress specified in units of Pa as [sigma11, sigma22, sigma33, sigma23, sigma31, sigma12]. Default is all components zero. This quantity is adjusted by a feedback mechanism through plastic strain increment in strain-rate controlled simulations.
eRate	Double	Strain rate applied in strain-rate controlled simulations. Default is 1.0/sec. If <loadType> is zero, this will explicitly be set to 1.0/sec for output consistency.
indxErate	Integer	Index to indicate normal or shear deformation. Used only for stress increment in strain-rate controlled simulations to update associated stress components. Valid values are: 1: normal (default) 2: shear
edotdir	Double[3]	Vector specifying an arbitrary uniaxial loading direction. Example could be a [1 2 5] direction. Used for strain-rate controlled simulations. Its unit vector is also used to rotate quantities from [100]-[010]-[001] coordinate frames to the user specified loading direction. Default is [1 0 0].
cTimeOld	Integer	Timestep related to cyclic loading. Only used for cyclic load conditions (i.e. <loadType> is 4 or 5).
dCyclicStrain	Double	Incremental strain under cyclic load. Only used for cyclic load conditions (i.e. <loadType> is 4 or 5).
netCyclicStrain	Double	Net accumulated strain under cyclic load. Only used for cyclic loading conditions(i.e. <loadType> is 4 or 5).
numLoadCycle	Double	Number of cyclic cycles. Only used for cyclic loading conditions(i.e. <loadType> is 4 or 5).

eAmp	Double	Strain amplitude. Only used for cyclic loading conditions(i.e. <loadType> is 4 or 5).
uselabFrame	Integer	Toggle indicating if the standard crystallographic frame or a user-defined laboratory frame is used. (See <labFrameXDir> below) Default is zero indicating use of crystallographic frame.
labFrameXDir, labFrameYDir, labFrameZDir	Double[3]	These three vectors specify the axes for a user-defined laboratory frame rather than the crystallographic frame. Note: the Z direction is informational only and will be computed explicitly from the X and Y directions. These values apply only if the <useLabFrame> toggle is set. Defaults: XDir [1 0 0] YDir [0 1 0] ZDir [0 0 1]
Material and mobility parameters ** Note: Default values for all material and mobility related parameters correspond to Tantalum at a temperature of 600(K) and a pressure of 0GPa.		
mobilityLaw	String	Specifies by name the set of rules governing dislocation motion for the simulation. Default is "BCC_0". (See section on Dislocation Mobility for available mobility law selections)
enableCrossSlip	Integer	Toggle to enable/disable the use of a dislocation cross-slip mechanism. Primarily for use with mobility functions that restrict dislocation motion to assigned glide planes. Default is zero (off).
vacancyConc	Double	Number density of vacancies per atomic volume. This value is only used when calculating osmotic forces, and must be used in conjunction with the <vacancyConcEquilibrium> parameter. Setting both this parameter and <vacancyConcEquilibrium> enables calculation of osmotic forces. No default.

vacancyConcEquilibrium	Double	Number density of vacancies in thermal equilibrium per atomic volume. This value is only used when calculating osmotic forces, and must be used in conjunction with the <vacancyConc> parameter. Setting both this parameter and <vacancyConc> enables calculation of osmotic forces. No default.
shearModulus	Double	Shear modulus (in Pa). Default is 6.488424e+10.
pois	Double	Poisson ratio. Default is 3.327533e-01.
burgMag	Double	Magnitude of the burgers vector (b) in units of meters. Default is 2.875401e-10.
YoungsModulus	Double	Youngs modulus (units of Pa). This value will be explicitly calculated in the code from E = 2G(1+pois). As such it is only included here for informational purposes.
rc	Double	Core radius (units of b) for self-force calculations. No default; must be supplied.
eCore	Double	Core energy used for self-force calculations. Default is (<shearModulus>/(4*PI))*log(<rc>/0.1) .
MobScrew	Double	Mobility of screw dislocations in units of 1/(Pa*sec). Default is 10.0. Not applicable to all mobility functions.
MobEdge	Double	Mobility of edge dislocations in units of (1/(Pa*sec). Default is 10.0. Not applicable to all mobility functions.
MobClimb	Double	Climb mobility of dislocations in units of 1/(Pa*sec). Default is 1.0e-02. Not applicable to all mobility functions.
sessileburgspec	Double[30	Array of burgers vectors to be considered sessile. First element of the array contains the number of burgers vectors specified, the remaining elements specify the X, Y and Z components of the sessile burgers vector. Maximum sessile burgers vectors allowed is 9. No burgers vectors are sessile by default.

sessilelinespec	Double[30]	Array of line directions related to <sessileburgspec>. The first element is ignored, remaining elements specify the X, Y and Z components of each sessile line. The number of sessile lines is assumed to be the same as the number of sessile burgers vectors.
includeInertia	Integer	Toggle to enable/disable inertial terms (if available) in the mobility functions. Default is zero (off).
massDensity	Double	Mass of material in units of kg/m^3. Only used if <includeInertia> flag is enabled. No default provided. Vanadium = 5800 Molybdenum = 10220 Tantalum = 16650
Flux decomposition		
totpSpn	Double[6]	Plastic strain tensor
totpStn	Double[6]	Plastic spin tensor
totstraintensor	Double[6]	Strain rate tensor with respect to global coordinate system.
Ltot	Double[4][4]	Decomposed density per burgers vector for screw and three edges (for BCC slip systems only).
FCC_Ltot	Double[6][4]	Decomposed density per burgers vector for screw and three edges (for FCC slip systems only).
fluxtot	Double[4][7]	For each burgers vector, contains: 1: flux due to climb 2-4: flux due to edge components 5-7: flux due to screw components (for BCC slip systems only).
FCC_fluxtot	Double[6][7]	For each burgers vector, contains: 1: flux due to climb 2-4: flux due to edge components 5-7: flux due to screw components (for FCC slip systems only).
Velocity statistics ** Note: these statistics will only be used if the VEL_STATISTICS pre-processor macro has been defined during compilation.		
vAverage	Double	Average dislocation velocity.
vStDev	Double	Standard deviation of dislocation velocities.

I/O controls and parameters

** Note: A number of the supported output forms are controlled by very similar control parameters. The general descriptions below apply to all I/O control parameters of like name.

*freq Sets the frequency (in cycles) at which the associated data will be written to disk. If the corresponding <*dt> parameter is greater than zero, this parameter will be ignored. Default for all such values is 100.

*dt Specifies the simulation delta time that will control the frequency at which the associated output will be written to disk. A positive value is interpreted as a delta time and will take precedence over any frequency specified by the corresponding <*freq> value. A value <= zero indicates the write frequency will not be determined by delta times. Default for all such parameters is -1.0

*time Specifies the simulation time at which the associated data was last written to disk. These values will be automatically updated during the simulation. If the corresponding <*dt> parameter is <= 0, this parameter will be ignored.

*counter Sequence number of the previously written file of the corresponding type. Default for all such parameters is 0.

dirname	String	Base output directory name
skipIO	Integer	Toggle for disabling generation of all output types other than timing files. Overrides output-specific toggles if set. Default is zero.
writeBinRestart	Integer	Toggle enabling disabling write of the nodal data portion of the restart file as an HDF5 file. Requires HDF_MODE in 'makefile.setup' to be set to "ON". Default is zero (off).
numIOGroups	integer	Sets the number of groups into which the domains will be separated for doing parallel I/O. All files generated in parallel will be created with this number of segments. This value must be at least 1 and no more than the total number of processing domains used. Default is 1.
armfile	integer	Toggle enabling/disabling generation of files identifying each unique dislocation segment. Default is zero (off).
armfilecounter	Integer	See description of <*counter> above.
armfiledt	Double	See description of <*dt> above.

armfilefreq	Integer	See description of <*freq> above.
armfiletime	double	See description of <*time> above.
fluxfile	Integer	Toggle enabling/disabling generation of flux decomposition files. Default is zero (off).
fluxcounter	Integer	See description of <*counter> above.
fluxdt	Double	See description of <*dt> above.
fluxfreq	Integer	See description of <*freq> above.
fluxtime	Double	See description of <*time> above.
fragfile	Integer	Toggle enabling/disabling generation of dislocation line fragment files for use with the VisIt visualization tool. Default is zero (off).
fragcounter	Integer	See description of <*counter> above.
fragdt	Double	See description of <*dt> above.
fragfreq	Integer	See description of <*freq> above.
fragtime	Double	See description of <*time> above.
gnuplot	Integer	Toggle enabling/disabling generation of files formatted for use with gnuplot. Default is zero (off).
gnuplotcounter	Integer	See description of <*counter> above.
gnuplotdt	Double	See description of <*dt> above.
gnuplotfreq	Integer	See description of <*freq> above.
gnuplottime	Double	See description of <*time> above.
polefigfile	Integer	Toggle enabling/disabling generation of <111> type burgers vector pole figures. Default is zero (off).
polefilecounter	Integer	See description of <*counter> above.
polefigdt	Double	See description of <*dt> above.
polefigfreq	Integer	See description of <*freq> above.
polefigtime	Double	See description of <*time> above.
povray	Integer	Toggle enabling/disabling generation of files with nodal data and domain boundaries for use with the povray image generator. Default is zero (off).

povraycounter	Integer	See description of <*counter> above.
povraydt	Double	See description of <*dt> above.
povrayfreq	Integer	See description of <*freq> above.
povraytime	Double	See description of <*time> above.
psfile	Integer	Toggle enabling/disabling generation of postscript files containing nodal data and domain boundaries. Default is zero (off).
psfiledt	Double	See description of <*dt> above.
psfilefreq	Integer	See description of <*freq> above.
psfiletime	Double	See description of <*time> above.
savecn	Integer	Toggle enabling/disabling writing of restart (control parameter and nodal data) files. Default is zero (off).
savecncounter	Integer	See description of <*counter> above.
savecndt	Double	See description of <*dt> above.
savecnfreq	Integer	See description of <*freq> above.
savecntime	Double	See description of <*time> above.
saveprop	Integer	Toggle enabling/disabling writing of various properties files. Default is zero (off).
savepropdt	Double	See description of <*dt> above.
savepropfreq	Integer	See description of <*freq> above.
savepproptime	Double	See description of <*time> above.
savetimers	Integer	Toggle enabling/disabling generation of the coarse-grain timing data files. Default is zero (off).
savetimerscounter	Integer	See description of <*counter> above.
savetimersdt	Double	See description of <*dt> above.
savetimersfreq	Integer	See description of <*freq> above.
savetimerstime	Double	See description of <*time> above.

savedensityspec	Integer[3]	Specifies the granularity of the 3D density field written to the density filed file in the X, Y and Z dimensions. If any element of this array is zero, the capability is disabled. Default is all zeros (off).
tecplot	Integer	Toggle enabling/disabling generation of output files formatted for use with tecplot. Default is zero (off).
tecplotcounter	Integer	See description of <*counter> above.
tecplotdt	Double	See description of <*dt> above.
tecplotfreq	Integer	See description of <*freq> above.
tecplottime	Double	See description of <*time> above.
velfile	Integer	Toggle enabling/disabling generation of output files containing velocity data for all nodes in the simulation. Default is zero (off).
velfilecounter	Integer	See description of <*counter> above.
velfiledt	Double	See description of <*dt> above.
velfilefreq	Integer	See description of <*freq> above.
velfiletime	Double	See description of <*time> above.
winDefaultsFile	String	Name of a file containing default options and attributes for the X-window display. This is ignored if the X display support was not enabled at compile time. Default is "inputs/paradis.xdefaults".
writeVisit	Integer	Toggle enabling/disabling generation of VisIt output files containing node and/or segment data. Default is zero (off). If enabled, one of <writeVisitNodes> or <writeVisitSegments> must also be enabled/
writeVisitCounter	Integer	See description of <*counter> above.
writeVisitDT	Double	See description of <*dt> above.
writeVisitFreq	Integer	See description of <*freq> above.
writeVisitTime	Double	See description of <*time> above.
writeVisitNodes	Integer	Enables/disables writing of VisIt node data files. Default is 0 (off).

writeVisitSegments	Integer	Enables/disables writing of VisIt segment data files. Default is zero (off).
writeVisitNodesAsText	Integer	Toggle selecting whether VisIt node data files are written in a text or binary format. Default is zero (binary format). This value is ignored if <writeVisitNodes> is not enabled.
writeVisitSegmentsAsText	Integer	Toggle selecting whether VisIt segment data files are written in a text or binary format. Default is zero (binary format). This value is ignored if <writeVisitSegments> is not enabled.
Other parameters		
elasticinteraction	Integer	Toggles between explicit calculation of elastic interaction and a simple line tension calculation. Default is 1 (on).
TensionFactor	Double	Factor used for simple line tension force calculations when <elasticinteraction> is zero. Default is 1.0