LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-SM-853563

ParaDiS User's Guide, Version 4

V. V. Bulatov, N. R. Bertin, S. Aubry, A. Arsenlis, W. Cai

August 23, 2023

Disclaimer

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

8/22/2023

LAWRENCE LIVERMORE NATIONAL LABORATORY

PARADIS USER'S GUIDE VERSION 4.0

ParaDIS

We create hairballs even a cat
wouldn't cough up

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>/materials/

Sample files containing material specific tables and constants

<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/gen_fmm_derivatives

This utility is used to create a file containing the Green's function derivatives needed in conjunction with anisotropic FMM. (See Utilities section for more information)

<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.

<ParadisDir>/bin/inclusiongen

Generates specifications for Eshelby inclusions in a format suitable for use with ParaDiS.

2.2.1 Enabling Multi-threading Support

(NOTE: The multi-threading of ParaDiS is currently still under development. There are portions of the code that have not yet been threaded, however, all portions that have been threaded are safe to use.)

The default execution mode for ParaDiS simulations is to exploit parallelism at the MPI task level utilizing a single processor per task. The ParaDiS code does support a mode in which a degree of parallelism is attained within each MPI task via multiple threads of execution via OpenMP *pragmas* inserted into the code.

The determination of whether to use the OpenMP modifications is done at compile time, and is controlled via the definition of "OPENMP_MODE" in the file 'makefile.setup'. By default, this value is not defined, disabling multi-threading support. To enable multi-threading, simply set "OPENMP_MODE = ON" before compiling.

Enabling OpenMP requires certain compiler flags to be set. The current ParaDiS 'makefile.sys' supports OpenMP for a number of the pre-defined system types. For other system types, 'makefile.sys' may have to be modified to add any necessary OpenMP compiler flags required by compilers at the user's site.

When OpenMP support has been enabled and compiled into the executable, the ParaDiS simulations will default to using 1 thread per MPI task. This default can be overridden via the ParaDiS command line option "-n <numThreads>" where <numThreads> defines the maximum number of threads per task.

2.2.2 Shutdown Signal Handling

The ParaDiS code provides an optional capability for the user to define a 'shutdown' signal. When this capability is enabled, the code will register to catch the specified signal. Upon receipt of the signal, the simulation attempts to complete the current simulation cycle then enter the standard termination processing during which any final output files are written, followed by a clean shutdown of the simulation.

This capability is enabled/disabled at compile time and is controlled by a flag in <ParadisDir>/makefile.setup. By default this capability is disabled but can be enabled by uncommenting the following definition in the make file:

```
DEFS += -D_SHUTDOWN_SIGNAL=SIGTERM
```

When enabled, the default shutdown signal is "SIGTERM", however, this signal can be replaced with any signal supported by the operating system.

There are various situations in which this capability may be useful.

- Some batch systems are able to send an 'out-of-time' signal to jobs shortly before the jobs exceed their time limits and are terminated. This capability allows the code to catch such a signal and may allow ParaDiS time to generate a final restart file before its time is exhausted so that progress since the previous restart file was written is not lost.
- At times, you may wish to temporarily stop a simulation without losing all progress made since the last restart file was written. This capability provides a way for you to do so, assuming a means of sending a signal to the job exists.
- On some systems, receipt of an un-trapped signal in a threaded application can result in core dumps. Defining the signal as a shutdown signal allows it to be caught and handled cleanly without generating unwanted core files.

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 an MPI task. This option applies only if the executable has been compiled with OpenMP support. (See section on Compiling with Multi-threading Support). If this command line option is not used, default behavior is to run with 1 thread per MPI task.

-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

If, for example, you were using a computer which used mpirun for parallel job initiation and you had a control parameter file 'testrun.ctrl' with 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.ctrl

mpirun -np 8 bin/paradis -d testrun2.data testrun.ctrl
```

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. Each of the sample problems requires a table for calculating the far-field forces from remote dislocations. The file containing the appropriate table is specified by the <fmCorrectionTbl> parameter in the sample control parameter file. This tbale *ma* have been provided with the source 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/form_binaryjunc.ctrl

<ParadisDir>/tests/form_binaryjunc.data
```

<ParadisDir>/tests/frank_read_src.ctrl

<ParadisDir>/tests/frank_read_src.data

<ParadisDir>/tests/Tantalum.ctrl

<ParadisDir>/tests/Tantalum.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 ‘Tantalum*’ files are just a general demonstration of the behavior of a small number of screw dislocations in BCC tantalum. This is an 8 processor simulation which can be executed by:

```
mpirun -np 8 ./bin/paradis ./tests/Tantalum.ctrl
```

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

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.

Eshelby inclusions may also be introduced into ParaDiS simulations. The inclusion data is provided in a separate file which is also described 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).

4.3 Eshelby Inclusion File

Eshelby inclusions can be inserted into a ParaDiS simulation by providing a file with a list of inclusions. The name of this file is specified via the <inclusionFile> control parameter. A sample inclusion file has been provided and is located at:

<ParadisDir>/inputs/eshelby.dat

The inclusion file consists of one or more inclusions defined by the following values:

id position radius strain_field

where:

id

Integer value used to uniquely identify the inclusion. (Used internally primarily for debugging purposes)

position

Coordinates (x, y and z) of the center of the inclusion.

radius

Radius of the inclusion in units of b

strain_field

Six values specifying the strain field (s) for the inclusion in the order: s[0][0], s[1][1], s[2][2], s[1][2], s[0][2], s[0][1]

Values in the file are delimited by white space, and data for a single inclusion may span multiple lines.

Additionally, blank lines or lines beginning with a '#' are treated as comments and ignored.

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 Simulation Frame

For typical ParaDiS simulations, the dislocation network is defined in a crystallographic frame appropriate to the type of material (BCC, FCC, etc.) being simulated. However, ParaDiS does provide a means by which the user may define a laboratory frame to be used rather than the crystallographic frame.

The <useLabFrame> control parameter is a simple toggle used to select between the lab and crystallographic frames. Setting this value to 1 enables use of a user-defined frame while setting it to 0 (the default setting) uses the standard crystallographic frame. If a laboratory frame has been selected, the frame axes must be defined via the control parameters <labFrameXDir>, <labFrameYDir> and <labFrameZDir>. Each of these parameters is a 3-component vector specifying the corresponding frame axis.

IMPORTANT NOTE: When use of a laboratory frame is enabled, the dislocation network defined in the nodal data file MUST be specified in the provided frame. This includes the dislocation burgers vectors and glide planes. Although the standard ParaDiS release does provide a utility (paradisgen) for generating nodal data files with initial dislocation networks, this utility generates dislocation networks in the crystallographic frame only and does not have a means of specifying a user-defined frame. Additionally, any values for the following control file parameters must be specified in the laboratory frame:

- <edotdir>
- <appliedStress>
- <sessileburgspec>
- <sessilelinespec>

Additionally, if anisotropic elasticity has been enabled (see the Anisotropic Forces section), the following anisotropy-related parameters must be specified in the laboratory frame as well:

- C11
- C12
- C13
- C33
- C44
- ElasticConstantMatrix

7 Domain Decomposition

7.1 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.

7.2 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.

7.3 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.

8 Dislocation Mobility

8.1 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.

8.2 Available Mobility Modules

The currently supported mobility modules are:

- "BCC_0"
- "BCC_0b"
- "BCC_0_eshelby"
- "BCC_glide"
- "BCC_faceted"
- "BCC_Fe_nl"
- "BCC_Fe_nl_a"
- "BCC_nl"
- "BCC_Ta_nl"
- "BCC_Ta_nl_b"
- "BCC_Ta_nl_b_planar"
- "BCC_Va_nl"
- "BCC_Va_nl_planar"
- "FCC_0"
- "FCC_0b"
- "FCC_climb"
- "HCP_linear"
- "Rhombohedral_Va_nl_planar"
- "Relax"
- "Relax_glide"

For a list and brief description of each 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.

8.2.1 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.

8.2.2 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.

8.2.3 BCC_0_eshelby Mobility

This is a variant of the BCC_0 mobility which is intended for use when eshelby inclusions have been inserted into a simulation. There is a glide mobility for screws, a glide mobility for edges, and a climb mobility for edges. In this version, the edge climb mobility is also the mobility of junction dislocations. The generalization is that there can be different coefficients for the mobility of a dislocation inside an eshelby inclusion than in the matrix. Currently all of the mobilities are linear both in the matrix and within the particles. The nodal drag is calculated by performing a piecewise integral of the different drags operating on the

segment, convoluted with the shape function of the line segment.

The mobility is able to handle segments that intersect multiple particles (current limit is 4) along the segment, but may cause errors if three particles overlap in space. This condition is expected only in the densest of conditions and is truly a non-physical condition, so effort was not put into fixing the limitation.

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.

8.2.4 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

8.2.5 BCC_faceted Mobility

This mobility is intended for BCC materials in which mobility is reduced for certain line directions. The mobility is linear in terms of the velocity of a dislocation being proportional to the force for every orientation, however, the value of the linear constant (drag or mobility) is different for some of the line directions. The mobility is a planar mobility in that it restricts dislocation motion to the glide planes identified by the segment information.

The standard faceting is controlled by adding additional drag to dislocations oriented within a specific angular window of a given angle. One period of the (1/2*cos(theta)+1) from –pi < theta < +pi is used to specify the additional drag where theta/pi = (dislocation character angle – window center) / window_width. Up to 10 windows (each with a unique center, width and added drag) may be specified.

An optional "cusp" variant of the faceting can be selected via the <MobFacetedUseCusps> control file parameter. This simply provides a different way to describe the mobility change across segment character. A cusp shape will have a flat background that changes smoothly to a sharp tip at the specified angle. It is similar to the trench mobility in that both describe the mobility change across segment character (for instance from edge to mixed to screw). However, it is different in the sense that for the trench mobility one adds additional terms to slow the mobility inside the trench, while for cusps the mobility changes smoothly between fast and slow values at the tip of the cusp.

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

MobClimb

MobGlide

MobNumTrenches

MobTrenchAngle

MobTrenchWidth

MobTrench

MobFacetedUseCusps

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.

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.

8.3 BCC_Fe_nl Mobility

This mobility module is intended for simulating a wide range of temperature and rate behavior in BCC iron. It is primarily based on the BCC_Ta_nl module with the functions for calculating screw and edge drag replaced with new functions using iron-specific mobility parameters from molecular dynamics simulations fitted to smooth numerical functions for DD simulations.

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

TempK

Use of this module by default disables the <enforceGlidePlanes> control parameter toggle. 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 disabled with this mobility module.

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.

8.3.1 BCC_Fe_nl_a Mobility

Description TBD

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

MobClimb

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.

8.3.2 BCC_nl Mobility

The BCC_nl mobility module calculates motion in Tantalum by combining non-linear velocity components for screw dislocations and linear components for edges. The mobility formulae and constants used in this module are based on input from MD simulations and are described in detail in the file:

<ParadisDir>/docs/NonLinearmobility.ps

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

MobClimb

MobGlide

MobLine

MobExpP

MobExpQ

MobPieirls

MobDeltaH0

MobAlpha

MobCoeffA

MobCoeffC

MobCoeffC0

MobCoeffD

8.3.3 BCC_Ta_nl Mobility

This mobility module calculates dislocation motion in tantalum combining non-linear velocity components for both screw and edge. In this version of mobility, nodes attached to 4 or more segments are pinned while junction nodes are permitted to move only along the junction line, allowing the junction to expand or contract. The mobility formulae and constants used in this module are based on inputs from MD simulations and are described further in:

<ParadisDir>/docs/TaNonLinearMobility.ps

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

MobClimb

MobGlide

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.

8.3.4 BCC_Ta_nl_b Mobility

This module is a variant of BCC_Ta_nl in which the undesirable functions of bsline and beline have been replaced in the EdgeDrag() and ScrewDrag() routines with non-linear edge and screw functions along with a multiplier. As with the BCC_Ta_nl module, nodes attached to 4 or more segments are pinned while junction nodes are permitted to move only along the junction line, allowing the junction to expand or contract.

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.

See description of BCC_Ta_nl mobility module for additional information.

8.3.5 BCC_Ta_nl_b_planar Mobility

This is a variant of the BCC_Ta_nl_b mobility module in which the motion of dislocations has been restricted to defined glide planes.

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.

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.

See descriptions of BCC_Ta_nl and BCC_Ta_nl_b mobility modules for additional information.

8.3.6 BCC_Va_nl Mobility

The BCC_Va_nl mobility module combines non-linear velocity components for both edge and screw to calculate dislocation motion in Vanadium. The mobility formulae and constants used in this module are based on input from MD simulations.

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.

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

MobClimb

MobGlide

MobLine

MobG0

8.3.7 BCC_Va_nl_planar Mobility

This is a simple variant of the BCC_Va_nl mobility module which tries to impose planar motion of screws based on force projections. See description of BCC_Va_nl module for additional information.

8.3.8 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

8.3.9 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

8.3.10 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

8.3.11 HCP_linear

The HCP_linear mobility module uses a Newton Raphson procedure to compute the linear nodal velocities by solving the equation:

$$F_{elastic}(v) + F_{drag}(v) = 0$$

The elastic forces $F_{elastic}(v)$ at the node are given as input and the drag function $F_{drag}(v)$ is constructed using data from external models (i.e. molecular dynamics models). The MD data is supplied to the simulation via drag coefficients and a set of three core energy prefactors associated with specific types of burgers vector (see list of associated control file parameters below) for the different HCP planes for both edge and screw components. All HCP specific control parameters default to values corresponding to Beryllium at ambient temperature.

The hexagonal crystal lattice for HCP materials is often represented by four Miller indices. This is not convenient in the current implementation of ParaDiS so a different system of axes has been used to represent HCP crystal slip systems. This system is:

$$a\_1 = a\ ex$$

$$a\_2 = -\ \tfrac{1}{2}\ a\ ex + \sqrt{3}/2\ a\ ey$$

$$a\_3 = -\tfrac{1}{2}\ a\ ex - \sqrt{3}/2\ a\ ey$$

The axial relationships for HCP lattices are different from those of BCC and FCC crystals. In BCC and FCC crystals, lattices are such that the axial relationship for the cell geometry is a = b = c and the inter-axial angles are alpha = beta = gamma = 90. In HCP crystals,

the lattice is such that a = b != c, and alpha = beta = 90 and gamma = 120. The change of coordinate system takes care of the gamma angle so that in the new system angle(a1, a2) = angle(a1, a3) = angle(a2, a3) = 90.

For HCP simulations, a new parameter <cOVERa> has been introduced to insure the axes are normalized. By definition, Burgers vectors and planes in the HCP lattice belong to the set of axes (a_1, a_2, a_3) which means they will always be a function of (a a c) or (1 1 c/a).

Use of this module by default disables the <enforceGlidePlanes> control parameter toggle. 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.

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.

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:

cOVERa

HCP_A_Basal_EdgeDrag

HCP_A_Basal_ScrewDrag

HCP_A_Prismatic_EdgeDrag

HCP_A_Prismatic_ScrewDrag

HCP_A_1stPyramidal_EdgeDrag

HCP_A_1stPyramidal_ScrewDrag

HCP_A_2ndPyramidal_EdgeDrag

HCP_A_2ndPyramidal_ScrewDrag

HCP_CpA_Prismatic_EdgeDrag

HCP_CpA_Prismatic_ScrewDrag

HCP_CpA_1stPyramidal_EdgeDrag

HCP_CpA_1stPyramidal_ScrewDrag

HCP_CpA_2ndPyramidal_EdgeDrag

HCP_CpA_2ndPyramidal_ScrewDrag

HCP_C_Prismatic_EdgeDrag

HCP_C_Prismatic_ScrewDrag

HCP_Sessile_EdgeDrag

HCP_Sessile_ScrewDrag

HCP_LineDrag

8.3.12 Rhombohedral_Va_nl_planar Mobility

This module calculates dislocation motion in the rhombohedral phase of Vanadium by combining non-linear velocity components for edge and screw with a linear 'fast' component. The motion is heterogeneous in that velocity becomes reduced within certain 'trenches' leading to facets in the dislocation structure. This mobility is also planar in that it restricts dislocation motion to the glide plane identified by the normal direction in the segment information.

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.

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.

8.3.13 Relaxation Mobilities

Unlike most of the other mobility modules, these relaxation mobilities are not material (i.e. BCC, FCC, etc) specific and can be used for any type of material (see description of <materialTypeName> control parameter). If no <materialTypeName> is specified in the control file, these relaxation mobilities will default to "BCC".

The relaxation mobility modules employ a 'steepest descent' method which permits the dislocation structure to relax toward the lowest energy configuration. In the relaxation mobilities, the velocity of a dislocation node is directly proportional to its force using the calculation V = dot(M, F) / L where F is the nodal force, L is the dislocation length associated with the node, and M is a diagonal matrix. The control parameters <MobRelaxX>, <MobRelaxY>, and <MobRelaxZ> define the M_{11}, M_{22}, and M_{33} components of matrix M, and should typically be set to 0 or 1. For instance, when <MobRelaxX> is set to 0, there will be no motion along the X direction of the simulation space.

Optionally, the velocity calculation may be altered so that the velocity is proportional only to dot(M, F) with no scaling by dislocation length. This can be accomplished by setting the <MobRelaxScaleByLength> control parameter to zero. By default, this parameter is set to 1.

There are two variations of the relaxation mobility modules:

- "Relax"
- "Relax_glide"

The first behaves exactly as described above. The second differs in that dislocation motion is also restricted to the glide planes associated with the

dislocation segments, and use of this second 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 second relaxation mobility module.

8.4 Defining Sessile Dislocations

The ParaDiS code provides a mechanism by which to artificially 'freeze' dislocations with specific burgers vectors and prevent them from moving through the simulation space. This is accomplished by defining the sessile burgers vector and line sense pairs via the <sessileburgspec> and <sessilelinespec> control file parameters. These arrays are defined such that for each burgers vector in <sessileburgspec> there is a corresponding line sense defined in <sessilelinespec>. If all components of a particular line sense are zero, any dislocation with the specified burgers vector will be sessile, but if the associated line sense has non-zero components, only dislocations matching both the burgers vector and line sense will be sessile. (Note: the current implementation restricts to 9 the number of sessile burgers vector/line sense pairs.)

The contents of <sessileburgspec> are of the form:

```
sessileburgspec[0] = # of sessile burgers vectors

sessileburgspec[1] = burg0 X component

sessileburgspec[2] = burg0 Y component

sessileburgspec[3] = burg0 Z component

...

sessileburgspec[N*3+1] = burgN X component
```

sessileburgspec[N*3+2] = burgN Y component

sessileburgspec[N*3+3] = burgN Z component

The contents of <sessilelinespec> are of the form:

sessilelinespec[0] = <ignored>

sessilelinespec[1] = line sense 0 X component

sessilelinespec[2] = line sense 0 Y component

sessilelinespec[3] = line sense 0 Z component

...

sessilelinespec[N*3+1] = line sense N X component

sessilelinespec[N*3+2] = line sense N Y component

sessilelinespec[N*3+3] = line sense N Z component

NOTE: When a user-specified laboratory frame is being used rather than the default crystallographic frame (see section on "Simulation Frame"), the burgers vectors and line directions in <sessileburgspec> and <sessilelinespec> must be specified in the laboratory frame.

9 Dislocation Network Evolution

In ParaDiS simulations the dislocation network evolves naturally over time as the dislocations move through the simulation space and interact with each other. This evolution is affected by a variety of operations and events. Certain of these operations have been implemented in more than one manner, or have associated control parameters that can affect their behavior. Some of these operations are briefly described below along with their associated control parameters.

9.1 Rediscretization (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.

9.1.1 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

9.1.2 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.

9.2 Collision Handling

There are two basic methods for detecting and implementing a physical collision between two dislocation segments. The first is an older 'proximity' mechanism in which the decision to collide two segments is based primarily on their current proximity to each other. Any segments approaching within some critical distance will undergo a physical collision. The second is a 'predictive' method which uses an initial proximity criteria to select collision candidates, then uses a predictive algorithm to determine not only if the segments will collide, but when and where in time. This data is used to decide if a physical collision between segments is deemed appropriate.

Once the decision has been made to collide two dislocation segments, both collision mechanisms use a similar process to perform the actual collision. The physical point at which the segments will collide is calculated, as well as the closest position to that collision point along each segment involved in the collision. For each segment, one of two things occurs. Either the segment is bisected by a new node at the segment's collision point, or the segment collision point is sufficiently close to one of the segment's endpoints to allow the endpoint to be used as the collision point. The two nodes (whether new nodes or existing segment endpoints) are then merged together and the dislocation network is adjusted accordingly.

The <collisionMethod> control parameter can be used to select the collision handling method used for a simulation. Setting this parameter to 1 selects the old proximity collision handling while a setting of 2 selects the newer predictive method. If no collision method is explicitly provided, the code will use the predictive method by default.

9.3 Cross-Slipping

The ParaDiS code contains a mechanism by which a dislocation segment moving in a defined glide plane can shift (cross-slip) to a different glide plane. This mechanism is enabled/disabled via the control parameter toggle <enableCrossSlip>, however, it should only be enabled with the glide-restricted mobility modules. The default state of this capability is specific to the selected mobility module. See the descriptions of the individual mobility modules to determine the default state of the <enableCrossSlip> toggle.

Dislocation cross-slip is treated as a topology-altering operation because it modifies the node connectivity data. It neither introduces nor removes any nodes, but it does alter the glide plane data for dislocation segments and can effect small changes in the nodal positions.

There is a unique cross-slip module for each material type, but the different modules all share a similar basic design. Each of the routines evaluate all discretization nodes (nodes with exactly 2 associated segments) that are close to screw. Closeness to screw is determined by a critical angle (1 degree as of the writing of this document). If either of the node's segments is close to screw, the node is considered for a possible cross-slip operation. A test is conducted to determine which of the screw's glide directions sees the greatest force projection. A threshold is defined so the node's preference is to remain on the primary (current) glide plane, so only when the greatest force projection is on a glide plane other than the primary and the force on the cross-slip plane exceeds the threshold will a cross-slip event be attempted.

There are two types of cross-slip that must be considered.

A) Both of the nodes segments are on the same plane (standard cross-slip)
B) The node's segments are on two different planes with one plane being the intended cross-slip plane ('zipper' type cross-slip)

In case A, one of the neighboring nodes is moved (if necessary) into perfect screw alignment, the segment between the node and its neighbor is shifted to the cross-slip plane, and the node is moved onto the cross-slip plane

and a small areal nucleus is created. In case B, one of the neighboring nodes is moved into perfect alignment with the node and the connecting segment is shifted to the cross-slip plane.

Additionally, when a normal cross-slip event is attempted, a new force calculation is performed to determine if the cross-slip nucleus continues to move outward into the cross-slip plane, or whether it tends to move back and cause cross-slip flicker. If a flicker is detected, then the cross-slip event is backed out and the node returned to its original state. Similarly, when a 'zipper' type cross-slip event is anticipated, the force on the individual segments being affected is examined. If the force on that segment is higher in the cross-slip plane than on the primary plane, the cross-slip event is performed.

There are conditions under which a cross-slip cannot be attempted. See the actual cross-slip modules for details on the restrictions.

9.4 Multi-Node Splitting

Since ParaDiS has a mechanism allowing dislocation segments to collide and fuse together, it also provides a dissociation mechanism by which multi-nodes (nodes with 4 or more dislocation segments) can be split into two distinct nodes, moving at least 2 of the original node's segments to the new node.

For each multi-node in the simulation, ParaDiS uses this basic method to determine if dissociation is required:

- A power dissipation value is calculated for the original multi-node
- The multi-node is split into every permitted unique split configuration
- For each configuration:
    - Nodal velocities for the two nodes involved are recalculated and the nodes are separated a small distance based on those velocities
    - Full forces are recomputed for each of the two involved nodes and velocities once again computed

- A power dissipation value is calculated from the two dissociated nodes
- The configuration is fused back together into the original multi-node restoring it back to its pristine state

The decision as to whether to split the multi-node is done only after all configurations have been evaluated. If the rate of power dissipation is greatest with the multi-node intact, it is left untouched. However, if one or more dissociated configurations result in higher power dissipation, then the configuration with the highest power dissipation is selected and the multi-node is split apart to conform to that configuration.

There are several control file parameters that can affect the behavior of the multi-node splitting functions (see the descriptions of these parameters below or in the "Control File Parameters" section of the Appendix). These parameters are:

<splitMultiNodeFreq>

<useParallelSplitMultiNode>

<useUniformJunctionLen>

.

Due to the need to perform force calculations for the nodes in each dissociated configuration, the process of splitting multi-nodes can expensive. To help mitigate this expense, an optional control parameter <splitMultiNodeFreq> has been provided which allows the user to control the frequency with which the multi-node splitting will be attempted. During execution, multi-node splitting will be attempted each cycle that is a multiple of the specified value.

NOTE: Preventing ParaDiS from splitting multi-nodes for a small number of cycles is not likely to cause any problems, however, doing so for large numbers of cycles can create highly stressed configurations which (when finally split) result in nodes with extremely high velocities until the connected segments can release their store up energy. Additionally, artificially prolonging the lives of multi-nodes makes it more likely that the multi-nodes themselves will undergo additional collisions resulting in multi-nodes with more and more segments. When such multi-nodes begin

forming, they act as pinning points for large numbers of segments in very localized areas. Those segments may then begin interacting/colliding with each other forming yet more multi-nodes. This process can lead to artificial tangles that eventually cannot be cleared up, adversely affecting the simulation.

There are currently two mechanisms in ParaDiS for doing multi-node splitting, both of which implement the method described above. The mechanism used in a simulation can be controlled via the setting of the <useParallelSplitMultiNode> control file parameter. This parameter is a simple toggle. Its default value of zero selects the first multi-node splitting module, while setting the toggle to 1 enables the use of a 'parallel' multi-node splitting module.

The first mechanism is as described above where each domain evaluates every multi-node it owns, handling all configuration changes, force and velocity calculations, and the ultimate decision as to whether to dissociate the node. As mentioned above, this can be very computationally expensive. When a small subset of the simulations processors own a disproportionate number of multi-nodes, this can cause significant waste of cpu resources as the majority of processors idle while the smaller number split their multi-nodes.

The second mechanism was designed as a possible way to distribute the expense of the force calculations to processors that might otherwise be underutilized. In this version each ParaDiS domain identifies all locally owned multi-nodes and remote multi-nodes in nearby cells. Each domain then splits each multi-node into its various configurations as described above and calculates the components of the force on those dissociated nodes from the segments owned by the local domain. These 'partial' forces for a split multi-node are then communicated back to the domain owning the multi-node. The owning domains then sum the partial forces and proceed with evaluating the power dissipation calculations and make the final determination as to whether the multi-node should or should not be split. (NOTE: The cost of the MPI communications and other overhead involved in this parallel multi-node splitting is significant. It should only be employed when a small number of domains incur the largest percent of the cost of multi-node splitting and the simulation is spending a

significant portion of time splitting multi-nodes. If this mechanism is used when the cost or imbalance among processors for multi-node splitting is low, the result can be a loss in efficiency rather than a gain.)

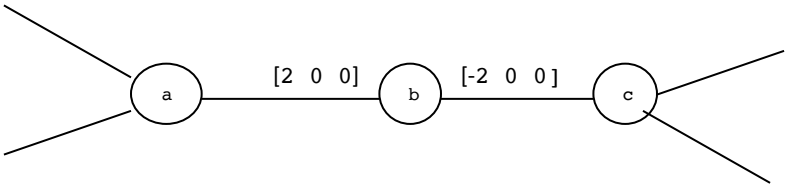
There is one additional control parameter that affects the behavior of the multi-node splitting functions. When a multi-node is split, the two resulting nodes are repositioned a small distance away from each other (with a new junction segment between them if appropriate). The <useUniformJunctionLen> toggle selects whether the splitting functions separate the two nodes by a uniform length, or attempt to use a line-tension based algorithm to attempt to find a more optimal length. The default value of this parameter is zero allowing the code to attempt to find an optimal length while setting the toggle to 1 forces the code to use a set length (currently 4 * <rTol>).

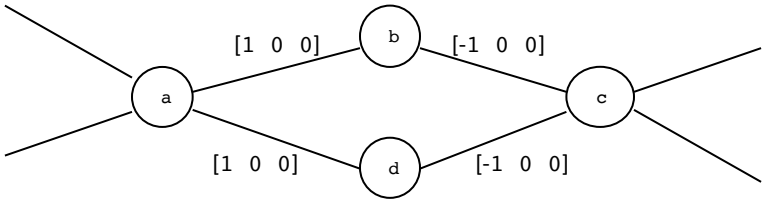
9.5 Segment Splintering

In some materials, it is possible to form dislocation segments that are permitted to undergo a form of mitosis, or "splintering". These segments are such that they can be decomposed into two identical and mutually repulsive segments of a different burgers vector in a manner that is energetically favorable. For example, in HCP materials a junction segment with a [2 0 0] burgers vector can be splintered into a pair of segments with a [1 0 0] glide burgers vectors.

The splintering process occurs when a node with exactly two segments (each of the same glide plane and burgers vector) is found to have a burgers vector that is considered splinterable. At this point, the burgers vector of each of the node's segments is given a new burgers vector (which sum to the original burgers vector). A new node is created at the same location as the original node and given two segments identical to the original node's segments. Lastly, the two nodes are given a small initial separation, after which they will continue to move apart due to the repulsive nature of the new segment pairs.

For instance, given the HCP [2 0 0] junction segments ab and bc below, a splintering into [1 0 0] segments would occur at node b resulting in the shown configuration change.


a
[2 0 0]
b
[-2 0 0]
c


b
[1 0 0]
[-1 0 0]
a
c
[1 0 0]
d
[-1 0 0]

10 Material Properties

10.1 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.

10.2 Elastic Constants File

Rather than explicitly specify the material specific control parameters <burgMag>, <shearModulus> and <pois>, the user may provide a set of files containing the material specific isothermal elastic constants and atomic volume per atom over a range of temperatures and pressures. ParaDiS will then use these files (along with the current simulation temperature and pressure) to calculate the proper values for the above mentioned parameters.

To make use of this capability, the user supplies the 'base' name of the files containing the constants via the <ecFile> control parameter. All related files are assumed to have the same base file name, using the naming convention:

<base>.c11.dat

<base>.c12.dat

<base>.c44.dat

<base>.burg.dat

The <base>.c11.dat, <base>.c12.dat and <base>.c44.dat files contain the values for the specified isothermal elastic constants through a range of temperature and pressure values. Temperatures are in degrees K, pressures and constants values are in units of Mbar. The format of these files is:

<temp1> <pressure1> <const_value>

<temp1> <pressure2> <const_value>

…

<temp1> <pressureN> <const_value>

<temp2> <pressure1> <const_value>

<temp2> <pressure2> <const_value>

…

<temp2> <pressureN> <const_value>

Additionally, blank lines or lines beginning with '#' are ignored as is anything following the pertinent values on a valid data line.

The <base>.burg.dat file contains a table of values for the atomic volume per atom though a range of temperature and pressure values. Temperatures are in degrees K, pressures in units of Mbar, and the volume constants in units of cubic Bohrs. The format of the file is the same as described above for the elastic constants files.

10.3 Melting Temperature vs. Pressure Table

Rather than explicitly specifying the material's melting temperature, the user may provide a file containing a table of the material's melting temperature over a range of pressures. ParaDiS will use this file to calculate the proper melting temperature for the current simulation pressure.

To make use of this capability, the user supplies the name of the file containing the table via the <meltTempFile> control parameter.

The contents of this file should be formatted as:

<temp1> <pressure1>

<temp2> <presure2>

…

<tempN> <pressureN>

where temperatures are specified in degrees K and pressures in units of Mbars.

Additionally, blank lines or lines beginning with '#' are ignored as is anything following the pertinent values on a valid data line.

10.4 Mobility Constants File

Rather than specify the material specific control parameters <MobPieirls>, <MobDeltaH0>, and <MobAlpha> which are required for certain mobility modules, the user may provide a file containing a table of these values over a range of pressures. ParaDiS will use this file to calculate the proper melting temperature for the current simulation pressure.

To make use of this capability, the user supplies the name of the file containing the table via the <mobFileNL> control parameter.

The contents of this file should be of the form:

<pressure1> <temp1> <deltaH_0> <pieirls> <alpha>

<pressure2> <temp1> <deltaH_0> <pieirls> <alpha>

…

<pressureN> <temp1> <deltaH_0> <pieirls> <alpha>

<pressure1> <temp2> <deltaH_0> <pieirls> <alpha>

<pressure2> <temp2> <deltaH_0> <pieirls> <alpha>

…

<pressureN> <tempM> <deltaH_0> <pieirls> <alpha>

where temperatures are specified in degrees K and pressures in units of GPa.

Additionally, blank lines or lines beginning with '#' are ignored as is anything following the pertinent values on a valid data line.

11 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 grouping all segments in the remote cell into a 'super-dislocation' where the group of remote dislocations is represented as an expansion of dislocation multipoles.

There are two methods by which segment self-force and segment to segment forces may be calculated. The default method uses isotropic elasticity, while an optional method using anisotropic elasticity can be selected at compile time. See the section on "Anisotropic Forces" for more details. Note: the anisotropic method has not yet been implemented for remote force calculations, so use of the anisotropic code will currently mix anisotropic local forces (self-force, seg/seg force) with isotropic remote forces.

With the default method of doing isotropic force calculations in ParaDiS, there is a divergence of the stress field on an element by element basis that cancels when the total segment configuration closes on itself or is bounded by semi-infinite segments. Modified versions of the functions for calculating isotropic forces versions can be selected at compile time by defining the _USE_RATIONAL_SEG_FORCES flag in <ParadisDir>/makefile.setup. These modified functions (or "rational" versions) include a correction in their formulation that makes the stress field due to a segment of dislocation divergence free on a segment by segment basis. Use of these rational functions means that the system does not have to be closed or terminated by semi-infinite segments to create a divergence free stress field. The correction term comes in the form of

$$d\sigma_{ij}^{corr}[R_i, b_l, t_k] = -\frac{\mu}{8\pi}\left[\frac{e_{jkl}(R_i t_k b_l + t_i R_k b_l) + e_{ikl}(R_j t_k b_l + t_j R_k b_l)}{R_a^3} + 3R_m t_m \frac{e_{jkl} R_i R_k b_l + e_{ikl} R_j R_k b_l}{R_a^5}\right]$$

where $R_a^5 = \sqrt{R_k R_k + a^2}$ and b_l is the burgers vector of the dislocation segment, t_k is the tangent line direction of the dislocation segment, and R_i is the distance vector from a point on the dislocation segment to a field point where the stress is being evaluated. In order to get the correction at a point due to a segment, the above expression is integrated over the dislocation segment. The result is the following expression

$$\sigma_{ij}^{corr}\left[R_i^1, R_i^2, b_l, t_k\right] = -\frac{\mu}{8\pi}\left[\frac{e_{jkl}R_i^2 R_k^2 b_l + e_{ikl}R_j^2 R_k^2 b_l}{\left(\sqrt{R_m^2 R_m^2 + a^2}\right)^3} - \frac{e_{jkl}R_i^1 R_k^1 b_l + e_{ikl}R_j^1 R_k^1 b_l}{\left(\sqrt{R_m^1 R_m^1 + a^2}\right)^3}\right]$$

Where R_i^1 and R_i^2 are the starting and ending distance vectors from the dislocation segment to the field point. The calculation of the segsegforces requires integration of the σ_{ij}^{corr} along a second dislocation line to calculate the force on the end nodes of the dislocation segment.

11.1 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 <vacancyConc> 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).

Lastly, if eshelby inclusions have been inserted into the ParaDiS simulation, the force on each dislocation will be adjusted to account for direct interactions with all inclusions in the same or immediately neighboring cells. This feature will automatically be enabled if the <inclusionFile> control parameter is set.

11.2 Far-Field Dislocation Interactions

11.2.1 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.

Additionally, if ParaDiS has been compiled with support for calculating anisotropic forces (See section on Anisotropic Forces), either isotropic or anisotropic FMM forces can be calculated. This selection is done at run time via the control file parameter <fmEnableAnisotropy>. It is important to note, however, that the FMM correction tables have a dependency on whether the FMM calculations are to be isotropic or anisotropic. (See the Utilities section on "Creating an Image CorrectionTable with Ctablegen" for details.)

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.

11.2.2 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.

11.2.3 FMM Far-Field Interactions with Eshelby Inclusions

When eshelby inclusions have been inserted into the ParaDiS simulation, the default behavior is for the dislocation segments to interact only with inclusions in the same or immediately neighboring simulation cells. However, the FMM algorithm described above can also be used to calculate forces on dislocations from more remote inclusions. This mechanism is enabled/disabled via the <eshelbyfmEnabled> control parameter toggle. The order of the multipole and taylor expansions for inclusions are separate from those used for the regular FMM and are set via the <eshelbyfmMPOrder> and <eshelbyfmTaylorOrder> control parameters. Currently there is a limit such that the combined multipole and taylor expansion orders for the eshelby FMM must not exceed a value of 6.

Note: One difference between the regular FMM and the eshelby FMM is that there is no correction table for use with the eshelby FMM.

11.3 Anisotropic Forces

As mentioned above, there are two methods by which segment self-force and segment to segment interaction forces may be calculated. The default method uses isotropic elasticity, but the use of anisotropic elasticity can be enabled at compile time by uncommenting the following line in the <ParadidDir>/makefile.setup file:

```
DEFS += -D_ANISOTROPIC
```

In most cases, when the anisotropic code is enabled, isotropic constants (shear modulus, Poisson ratio and Youngs modulus) are no longer treated as input parameters in the control file, but will be dynamically calculated based on the additional control parameters specific to the anisotropic code. Simulations involving rhombohedral crystals are an exception to this rule. See the section on "Control File Parameters for Anisotropic Forces" for details.

When anisotropy is enabled, stresses and forces are computed as truncated series. These series are based on an expansion in spherical harmonics of the angular part of the Green's function where the single and double integrals corresponding to stress and force respectively can be switched with the series and computed via recurrence. The derivative of the angular part of the Green's function is specified as

$$dGdx = g(T) / R^2$$

where

$$g(T) = \text{Sum\_q Sum\_m } g\_q^m \; Y\_q^m \; (T)$$

and

R is the distance between the two dislocation segments,

q is the expansion order,

g is the angular part of the derivative of the Green's function,

Y are the spherical harmonics functions,

g_q^m are the expansion coefficients.

The expansion coefficients g_q^m are defined as a double integral over spherical coordinates (theta, phi). The number of angles (theta, phi), are defined via the control file parameters <anisoNumThetaPoints> and <anisoNumPhiPoints>. A reasonable precision in the integral calculations of this double integral is obtained for 150 x 150 angles. See the paper XXXXX for more details regarding the number of chosen grid points and error in the integral calculations.

The spherical harmonics series are composed of a sum of <anisoHarmonicsNumTermsBase> terms (the sum on q above). The higher the anisotropy of the materials, the higher the required expansion order, however, the computational cost of the anisotropic calculations is also proportional to the expansion order.

From the expression of dGdx, we can deduce an expression of the stress and the force. These expressions are composed of a matrix that can be pre-computed and integrals (single for stress and double for force) that are computed for each dislocation segment or pair of segments.

11.3.1 Control File Parameters for Anisotropic Forces

NOTE: for cubic and hexagonal crystal materials, the use of the anisotropy code explicitly overrides any control file parameters provided for the isotropic constants <shearModulus>, <pois>, and <YoungsModulus>. These values will be dynamically recalculated as necessary. For rhombohedral crystals, the <shearModulus> and <pois> parameters MUST be provided.

Additional parameters for specifying the elastic constants and anisotropy-specific data are listed in

the table below. For cubic and hexagonal crystals, either the full 6 x 6 matrix <elasticConstantMatrix> or the individual elastic constants (<C11>, <C12>, etc.) must be provided. If both are provided, the <elasticConstantMatrix> will take precedence over the individual elastic constants.

By default, ParaDiS simulations are done in the crystallographic frame and all these elastic constants, whether provided individually or via the full <elasticConstantMatrix>, must be provided in the crystallographic frame as well. However, if the simulation is in a user-defined frame (see "Simulation Frame" section for more information) the elastic constants, must be provided in the frame matching that defined by the user.

Parameter	Material Type
C11	BCC, FCC, HCP
C12	BCC, FCC, HCP
C13	HCP
C33	HCP
C44	BCC, FCC, HCP
elasticConstantMatrix	ALL
anisoHarmonicsNumTermsBase	ALL
anisoNumPhiPoints	ALL
anisoNumThetaPoints	ALL

See the Appendix for brief descriptions of any of these parameters.

12 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 currently supported integration methods are:

"trapezoid"

"trapezoid-kinsol"

"arkode"

"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.

12.1 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 is within the maximum positioning error (defined by <rTol>), the deltaT is accepted, and the deltaT is incremented by the factor given in <dtIncrementFactor>

to be used as the initial timestep to attempt the next time cycle. If the positioning error exceeds the <rTol> value, the code performs up to <trapezoidMaxIterations> of the repositioning loop calculating new positions based on the previous positions, recalculating forces, velocities and positioning errors. If during any of the subsequent loop iterations, the maximum positioning error is within the <rTol> value, the deltaT value is accepted, but is not altered in preparation for the next timestep. However, if after all iterations of the loop to reposition the nodes, the positioning error still exceeds <rTol>, the deltaT value is decremented by the factor specified in <dtDecrementFactor>, and the entire timestep integration process is repeated with the lower deltaT.

12.2 Trapezoid-KINSOL Timestep Integrator

NOTE: This timestep integrator is dependent on KINSOL which is a general purpose non-linear system solver based on Newton-Krylov solver technology and is part of the SUNDIALS software suite. In order to use this timestep integrator, the KINSOL libraries must be installed and the use of KINSOL must have been enabled during compilation by setting the "KINSOL_MODE = ON" flag in 'makefile.setup' and by defining in 'makefile.sys' the following variables for the appropropriate system type:

KINSOL_DIR.<sys_type>

KINSOL_LIB.<sys_type>

KINSOL_INCS.<sys_type>

The trapezoid-kinsol integrator is similar to the basic "trapezoid" integrator but is implemented using KINSOL with either an Anderson accelerated fixed point nonlinear solver or a Newton-Krylov nonlinear solver.

As with the 'trapezoid' integrator, a maximum positioning 'error' defined by the <rTol> control parameter is used as the criteria for determining if the KINSOL solver was able to converge sufficiently to a solution. If the system solve does not converge within the permitted number of iterations, the timestep will be decremented by the factor specified in <dtDecrementFactor> and the process starts

again. Conversely, if KINSOL successfully converges on a solution, 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.

Additional control file parameters specific to this timestep integrator include:

KINSOL_UseNewton

KINSOL_MaxIterations

KINSOL_MaxLinIterations

KINSOL_NumPriorResiduals

KINSOL_EtaVal

When using this timestep integrator, the choice of methods used (fixed point or Newton) determines which of the above parameters are applicable. See the descriptions of the individual parameters in the Appendix for more details

For additional information on KINSOL, refer to the KINSOL User's Guide.

12.3 ARKODE Timestep Integrator

NOTE: This timestep integrator is dependent on ARKode, an adaptive-step time integration package for stiff, nonstiff and multi-rate ordinary differential equations based on Addative Runge Kutta, which is one of the components of the SUNDIALS suite of nonlinear and differential/algebraic equations solvers. In order to use this timestep integrator, the ARKode libraries must be installed and the use of ARKode must have been enabled during compilation by setting the "ARKODE_MODE = ON" flag in 'makefile.setup' and by defining in 'makefile.sys' the following variables for the appropropriate system type:

ARKODE_DIR.<sys_type>

ARKODE_LIB.<sys_type>

ARKODE_INCS.<sys_type>

The methods used by ParaDiS from ARKode are adaptive-step diagonally implicit Runge Kutta (DIRK) methods. Runge Kutta methods are composed of multiple stage solutions combined to obtain new nodal positions. Following the position update within the integrator, the forces and velocities are recalculated for the new positions. With DIRK methods, an iterative nonlinear solve, similar in cost to the trapezoid method, is necessary for each stage in the Runge Kutta method leading to additional nodal force and velocity calculations. The added expense of multiple force and velocity evaluations is usually offset by the gains from larger timestep sizes possible with higher order DIRK methods.

The DIRK methods implemented within ARKode have an embedded lower-order method used to estimate the error in a time step. The weighted root mean square norm (wrms) of the difference between the higher and lower order solutions is compared to the <rTol> control parameter. If the norm is greater than <rTol>, ARKode reduces the time step using an internal error control algorithm and reattempts the step. If the norm is less than <rTol>, the result of the higher order method is accepted and the internal error control algorithm sets the size of the next time step.

Additional control file parameters specific to this timestep integrator include:

ARKODE_FPsolver

ARKODE_FPaccel

ARKODE_MaxLinIters

ARKODE_MaxNonLinIters

ARKODE_LinCoef

ARKODE_NonLinCoef

ARKODE_MaxConvFailGrowth

ARKODE_IRKtable

ARKODE_PredictorMethod

ARKODE_AdaptivityMethod

When using this timestep integrator, the choice of method used (fixed point or Newton) determines which of the above

parameters are applicable. See the descriptions of the individual parameters in the Appendix for more details

For additional information on ARKode, refer to the ARKode documentation which is available in HTML format at http://runge.math.smu.edu/arkode_docs.html.

Note: while there is a one-to-one mapping between the available ARKode options and the ParaDiS input parameters, above, these are not identical. For example, the ParaDiS parameters below are specified through the corresponding ARKode input routines, each of which is documented more fully at the above site:

ARKODE_IRKtable -- ARKodeSetIRKTable()

ARKODE_PredictorMethod -- ARKodeSetPredictorMethod()

ARKODE_AdaptivityMethod -- ARKodeSetAdaptivityMethod()

ARKODE_NonLinCoef -- ARKodeSetNonlinConvCoef()

ARKODE_LinCoef -- ARKSpilsSetEpsLin()

12.4 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

13 Visualization

13.1 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 controled 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

13.2 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.

13.3 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'.)

13.4 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.

13.5 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.

13.6 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 can be used to select which of the VisIt visualization files are generated although by default both nodal and segment data files are written. 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.

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]

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. It is these ".meta" files that are opened via VisIt which uses the meta-data to determine what data is available for visualization.

The nodal data (*.node files) consist of the position of each node along with the number of dislocation segments associated with the node. There are certain optional values that can be included in this file as well. These additional values are the nodal force, and the nodal velocity enabled by the respective control file toggles <writeVisitForceVector> and <writeVisitVelocityVector>. When these toggles are enabled, the force/velocity values will each be included in the file as three-component vectors. By default, these toggles are turned off.

The segment data (*.seg files) consist of the following basic information for each segment: positions of both start and end points of the segment, burgers vector, and glide plane. Additionally, the following types of segment data can optionally be included in the segment files:

Burgers vector ID:

This is a scalar value that is used to associate an integer ID with a specific type or group of burgers vectors. The <writeVisitBurgID> control parameter toggle determines whether this value is included when the segment data is written out. By default this toggle is turned on, enabling the inclusion of this data. This burgers vector ID can be useful in having visit plot dislocation segments with different burgers vectors in different colors. The actual burgers vector ID assigned to the segment is dependent on the type of material being simulated as described in the table below.

BCC	
Burg ID	Burgers Vectors
0	[1 1 1]
1	[-1 1 1]
2	[1 -1 1]
3	[1 1 -1]

4	[1 0 0]
5	[0 1 0]
6	[0 0 1]
7	[1 1 0] [1 -1 0] [1 0 1] [1 0 -1] [0 1 1] [0 1 -1]
8	All other types
FCC	
Burg ID	Burgers vectors
0	[1 1 0]
1	[1 -1 0]
2	[1 0 1]
3	[1 0 -1]
4	[0 1 1]
5	[0 1 -1]
6	[1 0 0]
7	[0 1 0]
8	[0 0 1]
9	All [2 1 1] types
10	All other types
HCP	
Burg ID	Burgers vectors
0	a/2[-1 $\sqrt{3}$ 0]
1	[a 0 0]
2	a/2[1 $\sqrt{3}$ 0]
3	½[-a a*$\sqrt{3}$ 2c]
4	[a 0 c]
5	½[a a*$\sqrt{3}$ -2c]
6	½[-a a*$\sqrt{3}$ -2c]
7	[-a 0 c]
8	½[a a*$\sqrt{3}$ 2]
9	[0 0 c]
10	All other types
Rhombohedral	
Burg ID	Burgers vectors
0	[1.45 1.45 1.45]
1	[-1.1870 1.3185 1.3185]
2	[1.3185 -1.1870 1.3185]
3	[1.3185 1.3185 -1.1870]
4	[0.1315 0.1315 2.637]
5	[0.1315 2.637 0.1315]
6	[2.637 0.1315 0.1315]
7	[0 2.5055 -2.5055] [2.5055 0 -2.5055] [2.5055 -2.5055 0]

	[2.7685 2.7685 0.263] [2.7685 0.263 2.7685] [0.263 2.7685 2.7685]
8	All other types

14 Output

14.1 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.

14.2 Scalable Checkpoint/Restart Files

14.2.1 Overview

Hooks have been placed into ParaDiS to integrate it with the Scalable Checkpoint/Restart (SCR) library. This library permits MPI applications to utilize the distributed storage on Linux clusters for high I/O bandwidth checkpointing and restarting. Using SCR, the application checkpoint files are cached in storage local to the individual compute nodes and a redundancy scheme applied such that cached files can be recovered even if a failure disables a portion of the node allocation associated with the job. SCR supports the use of spare nodes such that it is possible to restart a job directly from the cached checkpoints provided the redundancy scheme holds and there are sufficient spare nodes in the job’s allocation.

14.2.2 Dependencies

The SCR library is not included as part of the ParaDiS distribution. It must be installed separately on local system and the <ParadisDir>/makefile.sys file modified to reflect the location of the installation

before ParaDiS is compiled. The current implementation of SCR also has several dependencies. First, the application must be an MPI based parallel applications which means ParaDiS must be compiled with the "PARALLEL" macro defined in <ParadisDir>/makefile.setup. Additionally, SCR is integrated with SLURM (Simple Linux Utility for Resource management) for job initiation and control. Both SCR and SLURM are freely available and can be downloaded from the web.

SCR can be downloaded from:

http://sourceforge.net/projects/scalablecr

SLURM can be downloaded from:

http://www.schedmd.com/#repos

14.2.3 Enabling SCR

Use of the SCR library in ParaDiS is selected at compilation time via a flag in <ParadisDir>/makefile.setup. By default, this capability is disabled, but can be enabled by simply uncommenting the following line in the make file:

```
DEFS += -DUSE_SCR
```

14.2.4 Using SCR

The behavior of the SCR functions is primarily controlled through a variety of environment variables. A sample job submission script

<ParadisDir>/inputs/msub_SCR_example.sh

has been included in the ParaDiS distribution. This example script is for submitting to the MOAB batch system and includes commands needed to setup the SCR environment and execute the job via SLURM. The script contains a list of many of the standard SRC environment variables and a brief description of how they affect the behavior of the job.

For full details on the capabilities and use of SCR, however, refer to the SCR documentation. This documentation is included in the SCR sourcfe downloadable from the web.

14.3 Property Outputs

14.3.1 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.

14.3.2 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
2	Strain
3	Dislocation density
4	Deleted dislocation density (lost through annihilation, junction formation, etc)

Column	Description
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

Column	Description
	HCP Note: The hexagonal crystal lattice is often represented by four Miller indices. This is not convenient in the current implementation of ParaDiS so a different system of axes has been used to represent HCP crystal slip systems. This system is: a_1 = a ex a_2 = - ½ a ex + $\sqrt{3}$/2 a ey a_3 = - ½ a ex - $\sqrt{3}$/2 a ey The axial relationships for HCP lattices are different than those of BCC and FCC crystals. In BCC and FCC crystals, lattices are such that the axial relationship for the cell geometry is a = b = c and the inter-axial angles are alpha = beta = gamma = 90. In HCP crystals, the lattice is such that a = b != c, and alpha = beta = 90 and gamma = 120. The change of coordinate system takes care of the gamma angle so that in the new system angle(a1, a2) = angle(a1, a3) = angle(a2, a3) = 90. The control parameter <cOVERa> = c/a has been introduced to insure the axes are normalized. By definition, Burgers vectors and planes in the HCP lattice belong to the set of axes (a_1, a_2, a_3) which means they will always be a function of (a a c) or (1 1 c/a).
8	Burgers vectors a/2[-1 $\sqrt{3}$ 0] a/2[1 - $\sqrt{3}$ 0]
9	Burgers vectors [a 0 0] [-a 0 0]
10	Burgers vectors a/2[1 $\sqrt{3}$ 0] a/2[-1 -$\sqrt{3}$ 0]
11	Burgers vectors ½[-a a*$\sqrt{3}$ 2c] ½[a -a*$\sqrt{3}$ -2c]
12	Burgers vectors ½[a 0 c] ½[-a 0 -c]
13	Burgers vectors ½[a a*$\sqrt{3}$ -2c] ½[-a -a*$\sqrt{3}$ 2c]
14	Burgers vectors ½[-a a*$\sqrt{3}$ -2c] ½[a -a*$\sqrt{3}$ 2c]
15	Burgers vectors ½[-a 0 c] ½[a 0 -c]
16	Burgers vectors ½[a a*$\sqrt{3}$ 2] ½[-a -a*$\sqrt{3}$ -2]
17	Burgers vectors [0 0 c] [0 0 -c]
18	All other burgers vectors

14.3.3 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.

14.3.4 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.

14.3.5 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)	

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

14.3.6 Total Stress 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/total_stress

Each time the code writes out properties data a single line will be appended to this file. Each line will contain eight items of data. The first two values are the current simulation cycle number and the current simulation time. The remaining 6 values are components of the appliedStress tensor as ($sigma_{11}$, $sigma_{22}$, $sigma_{33}$, $sigma_{23}$, $sigma_{31}$, $sigma_{12}$).

14.3.7 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

14.3.8 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.

14.3.9 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]

14.4 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.

14.4.1 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)
10	Simulation timestep duration

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
1	Plastic strain
2	Strain
3	Flux due to climb
10	Simulation timestep duration
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]

14.4.2 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)
10	Simulation timestep duration

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
10	Simulation timestep duration

Column	Description
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]
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]

Column	Description
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]

14.4.3 HCP Flux Decomposition

The hexagonal crystal lattice is often represented by four Miller indices. This is not convenient in the current implementation of ParaDiS so a different system of axes has been used to represent HCP crystal slip systems. This system is:

a_1 = a ex

a_2 = - ½ a ex + $\sqrt{3}$/2 a ey

a_3 = - ½ a ex - $\sqrt{3}$/2 a ey

The axial relationships for HCP lattices are different than those of BCC and FCC crystals. In BCC and FCC crystals, lattices are such that the axial relationship for the cell geometry is a = b = c and the inter-axial angles are alpha = beta = gamma = 90. In HCP crystals, the lattice is such that a = b != c, and alpha = beta = 90 and gamma = 120. The change of coordinate system takes care of the gamma angle so that in the new system angle(a1, a2) = angle(a1, a3) = angle(a2, a3) = 90.

The control parameter <cOVERa> = c/a has been introduced to insure the axes are normalized. By definition, Burgers vectors and planes in the HCP lattice belong to the set of axes (a_1, a_2, a_3) which means they will always be a function of (a a c) or (1 1 c/a).

The HCP flux decomposition data consists of two sets of files. Each set contains 10 files, one file per burgers vector. The files Ltot_b1 thru Ltot_b10 contain statistics for burgers vector types as indicated below

- Ltot_b1 Burgers vector a/2[-1 $\sqrt{3}$ 0]

- Ltot_b2 Burgers vector [a 0 0]
- Ltot_b3 Burgers vector a/2[-1 $-\sqrt{3}$ 0]
- Ltot_b4 Burgers vector ½[-a a$*\sqrt{3}$ 2c]
- Ltot_b5 Burgers vector [a 0 c]
- Ltot_b6 Burgers vector ½[a a$*\sqrt{3}$ -2c]
- Ltot_b7 Burgers vector ½[-a a$*\sqrt{3}$ -2c]
- Ltot_b8 Burgers vector [-a 0 c]
- Ltot_b9 Burgers vector -½[a a$*\sqrt{3}$ 2c]
- Ltot_b10 Burgers vector [0 0 c]

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

Note: the 10th burgers vector has only 3 associated glide planes; hence column 7 of Ltot_b10 will always be zero.

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

The second set of files are fluxtot_b1 through fluxtot_b10, 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 component on 4 planes (columns 4 thru 7), and the flux from screw components on the four planes (columns 8 thru 11) differ between the files as seen

in the table below. The edge direction in all cases is determined by b X n.

Note: the 10th burgers vector has only 3 associated glide planes; hence columns 7 and 11 of fluxtot_b10 will always be zero

Column	Description
1	Plastic strain
2	Strain
3	Flux due to climb
12	Simulation timestep duration
For burgers vector b1 = a/2[-1 $\sqrt{3}$ 0]	
4	Plane (0 0 1)
5	Plane ($\sqrt{3}$ 1 0)
6	Plane (c*$\sqrt{3}$ c -a*$\sqrt{3}$)
7	Plane (c*$\sqrt{3}$ c a*$\sqrt{3}$)
8	Plane (0 0 c)
9	Plane ($\sqrt{3}$ 1 0)
10	Plane (c*$\sqrt{3}$ c -a*$\sqrt{3}$)
11	Plane (c*$\sqrt{3}$ c a*$\sqrt{3}$)
For burgers vector b2 = [a 0 0]	
4	Plane (0 0 1)
5	Plane (0 1 0)
6	Plane (0 2c a*$\sqrt{3}$)
7	Plane (0 -2c a*$\sqrt{3}$)
8	Plane (0 0 1)
9	Plane (0 1 0)
10	Plane (0 2c a*$\sqrt{3}$)
11	Plane (0 -2c a*$\sqrt{3}$)
For burgers vector b3 = a/2[1 $\sqrt{3}$ 0]	
4	Plane (0 0 1)
5	Plane ($\sqrt{3}$ -1 0)
6	Plane (c*$\sqrt{3}$ -c a*$\sqrt{3}$)
7	Plane (-c*$\sqrt{3}$ c a*$\sqrt{3}$)
8	Plane (0 0 1)
9	Plane ($\sqrt{3}$ -1 0)
10	Plane (c*$\sqrt{3}$ -c a*$\sqrt{3}$)

Column	Description
11	Plane (-c*√3 c a*√3)
For burgers vector b4 = ½[-a a*√3 2c]	
4	Plane (c -c*√3 2a)
5	Plane (√3 1 0)
6	Plane (0 -2c a*√3)
7	Plane (c*√3 -c a*√3)
8	Plane (c -c*√3 2a)
9	Plane (√3 1 0)
10	Plane (0 -2c a*√3)
11	Plane (c*√3 -c a*√3)
For burgers vector b5 = [a 0 c]	
4	Plane (c 0 a)
5	Plane (0 1 0)
6	Plane (-c*√3 c a*√3)
7	Plane (c*√3 c a*√3)
8	Plane (c 0 a)
9	Plane (0 1 0)
10	Plane (-c*√3 c a*√3)
11	Plane (c*√3 c a*√3)
For burgers vector b6 = ½[a a*√3 -2c]	
4	Plane (c c*√3 2a)
5	Plane (a*√3 -a 0)
6	Plane (c*√3 c a*√3)
7	Plane (0 2c a*√3)
8	Plane (c c*√3 2a)
9	Plane (a*√3 -a 0)
10	Plane (c*√3 c a*√3)
11	Plane (0 2c a*√3)
For burgers vector b7 = ½[-a a*√3 -2c]	
4	Plane (-c c*√3 -2a)
5	Plane (√3 1 0)
6	Plane (0 2c a*√3)
7	Plane (-c*√3 c a*√3)
8	Plane (-c c*√3 -2a)
9	Plane (√3 1 0)

Column	Description
10	Plane (0 2c a*√3)
11	Plane (-c*√3 c a*√3)
For burgers vector b8 = [-a 0 c]	
4	Plane (c 0 a)
5	Plane (0 1 0)
6	Plane (c*√3 c a*√3)
7	Plane (c*√3 -c a*√3)
8	Plane (c 0 a)
9	Plane (0 1 0)
10	Plane (c*√3 c a*√3)
11	Plane (c*√3 -c a*√3)
For burgers vector b9 = -½[a a*√3 2c]	
4	Plane (c c*√3 -2a)
5	Plane (√3 -1 0)
6	Plane (0 -2c a*√3)
7	Plane (c*√3 c -a*√3)
8	Plane (c c*√3 -2a)
9	Plane (√3 -1 0)
10	Plane (0 -2c a*√3)
11	Plane (c*√3 c -a*√3)
For burgers vector b10 = [0 0 c]	
4	Plane (√3 1 0)
5	Plane (√3 -1 0)
6	Plane (0 1 0)
7	Zero: This burgers vector has only three planes
8	Plane (√3 1 0)
9	Plane (√3 -1 0)
10	Plane (0 1 0)
11	Zero: This burgers vector has only three planes

14.4.4 Rhombohedral Vanadium Flux Decomposition

The flux decomposition data for rhombohedral vanadium 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 [1.45 1.45 1.45]
- Ltot_b2 Burgers vector [-1.1870 1.3185 1.3185]
- Ltot_b3 Burgers vector [1.3185 -1.1870 1.3185]
- Ltot_b4 Burgers vector [1.3185 1.3185 -1.1870]

Each of the burgers vectors has three associated glide planes for which flux data is defined. These specific glide planes are:

- Burgers vector 1
  1. (-1 1 0)
  2. (1 0 -1)
  3. (0 -1 1)
- Burgers vector 2
  1. (0 -1 1)
  2. (10 -1 10)
  3. (-10 -10 1)
- Burgers vector 3
  1. (1 0 -1)
  2. (1 -10 -10)
  3. (-10 -10 1)
- Burgers vector 4
  1. (-1 1 0)
  2. (1 -10 -10)
  3. (10 -1 10)

The format of the contents of files Ltot_b1 thru Ltot_b4 is the same, and consists of lines containing data as specified in following table.

Column	Description
1	Plastic strain
2	Strain
3	Screw density for plane 1

4	75 degree (trench) density for plane 1
5	Screw density for plane 2
6	75 degree (trench) density for plane 2
7	Screw density for plane 3
8	75 degree (trench) density for plane 3
9	Simulation timestep duration

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 is the same, and consists of lines containing data as specified in the table below.

Column	Description
1	Plastic strain
2	Strain
3	Climb flux for plane 1
4	Screw flux for plane 1
5	75 degree trench flux for plane 1
6	Climb flux for plane 2
7	Screw flux for plane 2
8	75 degree trench flux for plane 2
9	Climb flux for plane 3
10	Screw flux for plane 3
11	75 degree trench flux for plane 3
12	Simulation timestep duration

14.5 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)".

14.6 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)".

14.7 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)

14.8 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.

15 Utilities

15.1 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.

15.2 Creating Eshelby Inclusions with Inclusiongen

The 'inclusiongen' utility is used to create a set of randomly positioned Eshelby inclusions which can be inserted into ParaDiS simulations. Via the command line options, the user can control the number and size of inclusions, the mix of periodic and free surfaces, whether inclusions are permitted to overlap, and so on. For a description of the command line options and default behavior of the utility, execute 'inclusiongen –help'. After using inclusiongen to create a file of inclusion definitions, the inclusions can be inserted into a ParaDiS simulation by providing the inclusion file name to the simulation via the <inclusionFile> control file parameter.

15.3

15.4 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.

15.5 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.

15.5.1 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
```

15.6 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. When ParaDiS has been compiled with support for anisotropic elasticity, additional dependencies are imposed.

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 for an isotropic simulation, 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 as the parallel program initiator, one could execute:

```
mpirun -np 8 bin/ctablegenp -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 ctablegen creates correction tables for use with isotropic FMM. When creating a correction table for use with anisotropic FMM, additional command line arguments are required. First the "-aniso" option is required to

specify that the table is being built for anisotropic FMM, and the "-derivfile" option must be specified to provide the name of a file containing the Green's function derivatives needed to create the correction table. The derivatives file can be created using the "gen_fmm_derivatives" utility described in this document.

Building on the sample command above for creating the correction table, to generate a corresponding table for anisotropic FMM, something like the following command could be used:

```
bin/ctablegen –a –nu 3.327533e-01 –mu 6.488424e+10 –
mporder 2 –torder 5 –derivfile inputs/fmm-derivatives.dat
–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.

15.7 Creating FMM Derivatives Files with gen_fmm_derivatives

The 'gen_fmm_derivatives' tool is used to create a file containing Green's function derivatives that are needed both by 'ctablegen' and ParaDiS when anisotropic FMM calculations are to be done. The resulting file name is provided as a command-line input to ctablegen and via the <fmAnisoDerivTbl> control-file parameter to ParaDiS. When support for anisotropic elasticity has not been compiled into ParaDiS, this utility serves no purpose.

Since the derivatives in this file are dependent on the type of material (i.e. bcc, fcc, etc), various elastic constants, the multipole expansion order and so on, the file must be created for the particular values of these items needed in the simulation.

For detailed information on the command-line options and how to use this utility, execute 'gen_fmm_derivatives –help'.

15.8 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'

15.9 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'.

16 Tools

16.1 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
```

16.2 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
```

16.3 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
- Force data files
- Pole figure 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.

17 Appendix

17.1 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. If <timeEnd> has also been specified and the maximum number of steps have been executed, this parameter takes precedence over <timeEnd>. Default is 100.
timeNow	Double	Current simulation time (in seconds).
timeStart	Double	Initial simulation time (in seconds).
timeEnd	Double	Absolute simulation time (in seconds) at which to terminate the simulation. If the specified simulation time is reached before <maxStep> cycles have been simulated, this parameter takes precedence over <maxStep>. Default is -1.0 indicating no specified end time.
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>.
trapezoidMaxIterations	Integer	Maximum number of loop iterations the timestep integrator will attempt to converge on a solution before cutting the timestep and trying again. Only applicable to the "trapezoid" timestep integrator. Value must be greater than 0. Default value is 2.
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>.
KINSOL_UseNewton	Integer	Toggle enabling/disabling the Newton-Krylov solver in KINSOL based timestep integrator. 0 == disabled, 1 == enabled. Default is zero. This applies only to the 'trapezoid-kinsol' timestep integrator.
KINSOL_MaxIterations	Integer	Maximum number of non-linear solver iterations are

		attempted by the 'trapezoid-kinsol' timestep integrator before cutting the timestep. This applies only to the 'trapezoid-kinsol' timestep integrator. Default is 3.
KINSOL_MaxLinIterations	Integer	Maximum number of linear iterations used within KINSOL's Newton-Krylov solver. Only applicable when the <KINSOL_UseNewton> is set. Default = 0 (which internally set the number of linear iterations to the KINSOL default: currently 5)
KINSOL_NumPriorResiduals	Integer	The number of prior residual vectors used with the Anderson Acceleration within the KINSOL based timestep integrator. This applies only when the Anderson accelerated fixed-point solver is in use. Default is <KINSOL_MaxIterations> - 1.
KINSOL_EtaVal	Double	Constant in stopping criteria of linear solve within KINSOL based timestep integrator. Only applicable when Newton-Krylov solver is selected. Permitted range is 0.0-1.0. Default is 0.2. A value of zero set the nonlinear residual factor to the KINSOL default of 0.1. See "Stopping criteria for linear solvers" in the Mathematical Considerations portion of the KINSOL user guide for more info.
KINSOL_PrintLevel	Integer	Specifies the level of verbosity in the output from KINSOL. This applies only to the 'trapezoid-kinsol' timestep integrator. Permitted range is 0 – 3 where 3 is most verbose. Default is 3. (If <KINSOL_LogFile> is not provided, no output will be written.

KINSOL_LogFile	String	Name of the file to which the KINSOL solver writes its debug information. This applies only to the 'trapezoid-kinsol' timestep integrator. Default string is NULL.
ARKODE_FPsolver	Integer	Toggle enabling/disabling the fixed-point solver in the ARKode timestep integrator. 0 = Newton solver, 1 = fixed-point solver. Default = 0.
ARKODE_FPaccel	Integer	Number of acceleration vectors used with the accelerated fixed-point solver within the ARKode timestep integrator. Default = 0 (no acceleration).
ARKODE_MaxLinIters	Integer	Maximum number of linear iterations used with Newton method within ARKode timestep integrator. Only applicable when <ARKODE_FPsolver> == 1. Default = 0. (A value of zero set the number of iterations to the ARKode default: currently 5.)
ARKODE_MaxNonLinIters	Integer	Max number of nonlinear iterations used in the ARKode timestep integrator. Default = 0. (A value of zero sets the number of iterations to the ARKode defaults: currently 3 for Newton and 10 for fixed-point iterations.)
ARKODE_MaxConvFailGrowth	Double	Factor by which to adjust timestep after convergence failure in solver. Default = 0.5
ARKODE_LinCoef	Double	Factor relating the nonlinear solver tolerance and the inner GMRES solver tolerance. The smaller the value, the closer the Inexact Newton will be to having quadratic convergence. Only applicable the Newton solver in the ARKode timestep

		integrator. Permitted range 0.0-1.0. Default = 0.05
ARKODE_NonLinCoef	Double	This factor relates the estimated temporal error with the desired nonlinear solver tolerance, allowing one to solve each implicit problem to an accuracy greater than that of the estimated temporal error. Applicable to the ARKode timestep integrator only. Permitted range 0.0 — 1.0. Default = 0.1
ARKODE_IRKtable	Integer	Selects the implicit Runge-Kutta method used by the ARKode timestep integrator. Default = 12 (3rd order diagonally implicit RK method by Billington). See ARKode documentation for available methods.
ARKODE_PredictorMethod	Integer	Predictor method for non-linear solves within the ARKode timestep integrator. Default = 0 (trivial predictor - previous step solution). See ARKode documentation for available methods.
ARKODE_AdaptivityMethod	Integer	After a successful time step, ARKode has an estimate the temporal error that it uses to predict the optimal subsequent time step size. This parameter selects which method is used there. Default is 1 (PI step controller which uses the 2 most recent step sizes for predicting the next step). See ARKode documentation for available methods.
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.
rann	Double	Annihilation distance (in units of b). Defines the closest distance before dislocations are considered touching. Used in collision handling, rediscretization, multi-node splitting, etc. Default is 2.0 * <rTol>.
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.
useParallelSplitMultiNode	Integer	Toggle indicating whether to enable use of the optional 'parallel' multi-node splitting mechanism. Default is 0 (off).
useUniformJunctionLen	Integer	Toggle indicating whether junctions formed during SplitMultiNode() are created with a uniform length (currently 4.0 *<rTol>) or whether the code attempts to calculate the optimal length

		for each junction. Default is 0 (off).
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.
fmEnableAnisotropy	Integer	Flag to enable/disable anisotropic FMM. Default is 0. Additionally, if general anisotropy is not enabled via the _ANISOTROPIC compile-time definition, this flag will explicitly be turned off.
eshelbyfmEnabled	Integer	Toggle controlling use of a fast Multipole Method (FMM) for computing force contributions from remote eshelby inclusions. Any value other than zero

		enables the eshelby 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.
fmAnisoDerivTbl	String	Name of the file containing the Green's function derivatives needed when the <fmEnableAnisotropy> flag is turned on. Default name is "fm-aniso-deriv-table.dat"
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.
eshelbyfmMPOrder	Integer	Defines the multipole expansion order used by the eshelby FMM. This value is ignored if <eshelbyfmEnabled> is zero. Default value is 3.
eshelbyfmTaylorOrder	Integer	Defines the order of the taylor expansions used by the eshelby FMM. This value is ignored if <eshelbyfmEnabled> is zero. Default value is 3.
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)
Pressure	Double	Simulation pressure (in units of Pa)
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: 0-2: uniaxial (default 0) 3-5: shear loading More specifically, 0 indicates load in the x direction and similarly:1-yy, 2-zz,3-yz,4-xz,and 5-xy.
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: Unless otherwise indicated, default values for all material and mobility related parameters correspond to Tantalum at a temperature of 600(K) and a pressure of 0GPa.		
ecFile	String	Base name for a set of files containing material specific elastic constants at various temperature and pressure combinations. If specified, the shear modulus, poisson ratio and burgers vector magnitude will be calculated for the selected temperature and pressure from the data in the <ecFile> file set. Data file names will be of the form: <ecFile>.c11.dat <ecFile>.c12.dat <ecFile>.c44.dat> <ecFile>.burg.dat
meltTempFile	String	Name of a file containing a table specifying the material melting temperature over a range of pressures. No default.
MobFileNL	String	Name of a file containing values for <MobAlpha>, <MobDeltaH0>, and MobPieirls> over a range of temperatures/pressures. No default.

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)
materialTypeName	String	Specifies the type of material (“BCC”, “FCC”, “HCP”, or “RHOMBO_VA) being simulated. If unspecified, this will take on the default type associated with the mobility specified in the <mobilityLaw> parameter.
C11	Double	Elastic constant. Only applies when anisotropic elasticity is used for HCP simulations. No default and must be provided if <elasticConstantMatrix> is not specified. Must be specified in the laboratory frame if <useLabFrame> is 1 or the crystallographic frame otherwise.
C12	Double	Elastic constant. Only applies when anisotropic elasticity is used for BCC, FCC or HCP simulations. No default and must be provided if <elasticConstantMatrix> is not specified. Must be specified in the laboratory frame if <useLabFrame> is 1 or the crystallographic frame otherwise.
C13	Double	Elastic constant. Only applies when anisotropic elasticity is used for BCC, FCC or HCP simulations. No default and must be provided if <elasticConstantMatrix> is not specified. Must be specified in the laboratory frame if <useLabFrame> is 1 or the crystallographic frame otherwise.
C33	Double	Elastic constant. Only applies when anisotropic elasticity is used for HCP simulations. No default and

		must be provided if <elasticConstantMatrix> is not specified. Must be specified in the laboratory frame if <useLabFrame> is 1 or the crystallographic frame otherwise.
C44	Double	Elastic constant. Only applies when anisotropic elasticity is used for BCC, FCC, or HCP simulations. No default and must be provided if <elasticConstantMatrix> is not specified. Must be specified in the laboratory frame if <useLabFrame> is 1 or the crystallographic frame otherwise.
elasticConstantMatrix	Double[6][6]	Full elastic constant matrix for the material. Only applies when anisotropic elasticity is used. For rhombohedral crystals, this is required. For BCC, FCC and HCP materials, either this matrix or the individual elastic constants must be provided. Must be specified in the laboratory frame if <useLabFrame> is 1 or the crystallographic frame otherwise. (If both the matrix and indivual constants are provided, the matrix takes precedence.)
anisoHarmonicsNumTermsBase	Integer	Expansion order. Number of terms taken in the spherical expansion series. The series contains an infinite number of terms, this parameter sets the truncature. Permitted value is >= 0 and <= 20. Default is 3. See more details in the XXXXX paper about how the expansion order should be set for a given anisotropy ratio and error in stress/force calculations.
anisoNumPhiPoints	Integer	Half-sphere number of angles (theta, phi) in which the expansion coefficients of the angular part of the

		Green's function are defined.
anisoNumThetaPoints	Integer	Half-sphere number of angles (theta, phi) in which the expansion coefficients of the angular part of the Green's function are defined.
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.
meltTemp	Double	Material specific melting temperature in degrees K. Ignored if <meltTempFile> has been specified. Default is 3.170683e+03. Only used with certain mobility functions.
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.
cOVERa	Double	The hexagonal crystal lattice is often represented by four Miller indices. This is not convenient in the current implementation of ParaDiS so a different system of axes has been used to represent HCP crystal slip systems. This system is: a_1 = a ex a_2 = - ½ a ex + $\sqrt{3}$/2 a ey a_3 = - ½ a ex - $\sqrt{3}$/2 a ey The axial relationships for HCP lattices are different than those of BCC and FCC crystals. In BCC and FCC crystals, lattices are such that the axial relationship for the cell geometry is a = b = c and the inter-axial angles are alpha = beta = gamma = 90. In HCP crystals, the lattice is such that a = b != c, and alpha = beta = 90 and gamma = 120. The change of coordinate system takes care of the gamma angle so that in the new system angle(a1, a2) = angle(a1, a3) = angle(a2, a3) = 90. <cOVERa> = c/a has been introduced to insure the axes are normalized. By definition, Burgers vectors and planes in the HCP lattice belong to the set of axes (a_1, a_2, a_3) which means they will always be a function of (a a c) or (1 1 c/a). Applies only to simulations of HCP crystals. Beryllium = 1.5680 (Default)

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 prefactor used in self-force calculations. For isotropic simulations the default is (<shearModulus>/(4*PI))* log(<rc>/0.1). For anisotropic simulations the default is log(<rc>/0.1) NOTE: for HCP simulations this value is replaced by the parameters HCPEcoreA, HCPEcoreC, and HCPEcoreCpA described below.
For HCP materials only, there are three core energy prefactors (HCPEcoreA, HCPEcoreC, HCPEcoreCpA), each of which applies to a specific type of burgers vectors. These prefactors are dependent on the core radius <rc> and are in units of shear modulus (energy per unit length / b^2 units). These prefactors replace the value of <Ecore>. See the file MD_Ecore.pdf in the docs directory for a description of how these prefactors are calculated.		
HCPEcoreA	Double	Core energy prefactor associated with <a> type burgers vectors. Applicable only to HCP simulations. Default of 18.06977e9 is for beryllium using <rc>=3.
HCPEcoreC	Double	Core energy prefactor associated with <c> type burgers vectors. Applicable only to HCP simulations. Default of 21.70031e9 is for beryllium using <rc>=3.
HCPEcoreCpA	Double	Core energy prefactor associated with <c+a> type burgers vectors. Applicable

		only to HCP simulations. Default of 7.18520e9 is for beryllium using <rc>=3.
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.
MobLine	Double	Mobility of dislocations in the direction of the dislocation line in units of 1/(Pa*sec). Default is 1.0e+04. Not applicable to all mobility functions.
MobGlide	Double	Glide mobility of edge dislocations in units of 1/(Pa*sec). Default is 1.0e+02. Not applicable to all mobility functions.
MobDeltaH0	Double	Screw dislocation kink pair formation energy in units of joules. Default is 1.728748e-19. Not applicable to all mobility functions.
MobPieirls	Double	Screw dislocation pieirls stress in units of Pa. Default is 3.24e+08. Not applicable to all mobility functions.
MobExpP	Double	One of two exponents in the kink pair activation enthalpy. Default is 0.5. (See document 'NonLinearMobility.pdf' for more information) Not applicable to all mobility functions.
MobExpQ	Double	One of two exponents in the kink pair activation enthalpy. Default is 1.23. (See document

		'NonLinearMobility.pdf' for more information) Not applicable to all mobility functions.
MobAlpha	Double	Ratio of the transition stress from kink pair to phonon drag for screw dislocation mobility versus pieirls stress. (i.e. transStress = MobAlpha * MobPieirls) (See document 'NonLinearMobility.pdf' for more information) Not applicable to all mobility functions.
MobCoeffA	Double	Fitted constant for screw dislocation mobility in phonon drag regime. Default is 1.525e+0. (See document 'NonLinearMobility.pdf' for more information) Not applicable to all mobility functions.
MobCoeffC	Double	Fitted constant for screw dislocation mobility in phonon drag regime. Default is 2.03237e-1. (See document 'NonLinearMobility.pdf' for more information) Not applicable to all mobility functions.
MobCoeffC0	Double	Fitted constant for screw dislocation mobility in phonon drag regime in units of meter/sec. Default is 1.62695e-4. (See document 'NonLinearMobility.pdf' for more information) Not applicable to all mobility functions
MobG0	Double	Not applicable to all mobility functions.
MobNumTrenches	Integer	Number of angular windows within which drag on dislocations is increased. Defines the number of valid elements in the trench related arrays <MobTrenchAngle>, <MobTrenchWidth>, and <MobTrench>. Default is

		zero. Not applicable to all mobility functions.
MobTrenchAngle	Double[10]	Centers (specified in angles) of the corresponding windows (trenches) of reduced mobility. Array always contains 10 elements, but only the first <MobNumTrenches> values are used. Each angle is associated with the corresponding elements of the <MobTrenchWidth> and <MobTrench> arrays. Not applicable to all mobility functions.
MobTrenchWidth	Double[10	Half the angular period (in degrees) of the corresponding windows (trenches) of reduced mobility. Array always contains 10 elements, but only the first <MobNumTrenches> values are used. Each angle is associated with the corresponding elements of the <MobTrenchAngle> and <MobTrench> arrays. Not applicable to all mobility functions.
MobTrench	Double[10]	Decrease in mobility (in units of 1/(Pa*sec)) along the corresponding trenches. Array always contains 10 elements, but only the first <MobNumTrenches> values are used. Each angle is associated with the corresponding elements of the <MobTrenchAngle> and <MobTrenchWidth> arrays. Not applicable to all mobility functions.
MobFacetedUseCusps	Integer	Toggle used with the "BCC_faceted" mobility module to enable the "cusp" variant of that mobility. If this toggle is enabled, the trench angles defined in <MobTrenchAngle> are not used. Default is 0 (off).

The following drag coefficients for HCP mobility functions are in units of Pa*s and all correspond to Beryllium at ambient temperature.		
HCP_A_Basal_EdgeDrag	Double	Drag coefficient for <a> type edge dislocations in the basal plane. Only applicable to HCP materials. Default is 1.1843e-04.
HCP_A_Basal_ScrewDrag	Double	Drag coefficient for <a> type screw dislocations in the basal plane. Only applicable to HCP materials. Default is 1.6542e-04.
HCP_A_Prismatic_EdgeDrag	Double	Drag coefficient for <a> type edge dislocations in the prismatic plane. Only applicable to HCP materials. Default is 5.6951e-05.
HCP_A_Prismatic_ScrewDrag	Double	Drag coefficient for <a> type screw dislocations in the prismatic plane. Only applicable to HCP materials. Default is 7.1509e-05.
HCP_A_1stPyramidal_EdgeDrag	Double	Drag coefficient for <a> type edge dislocations in the 1st pyramidal plane. Only applicable to HCP materials. Default is 7.2826e-05.
HCP_A_1stPyramidal_ScrewDrag	Double	Drag coefficient for <a> type screw dislocations in the 2nd pyramidal plane. Only applicable to HCP materials. Default is 1.0515e-04.
HCP_A_2ndPyramidal_EdgeDrag	Double	Drag coefficient for <a> type edge dislocations in the 2nd pyramidal plane. Only applicable to HCP materials. Default is 7.2826e-05.
HCP_A_2ndPyramidal_ScrewDrag	Double	Drag coefficient for <a> type screw dislocations in the 2nd pyramidal plane. Only applicable to HCP materials. Default is 1.0515e-04.
HCP_CpA_Prismatic_EdgeDrag	Double	Drag coefficient for <c+a> type edge dislocations in the prismatic plane. Only

		applicable to HCP materials. Default is 2.3686e-03.
HCP_CpA_Prismatic_ScrewDrag	Double	Drag coefficient for <c+a> type screw dislocations in the prismatic plane. Only applicable to HCP materials. Default is 2.3686e-03.
HCP_CpA_1stPyramidal_EdgeDrag	Double	Drag coefficient for <c+a> type edge dislocations in the 1st pyramidal plane. Only applicable to HCP materials. Default is 2.3686e-03.
HCP_CpA_1stPyramidal_ScrewDrag	Double	Drag coefficient for <c+a> type screw dislocations in the 1st pyramidal plane. Only applicable to HCP materials. Default is 2.3686e-03.
HCP_CpA_2ndPyramidal_EdgeDrag	Double	Drag coefficient for <c+a> type edge dislocations in the 2nd pyramidal plane. Only applicable to HCP materials. Default is 2.3686e-03.
HCP_CpA_2ndPyramidal_ScrewDrag	Double	Drag coefficient for <c+a> type screw dislocations in the 2nd pyramidal plane. Only applicable to HCP materials. Default is 2.3686e-03.
HCP_C_Prismatic_EdgeDrag	Double	Drag coefficient for <c> type edge dislocations in the prismatic plane. Only applicable to HCP materials. Default is 2.3686e-03.
HCP_C_Prismatic_ScrewDrag	Double	Drag coefficient for <c> type screw dislocations in the prismatic plane. Only applicable to HCP materials. Default is 2.3686e-03.
HCP_Sessile_EdgeDrag	Double	Drag coefficient for sessile edge dislocations. Only applicable to HCP materials. Default is 23.6868.
HCP_Sessile_ScrewDrag	Double	Drag coefficient for sessile screw dislocations. Only applicable to HCP materials. Default is 23.6868.
HCP_LineDrag	Double	Drag coefficient for dislocations in the line direction. Only applicable

		to HCP materials. Default is 5.0e-06.
MobRelaxX	Double	Used in velocity calculation for the relaxation mobility modules. Defines component M_{11} of matrix M in V=dot(M,F) calculation in units of 1/(Pa*sec).
MobRelaxY	Double	Used in velocity calculation for the relaxation mobility modules. Defines component M_{22} of matrix M in V=dot(M,F) calculation in units of 1/(Pa*sec).
MobRelaxZ	Double	Used in velocity calculation for the relaxation mobility modules. Defines component M_{33} of matrix M in V=dot(M,F) calculation in units of 1/(Pa*sec).
MobRelaxScaleByLength	Integer	Toggle to enable velocity scaling by dislocation length in 'relaxation' mobility modules. Default is 1 (on).
sessileburgspec	Double[30	Array of burgers vectors to be considered sessile. This array MUST be exactly 30 elements in length. First element of the array contains the number of burgers vectors specified and the remaining elements specify the X, Y and Z components of the sessile burgers vectors. Maximum sessile burgers vectors allowed is 9. No burgers vectors are sessile by default. Note: the 30 element array will have spare values that must be included but will be ignored.
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. This array MUST be exactly 30 elements in length. The 30 element array will have spare values that must be included but will be ignored.
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 spin tensor
totpStn	Double[6]	Plastic strain 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).
HCP_Ltot	Double[10][5]	Decomposed density per burgers vector for screw and four edges (for HCP slip systems only).
rhombo_Ltot	Double[4][3][2]	Decomposed density per burgers vector (rhombohedral vanadium only). 4 burgers vectors 3 slip planes per burgers vector 2 types of density info per slip plane - density for screw dislocations

		- density for 75 degree dislocations
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).
HCP_fluxtot	Double[10][9]	For each burgers vector, contains: 1: flux due to climb 2-5: flux due to edge components 6-9: flux due to screw components Exception: For 10th burgers vector which has only 3 glide planes: 2-4: flux due to edge components 5-7: flux due to screw components 8-9: zero (for HCP slip systems only)
rhombo_fluxtot	Double[4][3][3]	Flux per burgers vector (rhombohedral vanadium only). 4 burgers vectors 3 slip planes per burgers vector 3 flux values per slip plane: 1: climb flux 2: flux for screw 3: flux for 75 degree
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.
saveproptime	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).
writeVisitForceVector	Integer	Enables/disables writing the nodal force vectors to the VisIt node files if both the <writeVisit> and <writeVisitNodes> toggles are enabled. Default is 0 (off).
writeVisitVelocityVector	Integer	Enables/disables writing the nodal velocity vectors to the VisIt node files if both the <writeVisit> and <writeVisitNodes> toggles are enabled. 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.
writeVisitBurgID	Integer	Toggle indicating if the segment data should include an integer burgID value identifying the segment's specific type or group of burgers vector. Default is zero (off). This value is ignored if <writeVisitSegments> is not enabled.
writeForce	Integer	Toggle enabling/disabling generation of output files containing nodal force data. Default is zero (off).
writeForceCounter	Integer	See description of <*counter> above.
writeForceDT	Double	See description of <*dt> above.
writeForceFreq	Integer	See description of <*freq> above.
writeForceTime	Double	See description of <*time> above.
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 in calculating core energy for simple line tension force when <elasticinteraction> is set to zero. The core energy value is calculated as: 0.5 * TensionFactor * MU Default value is 1.0.

		This parameter does not apply when using anisotropic forces.
inclusionFile	String	Name of the file specifying Eshelby inclusions to be inserted into the simulation. Default is an empty string indicating no inclusions are to be simulated.