User Manual

Version 11Dec2023

Levi Felix

December 11, 2023

Contents

1 Introduction

This document is a manual to the JC++ code to compute a fully lattice-based *J*-integral from atomistic simulations.

2 Installation

The installation is as simple as compiling a `C++` code on a Linux terminal. From the parent directory (`jcpp`), do the following steps:

```
cd src
./install.sh
```

An executable file called `jc++` should be created in the `src` directory. You also have the option to export the folder to the `PATH` environment variable.

3 Input file description

In this section, a detailed description of the input file parameters is given. *J* is calculated done by a `.jparams` control file that specifies the parameters of the calculation, such as `LAMMPS` trajectory files, loop sizes, timesteps used in the calculation, etc.

SystemLabel [string]

It is used as a prefix to name the result's folder.

Dimensions [integer]

System dimensionality (e.g. 2 or 3). Necessary when dealing with 2D systems. The computation of the deformation gradient tensor *F* needs that all atoms should have a coordination number greater or equal to `Dimensions` to avoid singular matrix problems [1].

PBC [boolean]

Periodic boundary conditions out-of-plane?

NumTypes [integer]

Number of atom types in the simulation.

BulkCoordination [integer array]

The coordination number of each atom type in the bulk region. This is used to search for the crack tip position, where atoms nearby have lower atomic coordination.

NeighborCutoff [float array]

Pairwise cutoff radii to determine atom neighbors. If more atom types are present, provide cutoff radii for each interaction. For example, if a simulation has 2 types, provide $2 \times 2 = 4$ numbers separated by spaces. Each element will be part of a square matrix of `NumTypes`^{2} in the following order: r_{11} r_{12} r_{21} r_{22}. This is very system-dependent. Coordination Analysis tool from OVITO may help to get suitable values.

NeighborScaling [float array]

Scaling factor for cutoff radii for the calculation of F. The need for these are to avoid singular matrices due to atoms with coordination number less than the system's dimensionality. It is a multiplicative factor for the respective cutoff radius. It has the same format as the `NeighborCutoff` keyword.

NeighborSkin [float]

Avoids having to loop over all atoms in the simulation to get neighbor lists.

Thickness [float]

Out-of-plane length of simulation box for 3D problems with PBC. Chunk (3D) or membrane (2D) thickness otherwise.

UnitsLAMMPS [string]

Units used in the `LAMMPS` simulation to convert all units to yield J in J/m^2.

CrackTrajFile [string]

Path to simulation trajectory file of the crack propagation simulation.

BulkTrajFile [string]

Path to simulation trajectory file of the simulation containing the atomic potential energy and stresses reference values from the bulk configuration. If the crack faces have their interactions excluded and the loading simulation starts from a completely "closed" crack, then the same trajectory file can be used.

SurfaceTrajFile [string]

Path to simulation trajectory file of the simulation containing the atomic potential energy and stresses reference values from the surface configuration. If the crack faces have their interactions excluded and the loading simulation starts from a completely "closed" crack, then the same trajectory file can be used.

CSVrestart [boolean]

Weather generates new data from `LAMMPS` or restarted from pre-existing CSV files. If no previous CSV files are present, set to `False`.

StepMin [integer]

Minimum timestep to compute *J*. Only used if `CrackStep=-1`.

StepMax [integer]

Maximum timestep to compute *J*. Only used if `CrackStep=-1`.

CrackStep [integer]

Timestep at which *J* should be computed. If -1, compute all timesteps between `StepMin` and `StepMax`

BulkStep [integer]

Timestep at which the reference "bulk" structure is taken from the trajectory of a bulk perfect structure used to get energy and stress reference values for the atoms in the system [2].

SurfaceStep [integer]

The same but corresponding to atoms "behind" the crack tip, where reference values should be taken from a surface configuration [2].

ContourShape [string]

Shape of loop used in the *J*-integral calculation. Currently, only `circular` and `rectangular` are implemented.

ContourParams [float array]

Parameters giving the size of the loop. If `ContourShape=rectangular`, then the values given are L_x and L_y, which are the size of the edges of the rectangle along *x* and *y* axis, respectively. If `ContourShape=circular`, then the values given correspond to R_{in} and R_{out}, which are the inner and outer radius of the closed circular annulus, respectively.

DomainCSV [boolean]

Enable saving CSV files for the integration domains. It can be used to plot calculated quantities by the code. Default: `False`.

FindCrackTip [boolean]

Enable automatic crack tip search.

TipSearchMethod [string]

Currently, two methods are implemented. The first one is `Coordination`, where atoms with a coordination number less than the bulk values (e.g., in a honeycomb lattice, atoms have a coordination number of 3 in the bulk and 2 or 1 at the edges) are filtered and those with maximum *x* coordinate values (closer to the tip) will identify the crack tip, where its coordinates are

given by the average of these atoms. Note that this option may give wrong values if there is any structural defect within `CrackRegion`. The other one is `vonMises`, which looks for the atoms with the largest von Mises stress and get crack tip position from their positions.

TipSearchStyle [string]

If `TipSearchMethod=Coordination`, then two styles are currently available. The style `center`, which explicitly specifies the medium y (y_m) and its thickness Δy of the crack region (defined in the next keyword). The style `center`, then the range x_{lo} and x_{hi} are specified, where *lo hi* refers to lower (higher) bounds of the rectangle. If `TipSearchMethod=vonMises`, then two styles are currently available. The style `one` is given, then the crack tip position is given the (x,y) position of the atom with the largest von Mises stress. If the style is `two`, then the midpoint of the two atoms with the largest von Mises stress will be used as the crack tip position.

CrackRegion [float array]

Rectangular region where the crack tip should be in. When `FindCrackTip=False` this options is ignored. For `TipSearchStyle=center`, the format is x_{lo} x_{hi} y_m Δy. For all other cases, we have x_{lo} x_{hi} y_{lo} y_{hi}. All values are given in Å

CrackTipPos [float array]

Manually-defined coordinates of the crack tip position (x,y) are given in Å.

4 Running examples

The `examples` folder contains tests involving the main features of the code, to test if the compilation was done successfully.

4.1 Crack propagation in monolayer graphene

A crack propagation simulation in graphene is located in the `/examples/graphene/` folder. A precalculated test is located therein. To run it just open a terminal in that folder and type:

```
./run.sh
```

A new folder named `j_integral_graphene` will be created, which contains CSV files for each timestep of the simulations, alongside the *J*-integral calculated for each one of them in files with the `j_integral_*.txt` pattern.

5 Plotting tools

In the `/jcpp/tools/` folder, there are some Python scripts used to plot results and convert CSV to `LAMMPS` dump files in order to visualize quantities computed by the code, such as the deformation gradient tensor *F* and the atomic coordination, for a specific domain inside a loop used to compute *J*. NOTE: The `pandas` library is required to use these scripts!

References

[1] Jonathan A. Zimmerman, Douglas J. Bammann, and Huajian Gao. Deformation gradients for continuum mechanical analysis of atomistic simulations. *International Journal of Solids and Structures*, 46(2):238–253, 2009.

[2] Keiko Nakatani, Akihiro Nakatani, Yoshihiko Sugiyama, and Hiroshi Kitagawa. Molecular dynamics study on mechanical properties and fracture in amorphous metal. *AIAA Journal*, 38(4):695–701, 2000.