User Manual

Bridget Toomey, Chen Dan Dong, Verne Edward, and Xuchen Han

August 2013

This software takes as input a list of xyz coordinates of atomic positions. This list must adhere to the format of all the files we have received from the Intel Corporation thus far. Specifically, the software requires that all atoms of the same element type appear consecutively in the list. Additionally, the sixth line of the data file must contain a list of natural numbers. The n^{th} element in this list should specify the number of atoms of the n^{th} element type in the sample (that is, the n^{th} cluster in the list of atoms of the same type). The software's GUI lets the user choose which input file he or she would like to use. The user can specify which folder (s)he would like to save the data in, but the software will save the data in a default location otherwise. The software will save the output for each region in a separate text file in this folder.

The software stores a cell array, a Matlab data type that consists of indexed data containers known as cells. Each of these individual cells can contain any type of data, including other cell arrays. In the case of our program, each row in the cell array corresponds to one crystalline region of the data. (If there are no crystalline regions, the user can still choose to have the program output the radial distribution function for any subset of the data they wish to specify).

The first column of each row in the cell array serves as an index for each region. The second column lists all of the points in the interior of the region. In order to be designated as an interior point, an atom must be surrounded entirely by other atoms in the same region. The third column lists all of the points on the boundary of the region. All of the atoms in this region are adjacent to at least one atom in a different region. The fourth column combines all of the points in the second and third columns to yield all of the points in the entire region. The fifth column identifies the Bravais lattice type of the region. In the sixth column, the primitive vectors are given. The purpose of the seventh column is to aid in Miller index identification. Essentially, this column provides a "perfect projection" of the data. It accounts for slight imperfections in the data to create a perfect crystal structure from which the Miller index can be found more easily. The eighth column contains the total number of atoms in the whole region. The ninth column stores the starting point from which the projection of that region began. The tenth column provides the local coordinates used in Miller index determination. The eleventh column stores the radial distribution function for each region (see below for more information about RDF capabilities). The twelfth column contains the Miller index information for the region.

However, the information saved in the file corresponding to each region is represented somewhat differently. The first section in this file provides the number of atoms in the region, as well as the region's lattice type. The second section lists the interior points, and the third lists the points on the boundary. The fourth section combines the previous two sections to list all of the points in the region. The fifth section lists the local coordinates used in Miller index determination. The sixth section lists the Miller indices for each face of the region, followed by the boundary points of each face.

Additionally, the user can also choose to have the software output the radial distribution function (RDF) for different areas of the sample. Though the software automatically calculates the RDF for each crystalline region in the sample, the software only outputs the RDF for the entire sample by default. However, the user can utilize the GUI to specify which RDF he or she would like to view. The GUI lets the user choose whether he or she would like to view the RDF of one of the crystalline regions or another user-specified area of the sample. In all cases, the default option for dr (the step size between distance categories) is 0.01, the default for r_{min} (the minimum distance between two atoms) is 0, and the default for r_{max} is equal to the largest length between two atoms in the given area. If the area is a regular box, for computational efficiency, we find r_{max} by computing the length of the diagonal of the box. The default option for element pairings is to output the RDF for all possible element pairings in the given sample. We would like to caution the user that correct spelling and case are crucial if he or she chooses to output only one possible element pairing.

If the user chooses to output the RDF for one of the crystalline regions without inputting a region index, the default option is to compute the RDF of the first region in the sample. The user can also output the RDF for any box-shaped area of the sample. In this case, the default option for the center of the box is the point $(0, 0, 0)$, and the default dimensions for the box are $10 \times 10 \times 10$. Additionally, the user must choose whether he or she desires periodic boundary conditions to be implemented in this case.

We initially wrote the software using several default tolerances. However, as per Intel's request, we changed the code to allow the user to specify the tolerance in almost all cases. The first exception is the tolerance we implemented for Niggli reduction based on the results obtained by Grosse-Kuntleve et al. The second exception is the tolerance we used in defining a convex hull to check for the presence of points in the interlocking diamond structure case. The level we chose for this tolerance (0.0001 Angstroms) was selected with the aim of allowing us to find points that otherwise would have been on the boundary of this hull. The following paragraphs list the tolerances which the user can choose to specify, their purposes, and suggested default levels. At this point, we would like to caution the user against changing the tolerances from their default levels. We have not tested the software extensively with other tolerances, so we cannot guarantee its accuracy in these cases.

The first user-specified tolerance (tolerance 1 on the GUI) is used to ensure that the three choices of primitive vectors are not coplanar. Its default level is 3, and it is used in two separate types of scenarios in the code. In the first instance,

we compare the first two choices of primitive vectors to check whether they are collinear. Slight imperfections in the data might make two points which should be collinear slightly out of alignment. Thus, we say that if the angle between the two first choices of primitive vectors is less than the designated tolerance, the points we chose to generate these vectors are collinear. Similarly, this tolerance is also used to check whether the first three choices of primitive vectors are not coplanar by using geometric properties of the dot and cross product. Here, we fist define a variable α such that

$$\alpha = \frac{180}{\pi} * arccos(\frac{(x \times y) \cdot z}{\|(x \times y)\|\|z\|}).$$

If the difference $|90 - \alpha|$ is less than *tolerance 1*, the program labels these three vectors as coplanar and proceeds to find alternate choices for the primitive vectors.

The second user-specified tolerance is used in several instances throughout the code to check whether the actual data points match the points predicted by the current projection. The default level for this tolerance is 0.01. We chose this level based on empirical evidence which seems to show that it is an ideal balance. That is, it is neither leads to us matching points incorrectly nor leads to us missing points that should be matched. *Tolerance 3* is used when determining Bravais lattice type using the Niggli reduction algorithm. Before implementing this tolerance, we noticed that the reduced Niggli cell we obtained would often not quite match the strict requirements for correct lattice type identification. For example, slight imperfections in the data in conjunction with rounding errors inherent in floating point arithmetic might lead to a cubic cell with reduced Niggli vectors A, B, and C such that $A = B + .001 = C - .003$. However, the technical definition of a cubic unit cell requires strict equality among A, B, and C. Our default level for tolerance 3 is 0.01.

The fourth tolerance is used when determining whether the next point in our projection lies on the boundary of a crystalline region. We compare the volume of the unit cell generated with the primitive vectors found from the starting point to the volume of the unit cell generated with the primitive vectors found from the current point which we are checking. If these two volumes differ by more than *tolerance 4*, then the program determines that the current point lies on the region's boundary. The default level for this tolerance is one-tenth the volume of the current unit cell. The fifth and final tolerance level is used in Miller index determination. Specifically, if a given intercept is within *tolerance 5* of a rational number with a lower denominator, we approximate the intercept with this rational number. The default level for this tolerance is 0.1.

Finally, the user can choose where he or she would like to save all of the output through the GUI. The default option for saving is the user's Matlab default files.