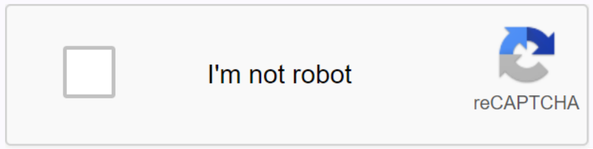

Continue

Lammps manual minimize

This tutorial describes how to use Atomsk to create data files for LAMMPS, and read the LAMMPS output files. It is already recommended to be familiar with LAMMPS to continue with this tutorial. ▶ For more information, see the relevant documentation page. LAMMPS is a code that performs simulations on classical particles - including, but not limited to, molecular dynamics. An LAMMPS calculation is typically controlled by a script file called in.something (see the input script structure in LAMMPS documents). This script contains information about units, interatomic potential, computational type (minimizing, molecular dynamics ...), and can also include commands to design and create an atomic system. If you're building your system using LammpS internal commands, you don't need any other software (such as Atomsk). Alternatively, read_data command, it is also possible to tell LAMMPS to read atomic locations from the external data file. In this case, you can design your system with a tool such as Atomsk and type atomic locations in a data file that is suitable for LAMMPS. This tutorial describes how to continue to process such files. Note: For now, Atomsk does not support bonds and molecules. 1. Let's create a data file for LAMMPS, create a unit cell from aluminum and write it in the form of a LAMMPS data file: atomsk --fcc 4046 Let's create al lammps with this command, Atomsk will create a LAMMPS data file named Al.lmp: # Fcc Al focused X=[100], Y=[010], Z=010]. 4 atoms 1 atom types 0.00000000 4.04600000 xlo xhi 0.00000000 4.04600000 ylo yhi 0.00000000 4.04600000 zlo zhi Atoms 1 1 0.00000000 0.00000000 0.00000000 2 1 2.02300000 2.02300000 0.00000000 3 1 0.00000000 2.02300000 2.02300000 4 1 2.02300000 0.00000000 2.02300000 The three lines xlo xhi, ylo yhi and zlo zhi define the box, and the section Atoms contains the index, type, and Cartesian coordinate of each atom. All atoms here are aluminum, so all type 1 (second column). Note that LAMMPS is not a standard extension for data files. Atomsk uses the extension .lmp, but this is not very important -they are already only text files. Such a file can be read from an LAMMPS input script by using the read_data command: 2. Converting an existing file to LamMPS is easy to convert to lammps if you already have an existing atomic system. For example, consider that there is an atomic system in the mysystem.cfg (i.e. Atomeye CFG format) file. Run Atomsk to convert to LAMMPS: atomsk mysystem.cfg lammps will generate a file called Atomsk mysystem.lmp. Again, this file can be used by LAMMPS read_data with the mysystem.lmp command. 3. Make the box suitable for lammps for computational efficiency reasons, lammps simulation box has two main requirements, it should be noted. These are detailed on this page of the LAMMPS guide First condition, condition, the first box vector is aligned with carter X and the second vector is located on the XY plane. Second, the box skew should not be too large, so the box should not be too distorted, and the box angles should be neither too acute nor too thick. If your simulation box does not meet these requirements, LAMMPS will most likely end in an error. First, let's create a unit cell of aluminum as before and rotate 45° around 45° and x around Y: atomsk --fcc 4046 Al -rotate 45° Y -rotate 45° X Al.xsf It is important to understand that this system is still an aluminum unit cell rotated by random vectors. If you open the Al.xsf file, you will notice that cell vectors have components in X, Y, and Z directions: # Fcc Al-oriented X=[100], Y=[010], Z=[001]. CRYSTAL PRIMVEC 2.86095404 2.023 billion -2.023 billion 0.86095404 2.86095404 2.86095404 -2.02300000 2.0230000 CONVVEC 2.86095404 2.023 billion -2.023 billion 0.0000000 2.86095404 2.86095404 2.86095404 -2.023000000 2.023 billion PRIMCOORD 4 113 billion billion billion billion 0.0.0 billion 0.0 billion 0.43047702 13 1.43047702 2.44197702 0.41897702 13 1.43047702 0.41897702 2.44197702 13 2.860995404 -0.0 billion Now, this issue: LAMMPS does not accept such vectors. In fact, it is not possible to write such vectors to a LAMMPS data file. So, if you try to write such an atomic system in a LAMMPS data file, Atomsk displays a warning: atomsk Al.xsf lammps /!\ WARNING: supercell does not create a sub triangle matrix, which is required by LAMMPS. Do you want to realigned the system? (y/n) If you select No, some components of the box vectors are missing from the LAMMPS data file, and the simulation will most likely go wrong. If you select Yes, Atomsk returns the system to comply with LAMMPS requirements, meaning the first box vector extends along the Carthesian X axis and the second box vector is inside the XY plane. Alternatively, if you always want to make sure that your simulation box is aligned appropriately for LAMMPS, you can use the -alignx option: atomsk Al.xsf -alignx lammps Now, the second restriction applied by LAMMPS is that the box should not be too skewed or trimmed too much. Let us consider a super cell of sizes 4x4x40 Å3, i.e. the original box vectors are: H1 = [4 0 0] H2 = [0 4 0] H3 = [0 0 40] Now, Let's cut this box by tilting the third vector a little (i.e. by changing the yz component): H1 = [4 0 0] H2 = [0 4 0] H3 = [0 1 40] This new box is still acceptable for a LAMS simulation. But now, if we bend further: H1 = [4 0] H2 = [0 4 0] H3 = [0 3 40] Such a box is unacceptable and LAMMPS ends with an error message: the box is skewed yz too large. This is because component Y of the last vector is larger than half of the Y component of the second vector. LAMMPS is a component of -0.5 and Multiplication of component Y of the second box vector. So here, it should be between -2 and +2. This can be achieved by assuming periodic limit conditions, adding or removing this component. In our example, the previous box is equivalent to the following (keeping all atoms in the same Carthesian positions): H1 = [4 0] H2 = [0 4 0] H3 = [0 -1 40] Now, the yz component is really between -2 and +2 and is suitable for these lammps. You can use Atomsk's -unkew option to make sure your system isn't too skewed. This option checks the box vectors and adds the appropriate components to make sure that the box is not skewed: atomsk Al.xsf -unskew lammps To summarize, if you want to make sure that your atomic system meets lammps requirements, use two options -alignx and -unskew when writing lammps data file: atomsk Al.xsf -alignx -unsw lamss lamss. If you work with ionic systems, you may want to add electricity charges to the LAMMPS data file. There are several ways to do this with Atomsk. First of all, if your first file already contains electricity charges, in most cases Atomsk reads them and keeps them in memory during conversion (and also if you convert the system with options). In this case, you don't have to worry: the LAMMPS data file produced by Atomsk will contain electrical loads. Now, if electrical loads are not defined in your first system, how to continue is absorbed? For example, let's create a unit cell of nacl: atomsk --create rocksalt 5.64 Na Cl lammps LAMMPS data file similar to: # Rocksalt NaCl-oriented X=[100], Y=[010], Z=[001]. 8 atom 2 atom tip 0.00000000 xlo xhi 0.00000000 0.60000000 ylo yhi 0.00000000 5.64000000 zlo zhi Atomlar 1 1 0.000000000 0.00000000 0000000 0.0000000 0 1 2.82000000 2.82000000 0.0000000 3 1 0.00000002 2.82000000 4 1 2.820000000 0...820000000 0.8200000000 00000000 0.82000000 5 2 2.82000000 0.0000000 0.00000000 6 2 00000000 0.820000000 0 0 2 0.000000000 0 0.00000000 0 .00000000 8 2 2.820000000 2.82000000 2.820000002.82000000 Not bu kez, ikinci sütun tip 1 ve tip 2 (sırasıyla Na ve Cl atomları) atomları olduğunu gösterir, ancak sütun sayısı önceki aynıdır. We expect this material to be ionic, but Atomsk cannot predict electrical loads alone - we need to provide them clearly and there are two ways to add them. If you have already built your system and just want to add charges, it is the first possibility to use the lmp_atom2charge.sh script provided with atomsk (/tools/folder). Provide your file name only as an argument: lmp_atom2charge.sh NaCl.lmp This script adds a new column that contains a second zero, as shown below: #Rocksalt NaCl-oriented X=[100], Y=[010], Z=[001]. 8 atom 2 atom type 0.0 billion 0.64 billion xlo xhi 0.0 billion 5.6 billion ylo yhi 0.000000 000 0 5.64 billion zlo zhi Atoms 1 1 0.0 billion 0.0 billion 0.0 billion 0.0 billion 0.0 billion 2 1 0.0 2.82 billion 0.0 billion 3 1 0.0 0 0.0 billion 2.82 billion 2.82 billion 4 1 0.0 0 2.00000 82 billion 0.82 billion 2.82 billion 2.82 billion 0.82 billion 0.82 billion 0.82 billion 0.8 billion 0.82 billion 0.8 billion 2 0.0 billion 2.82 billion 0.0 billion 7 2 0.0 billion 0.0 billion 0.0 billion Obviously, this approach does not add the correct electricity charges to the file. Therefore, electricity charges must be defined in the lammps input script, with the keyword set fee. However, make sure that this approach has the correct number atom_style to charge the data file NaCl.lmp - otherwise you would complain that the LAMMPS data file has the wrong format. A second possibility is to use the -properties option to read electrical loads in a file. Let's write this feature (fee) to a text file: Then, When you create the system, tell Atomsk to read this feature from this file: atomsk --rocksalt 5.64 Na Cl properties charges.txt lammps In this case, Atomsk will generate a LAMMPS data file that contains an additional column containing the electrical charges we provide: #Rocksalt NaCl-oriented X=[100], Y=[010], Z=[001]. 8 atom 2 atom tip 0.00000000 xlo xhi 0.00000000 0.60000000 ylo yhi 0.00000000 5.64000000 zlo zhi Atomlar # şarj 1 1 1.000 0.00000000000 0.00000000 0.0000000 2 1 1.000 2.82000000 2.82000000 0.0000000 3 1 1.0000000 2.820000000 4 1.0 000 2.820000000 0.0000000 0.82000000 5 2 -1.000 2.82000000 0.00000000 0.0000000 6 2 -1.000 0.000000000 2.820000000 0 0.000000000 7 2 -1.00000000 0.00000000 0.80000000 8 2 -1.000 2.82000000 2.820000002.820000000 Bu kez, dosya tüm iyonların elektrik ücretleri içerir, bu nedenle, LAMMPS giriş komut dosyasında set charge anahtar kelimesi ile yeniden tanımlamaya gerek yoktur. Also note that Atomsk added the keyword #charge after the keyword Atoms. This keyword is optional, but it helps the user understand that this data file atom_style for the cost of the data in LAMMPS. 5. Read and convert LamMPS dump files Now, let's say you do a calculation with LAMMPS. LAMMPS should know that it can produce dump files that contain atomic locations. This can be achieved by the dump command in the LAMMPS input command file. For example, To create atomic data files in Atomeye CFG format, the dump command can be paired with the dump_modify command so that the atomic elements are properly written: dump myCFG all cfg 100 dump_*.cfg mass type xs ys zs dump_modify element Al Reduced wrapped coordinates (xs, ys, zs) or reduced unwrapped coordinates (xs, ys, zsu, zsu) output is possible in this format. It is also possible to write down any amount calculated by Lammps, such as atomic speeds, forces, voltages. These are written as auxiliary properties in the Atomeye CFG file. The star in the file name (dump_*.cfg) will be replaced by the simulation step. Here we want a dump file every 100 steps, so LAMMPS will produce dump_0.cfg, dump_200.cfg, et cetera. A second possibility is to use the LAMMPS custom output format: dump all custom 100 dump_*.lmc id mass type element xu yu zu dump_modify element Using Al id attribute is preserved in the index of atoms (i.e. always listed in the same order). Again note, LAMMPS does not have a standard extension for custom dump files, so use this tutorial extension .lmc (short for LAMMPS custom format). As before, such a command will produce files called lammps dump_0.lmc, dump_100.lmc, dump_200.lmc, and so on. Atomsk can read files either way, i.e. Atomeye CFG format or LAMMPS custom format. For example, you can use XCrySDen or VESTA to convert the dump_100.lmc dump file for visualization: atomsk dump_100.lmc xsf If you do a long simulation that produces a large number of output files, you can use Atomsk to convert all these files using --list mode. First, create a list of all these files (for example, with ls) and save them to a text file (for example, myfiles.lst). Then run Atomsk in --list mode: ls dump*.lmc > myfiles.lst atomsk --list myfiles.lst xsf If you have already converted some files while simulation is still running and want to convert only the remaining files, then use the -ignore option. Atomsk will only ignore already converted files: atomsk --list myfiles.lst xsf -ignore 6. Ionic systems: Protect electrical charges If you work with Ionic systems, you may want to learn about electrical charges after LAMMPS simulation. To do this, tell me to write electrical charges when dumping lammps files, adding the dump command q as below: mydump all custom 100 dump_*.lmc mass type element xu yu zu q dump_modify element Al This way, the dump files will contain the electrical charge of the ions. When Atomsk reads these files, it stores fees as helpful features. This is useful for maintaining electrical overhead if you want to convert these dump files to new LAMMPS data files, or if you want to visualize electrical charges (for example, in Atomeye) or calculate a feature based on these charges. 7. Deformation simulations: If your box is orthogonal, the LAMMPS data file does not contain information about the slope factors (since they are all equal to zero), make the box triclinic by default. However, performing a simulation where you can shear the box (i.e. changing the bevel factors) requires that the lammps data file contain bevel factors. This can be easily obtained by bash script lmp_ortho2tri.sh provided in atomsk / folder: lmp_ortho2tri.sh myfile.lmp This script only adds a line containing 0.0 billion xy xz yz to the lammps data file. File.