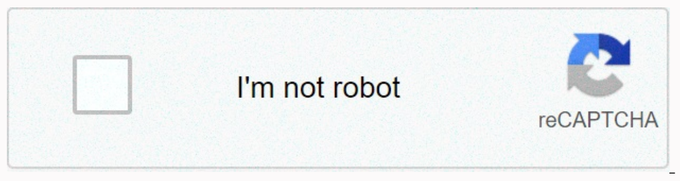

Continue

Vesta repeat unit cell

In order to study systems relating to heterogeneous catalysis, extended supercells of bulking plates must be formed and surrounded by vacuum to mimic the properties of the exposed area. Today, I'll go through how you can use the VESTA program to set the initial coordinates for your superceles. An example that we will build to this day is FCC Pt. We use the experimental parameter lattice 3.92 Angstroms. First, start by downloading vesta. Then go to file>New Structure... menu to start building your structure. Let us turn our structure into something useful, such as Pt(111). Then click the Drive Cell tab. Our material is from the F m -3 m space group or #225 and is cubic. To select this option, first click Cubic for Crystal, and then join the space group. Next, set a =3.92 Angstroms to match the experimental lattice parameter. You should see something like the window below: Finally press OK for now. It's also a good idea to save a file withFile>Save As ... and choose a name like Pt(111).vest We still need to add more properties to build our cell. So, let's go back and this time with Edit>Edit Data>Structure Parameters ... Here we have to add to our atoms. In our FCC space group, we only need one atom at its inception to represent the filling of all the seats in our grid. The rest is automatically generated. So we need to make one atom with the symbol Pt and the label Pt at 0.0.0. Click New on it. The window should look like this: Now when you press OK, you should see that the FCC unit cell has been generated: The next three steps are where it gets a little more complicated. But trust me, it's worth it! 1) We want to reorient our cell so that the surface (111) is exposed and normal for the x-axis. Return to the editing properties. Edit>Edit Data>Phase... and change the orientation to [u in w] 1 1 1 and perpendicular to 1 0 0 and parallel. The window should look like this: Press OK. Now you should see a different orientation. Note that dragging a cell in a graphics window changes the aperance, but not the alignment of atoms. Now he's moving on to harder things... 2) We want to build a supercela 2x2x2. To do this, we will want to adjust the boundaries to allow VESTA to draw more atoms for us. You can edit borders by clicking the border... in the left sidebar. We want to set the minimum and maximum in fractional coordinates to a range (-1.99,2.00). We don't duplicate atoms, i.e. -1.99 instead of -2.00 at the bottom end of the range. Your window should look like this: 3) Here comes the perishing parts ... Now we want to create a cutoff aircraft, so our supercela is shaped along the right dimensions. It takes a bit of trial and error, but here's how we do it: We want to add the following aircraft shape and beat up our atoms on supercela shape: Miller's Indices Distance (-1 -1 -1) 1.99 (-1 1 -1) 2.00 (-1 1 1) 2.00 (1 -1 -1) 1.99 (1 -1 1) 1.99 (1 1 1) 2.00 Differences in distances are due to differences in rounding in the way cutting planes are used. If you use all the cutting planes, as described above, you should end up with 64 atoms. Your window should look like this: Now we are ready to save our coordinates. Go to File>Export Data... and save the XYZ coordinates. You can check that everything you have done is correct in several different ways. First, you can look at the coordinate file to see if the generated coordinates make sense. You can also continue and visualize them in VMDs with periodic border conditions turned on. In the plane of our board, our regular repetition is sqrt(2)*2*a. For separation boards, we can only set that number to something large, eg 30. So we set pbcs with this command: pbc set {30.0 11.08 11.08 60 90 90} Now we can look at our extended system by increasing the number of cells drawn in each direction and verify that we have the correct coordinates. It should look a bit like this: I hope you found this construction tutorial useful. If you have any questions, please email me! Ball-and-stick representation of the diamond in two different unit cells. Left: A primitive unit cell containing two atoms. All atoms at the tops are regular copies of the same. Right: Conventional cubic unit cell containing eight atoms. Atoms on opposite faces are regular copies, while all atoms at the tops are regular copies of the same atom. When performing electronic structure calculations on complex systems, you prefer to do so on systems with as few atoms as possible. Such periodic cells are called unit cells. However, there are two types of unit cells: primitive unit cells and conventional unit cells. The primitive cell of the unit is the smallest possible periodic cell of crystalline material, which is very suitable for calculations. Unfortunately, it is not always the most beautiful unit of the cell to work with, because it can be difficult to recognize that it is symmetry (compare the example of a diamond on the right). The conventional unit cell on the other side shows symmetry more clearly, but is not (always) the smallest possible unit cell. To make it complex and confusing, people often refer to both types as simply a unit cell, which is not bad, but for many the term unit cell is uniquely associated with only one of these two types. When you perform calculations on a diamond, the conventional cell is not so large that standard calculations become impossible, even on laptop or desktop computer. On the other hand, when you study a Metal-Organic Framework like UiO-66 (Zr), which contains 456 atoms in your conventional cell unit, you will be very happy to use primitive unit cells with 'only' 114 atoms. Also MIL-47/53 topology, which is generally studied using a conventional unit of cells containing 72/76 can be reduced to smaller primitive cell units of only 36/38 atoms. As for diamond primitive cell units, this MIL47/53 primitive unit cell is not a nice cubic cell. Instead, you end up with a lattice with lattice angles of seventy-something degrees. Reduction of conventional cell MIL-53 to primitive cell. A conventional cell appears that expands slightly into regular copies. Primitive lattice vectors are displayed as colored arrows. The folded primitive cell shows that symmetry has broken in the hydroxyscupital groups of the metal oxide chain. Introducing some additional symmetry corrects it in the final primitive cells. How do I reduce a conventional cell unit to a primitive cell unit? Before you start, and if you are using VASP, make sure that you have a POSCAR file giving atomic positions as cartesian coordinates. (Using the HIVE-4 toolkit: Option TF, Suboption 2 (Dir->Cart).) If you are not using VASP, you can still use the system below. Open your structure using VESTA, and save it as a VASP file: POSCAR.vasp (File → Export data → select VASP as the file type, select cartesian coordinates (do not select Convert to Niggli Reduced Cell because it only works for perfect crystal symmetry)). Open the file you just saved in a text editor (such as a notebook or notebook++). The file format is fairly straight forward. The first line is a comment line, while the second is a general scale-factor that for our current purpose can be ignored. It is important to know that the third, second and 5th 6th and 7th rows give the order, type and number of atoms for each atomic species (In VASP 5.x, the older format of VASP 4.x does not have the 6th row). The eighth line should speak Cartesian. As from 9 October 2004, member states shall Select 1 atom in your conventional cell that you will use as a reference point. Get primitive cell lattice lattice vectors by generating vectors from the reference atom. (cf. figure above) Using VESTA it can be done as follows: Open a conventional cell in VESTA (if you closed it after step 1). Use the distance selector (5th symbol from above in the menu on the left) and select a reference atom for each of the primitive lattice vectors and it's a primitive copy. Subtract the fractional coordinate of selected atoms provided by VESTA to get a fractional primitive vector (the primitive vector will be called aprim,frac) Multiply each of the conventional vectors (aconv, bconv and cconv) with the relevant fractional primitive vector component, and add the resulting vectors to get a new primitive vector: ⇒ aprim = axprim, frac aconv + aprimy, frac bconv + azprim, frac cconv So imagine that the lattice vectors mof above are a = (20, 0, 0), b = (0, 15, 0) and c = (0, 0, 5). A primitive fractional vector was found to be aprim, frac = (0.5, -0.5, 0.5). In this case, the aprim vector becomes: aprim = (10, 0, 0) + (0, -7,5, 0) + (0, 0, 2,5) = (10, -7,5, 2,5). Replace conventional lattice vectors in POSCAR.vasp (cf. step 2) with new primitive lattice vectors. Save the file. Open POSCAR.vasp in VESTA. If all went well, and conventional cells weren't real primitive cells anymore, you should see nice new primitive cells with the equivalent of atoms perfectly overlapping one-another. This is also why you have the initial geometry in the card coordinates. If you had your atomic positions as fractional coordinates, this first check would not work at all. In addition, you should calculate the new fractional coordinates of the atoms in the primitive unit of the cell. If all is well, you can close POSCAR. VASP in VESTA. (If something is wrong: either you did something wrong, and you should start over, or it wasn't actually a super cell of primitive cells you started building.) Get the atoms of the primitive unit of the cell. Because our atomic positions are in cartesian coordinates in our original geometry set, now you just need to create a list of individual copies of equivalent atoms. Using VESTA (the original file structure you have kept open since step 1) you can click on each atom you want to keep and write down their index (this is the first number you can find on the line with cartesian coordinates) ... For example: In the case of MIL53-MOF you can select all metal and oxygen atoms 1 chain, and two more continuous molecules. Remove any redundant atoms (i.e. those from which you did not write the index) from POSCAR.vasp (using a text editor). You may want to create a backup of this file before you run :-). Update the number of atoms in the seventh line of the POSCAR.vasp file to make sure that the number is correct. The conventional cell should have had a whole multiple of the number of atoms in the primitive cell. Save the final structure as POSCAR_final.vasp . Data POSCAR_final.vasp should include new lattice vectors and a list of atoms for a single primitive unit cell. Check this by opening the file using VESTA, and make sure you haven't removed too many or too few atoms. If not, go back to step (a) and check twice. (If you are using hive4-toolbox you can first transform POSCAR_final.vasp back to direct coordinates because it can make atoms visible that nicely overlap in cartesian coordinates in VESTA and save the structure again. This is useful if you want to play with a molecule. Unfortunately for fixed you must also get new lattice vectors, which did not happen. As a result, you end up with some atoms floating around in a regularly repeated array with the original grid parameters. Steps 1-5 above provide an easy way not to jump in this situation, but require some writing on your part. PS 2: Reverse transformation, from primitive cell unit to conventional unit cell, using VESTA, is shown in this youtube video. Video.