Tetrahedral Passivation

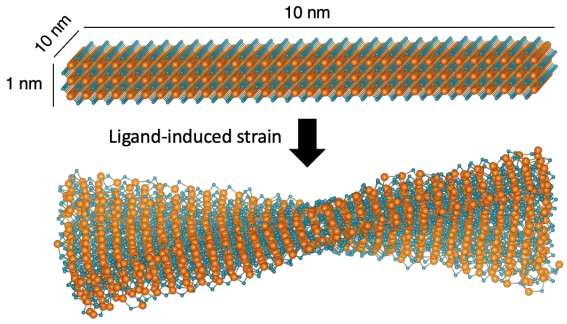
Daniel Chabeda

April 2023

Contents

1 Motivation

This document outlines a tetrahedral passivation strategy that can be used in general, even when it is not possible to modify the number of layers of the nanocrystal. In those cases, it is not viable to simply convert the outer layer of the nanocrystal into passivation ligands, but one must solve the geometric problem of finding the missing vectors to complete an arbitrarily oriented tetrahedron. For example, a collaborator sent us an output file from MD simulations of CdSe nanoplatelets passivated with acetate ligands. The acetate ligands induce a twist along the platelet, resulting in chiral nanoplatelets. This twist angle depends on the number of layers of the nanoplatelet, so no new layers can be added. And the acetate ligands do not fully passivate the surface nor coordinate with a tetrahedral geometry on the surface. If we want to have an electronically defect-free crystal, it is necessary to solve the vector geometry problem so we can appropriately place our passivation potentials.

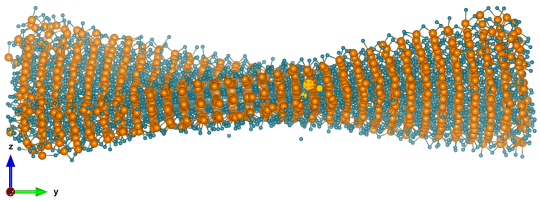


2 Algorithm Outline

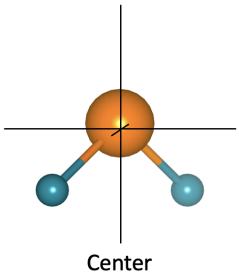
The script for passivating CdSe is "passivate_tetrahedral.py." The code will take a "conf.par" file as sys.argv[1] and an optional passivation ligand type ('P1', 'P2', 'H') as sys.argv[2]. It will loop over all atoms and populate a neighbor list that identifies which atoms have less than 4 nearest neighbors and thus need to be passivated. Atoms with 2 or 3 bonds will be passivated.

The general algorithm outline is as follows

1. Select center atom that needs passivation (and its 2 or 3 nearest neighbors).



2. Translate coordinates so center atom is at origin.



3. Rotate atoms to align with a **chosen** orientation that simplifies addition of tetrahedral bonds.

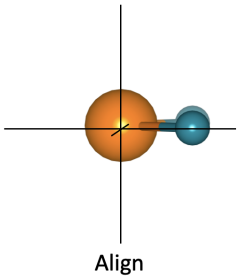
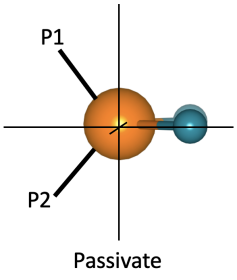
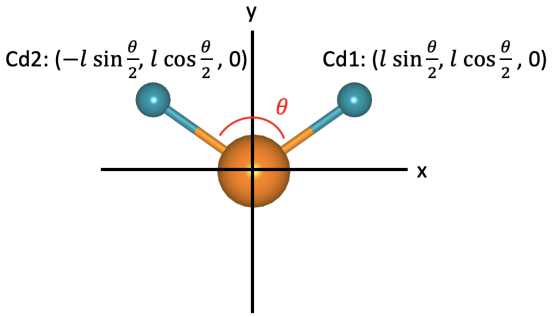


Figure 1: Caption

4. Add ligands.



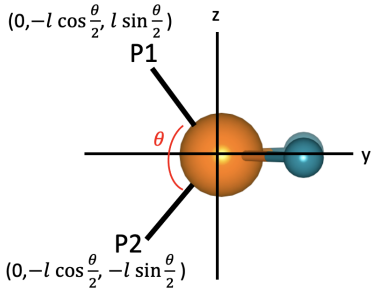
5. Invert rotation and translation; put the atom, with its ligands, back where they were.
6. Move to next atom that needs passivation.



2.1 Standard Orientation

The following orientation makes it simple to place tetrahedral passivation ligands.
The three selected atoms are in the xy-plane, with the y axis bisecting the ≈ 109.5 tetrahedral angle. In this configuration, the completed tetrahedron vectors just have coordinates

$$P1 :(0, -l\cos\frac{\theta}{2}, l\sin\frac{\theta}{2})$$
$$P2 :(0, -l\cos\frac{\theta}{2}, -l\sin\frac{\theta}{2})$$



2.2 Rotation matrices for atoms with 2 bonds

If the surface atom needs 2 passivation ligands, the exact procedure from "Algorithm Outline" step 3 will be:

1. Choose one of the Se-Cd bonds and align it to Se-Cd1 in Fig 2.1. This is achieved by constructing the rotation matrix R_1

$$R_1 = \mathbf{I} + [v]_\times + \frac{1}{1+c}[v]_\times^2$$

where $[v]_\times$ is the skew-symmetric cross product matrix

$$[v]_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_1 \\ -v_y & v_x & 0 \end{bmatrix}$$

and c is cosine of the angle between the arbitrarily located Se-Cd bond and the Se-Cd1 standard orientation.

$$c = \frac{(\text{Se-Cd}) \cdot (\text{Se-Cd1})}{\|\text{Se-Cd}\|\|\text{Se-Cd1}\|}$$

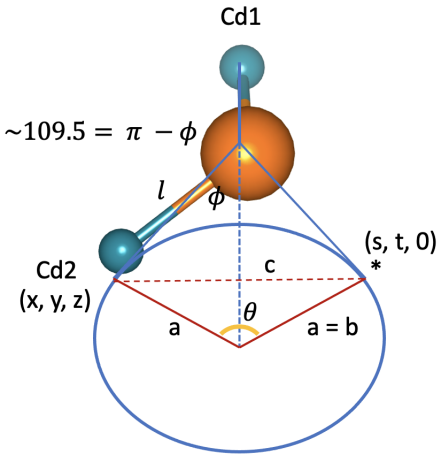
Note that the bond lengths should be normalized. See this StackExchange article for details. This has aligned one of our bond vectors into the standard orientation, but the other one can be randomly oriented anywhere.

2. Next, we determine the angle necessary to rotate the remaining Se-Cd bond to the Se-Cd2 position, considering a rotation about the Se-Cd1 bond axis. The rotation matrix to rotate about an arbitrary vector by θ is R_2

$$R_2 = \mathbf{I} + \sin\theta[v]_\times + (2\sin^2\frac{\theta}{2})[v]_\times$$

where v = Se-Cd1 is the previously oriented bond. See this StackExchange article for details. How do we find the rotation angle θ? This is the crux of the problem, so it has its own section.

2.2.1 Finding the rotation angle



The vector of the Se-Cd2 bond sweeps out a cone when rotated about the Se-Cd1 bond. If we want to rotate the Se-Cd2 vector to align with a point, $* \equiv (s,t,0)$, in the xy-plane, then we can find the rotation angle in the following manner.

1. We first find the length of the projection of Se-Cd2 into the plane of the base of the cone, a.

$$\frac{v_{Cd1}\cdot v_{Cd2}}{||v_{Cd1}||||v_{Cd2}||} = \cos(\pi-\phi)$$

$$\pi - \phi = \cos^{-1}\left(\frac{v_{Cd1}\cdot v_{Cd2}}{||v_{Cd1}||||v_{Cd2}||}\right)$$

$$\phi = \pi - \cos^{-1}\left(\frac{v_{Cd1}\cdot v_{Cd2}}{||v_{Cd1}||||v_{Cd2}||}\right)$$

The length of segment a is

$$a = l\sin(\phi)$$

2. The distance c gives the third side to the triangle formed in the plane of the base of the cone. We obtain its distance by using our knowledge of the coordinate $(s,t,0)$. We defined it in our standard representation as (-2.18,1.541,0.0) so that the tetrahedron has side lengths 2.67 and bond angles 109.4712. The distance is thus

$$c = \sqrt{(s-x)^2 + (t-y)^2 + z^2}$$

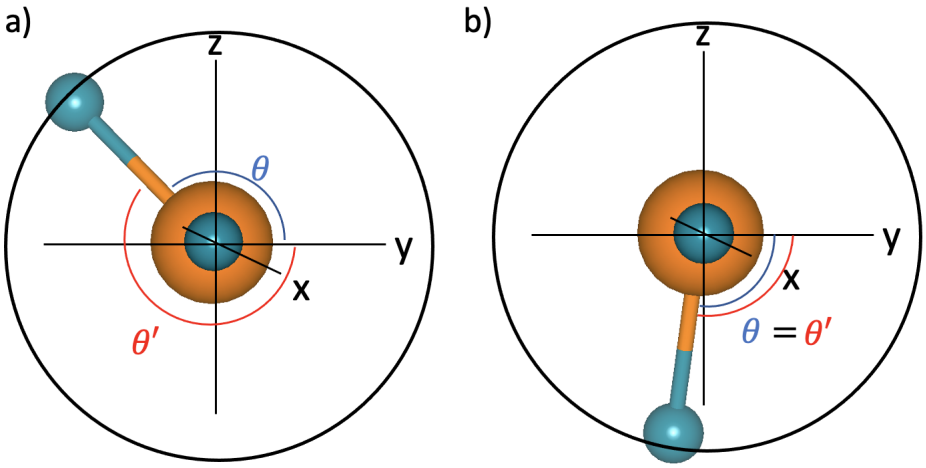
3. Using the law of cosines, we can find the rotation angle

$$c^2 = a^2 + b^2 - 2ab\cos(\theta)$$

In our case, $a = b$, so we have

$$\boxed{\theta = \cos^{-1}\left(1 - \frac{c^2}{2a^2}\right)}$$

Even with this angle obtained, we are not done. The rotation matrix will always rotate counter clockwise, and our method of obtaining the angle θ does not take into consideration whether we want the "clockwise" or "counterclockwise" angle. Consider the two examples below.



Looking down the Se-Cd1 bond, there are two possible cases for the orientation of Se-Cd2: a) above the xy plane and b) below the xy plane. If Cd2 is above the xy plane, then the angle calculated by the law of cosines, θ, **will not** equal the angle necessary to perform a counterclockwise rotation into the plane, θ'. The necessary angle θ' is just $\theta' = 2\pi - \theta$. If Ccd2 is below the xy plane, as in b), then the computed angle and the counterclockwise angle are equivalent.

IMPORTANT NOTE: The arc cosine function is only defined on a range of π radians so as to not be multivalued. This means that half of the range of possible θ angles is unsupported in the np.arccos() function (by mathematical definition, not a gap in code functionality). In addition, arguments close to the edge of the domain [0,π] are numerically unstable. If the position of Cd2 randomly ends up in the xy-plane, then this will be an issue. The code is currently written to check whether the z component of the Se-Cd2 vector is close to zero, and if so, rotate around Se-Cd1 by 30° before calculating the final rotation angle θ. This helps everything remain numerically stable. Make sure that, if you change the definitions of angles here that you properly handle angles outside the range $[0, \pi]$.

3 Implementation details

This information is enough to passivate an arbitrarily-oriented incomplete tetrahedron. The code uses functional style (though perhaps object-oriented might be useful for holding onto transformation matrices for each atom, idk). A tutorial for how to passivate structures is built into the code and can be accessed by providing the command line argument 'python passivate_tetrahedral.py tutorial'.

4 Output

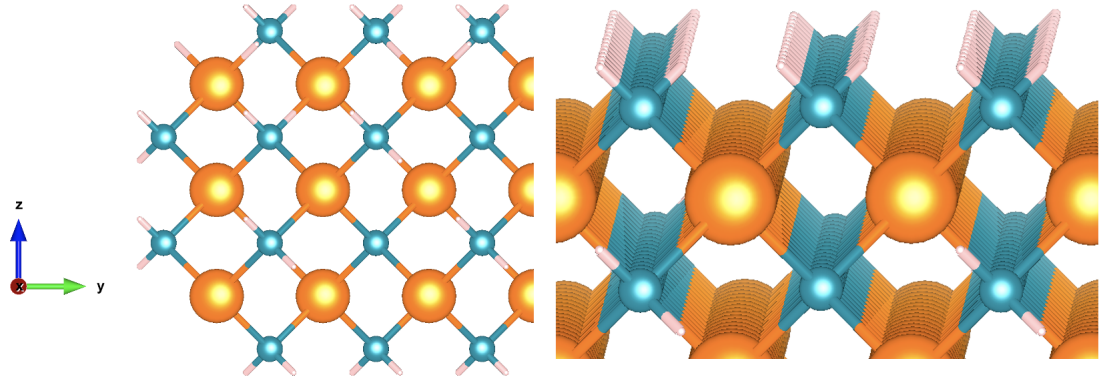


Figure 2: Top surface with the b axis in plane of the screen.

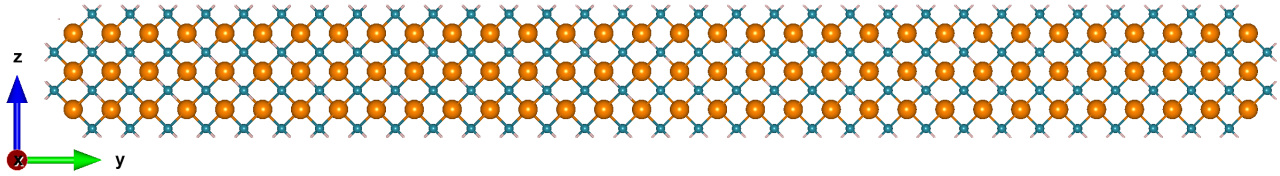


Figure 3: Passivation of flat nanoplatelet as viewed down the a axis

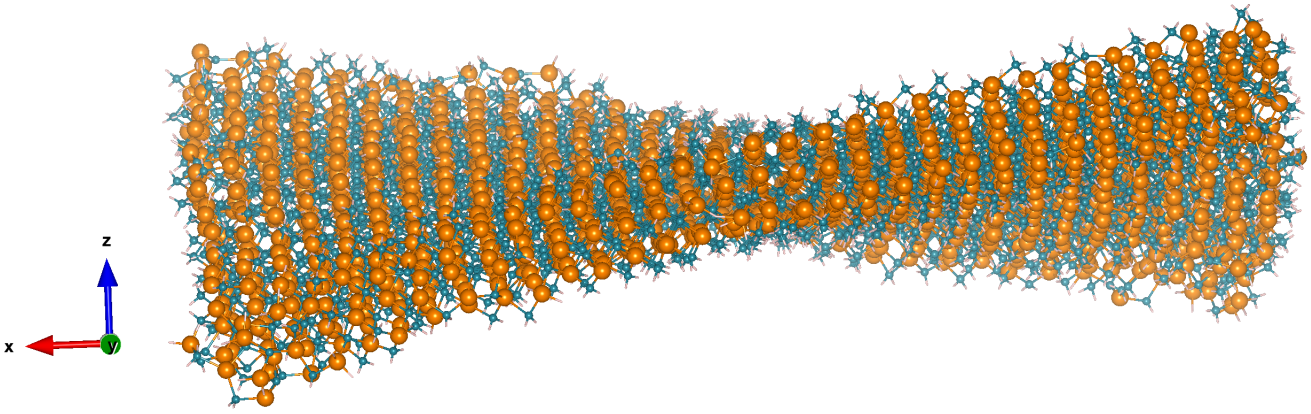


Figure 4: Passivation of twisted nanoplatelet as viewed down the b axis