molecular_plane_analyzer

Analyze the predominant orientation of molecular planes within a given morphology by simple specification of normal vectors within any MD format. A workflow for all MD formats is presented. The input is handled via a command .fcf file containing the necessary parameters.

run this program from the command line like:

```
normal_vectors.py [input.xyz] [control_file.fcf]
```

where input.xyz is a file containing only your molecule of interest, where multiple frames are possible. you can generate this file for example by selections in VMD and printing only your molecule of interest into an .xyz file. if you have multiple molecules in your system you need to repeat this step for each molecule type you want to analyze

if the .xyz file was generated the input paramters of the script need to be specified within the control file. this looks like, where comments start with a # :

```
#calculates the normal vectors on a molecule
#surface based on reference vectors
NORMALVEC
ATOMSPERMOLECULE 118
NORMALVECTORSTEP 1
BOXVECTORS 282.8459  304.3587  283.0572
DEFINECENTER
23 53
END_DEFINECENTER
DEFINEREFERENCE
19 28 54 49
END_DEFINEREFERENCE
END_NORMALVEC
```

ATOMSPERMOLECULE should equal the number of atoms within your molecule of choice. NORMALVECTORSTEP defines how often the normal vectors should be computed =1 means each step BOXVECTORS are BOX vectors in Angstrom DEFINECENTER then takes an arbitrary long list of atom indices, where 0 corresponds to the first atom in your molecule and computes the geometric center of these atoms to define an origin for the normal vector DEFINEREFERENCE takes an arbitrary long list of atom indices, where 0 corresponds to the first atom in your molecule, to define the reference vectors for computing the normal vector (see normalvectors.png) the cross products are then computed as (first reference - center) x (second reference - center), (second reference - center) x (third reference - center) .... where the last is definded with respect to the first (last reference - center) x (first reference - center) yields: n1, n2, n3 ... the cross product is always perpendicular to the plane of the 2 vectors, so it is the sought normal vector. the normal vector is then the average over all the reference vector cross products: (n1 + n2 + n3 + ...) / norm(n1 + n2 + n3 +

..)

The sections DEFINECENTER and DEFINEREFERENCE may repeat for as many normal vectors you define within your molecule.

If this has been prepared you are ready to run the script. This yields 4 files: normalvectors.xyz - normal vectors visualization file check_normalvectors.xyz - verbose visualization file angle_dist_vectors.xyz - file containing the center coordinates and the normal vectors sum_over_all_normalvectors.dat - file containing the sum over all normal vectors within each step

Where (see the following .tcl script for arrow like visualization) : C-Atom : center of the molecule O-Atom : head of the normal vector H-Atom : normal vectors again centered at 0 to visually judge the predominant orientation in VMD N-Atom : reference atoms

The first 2 files are for checking by visualization in vmd. Load normalvectors.xyz, set it as Top (little T in the VMD main menu) and then load the script draw_normalvectors_in_vmd.tcl if you now type: drawallnormalvectors into the VMD console all normal vectors will be displayed. You can load your initial geometry on top.

If you want even more checking with all reference vectors displayed use the check_normalvectors.xyz file and load it in VMD and set it as Top. Now load check_normalvectors_in_vmd.tcl if your now type: checkallvectors into the VMD console all reference vectors together with the normal vectors will be displayed.

If you are happy with the resulting normal vectors you can analyze their distribution by the script vector_angle_to_distance. This script also runs from command line like:

```
vector_angle_to_distance.py vectors_from_which_to_compute.xyz
 vectors_to_which_to_compute.xyz boxX boxY boxZ cutoff
```

where the 2 input .xyz are the angle_dist_vectors.xyz outputtet by the normal_vector script. If you e.g. want to compute all vectors from pbi to p3ht generate a file containing only pbi and send it into normal_vectors, the resulting angle_dist_vectors.xyz is the input file which then needs to go as vectors_from_which_to_compute.xyz then generate a second .xyz file only containing p3ht and send it into normal_vectors. The resulting resulting angle_dist_vectors.xyz is the input file which then needs to go as vectors_to_which_to_compute.xyz

boxX/Y/Z are again the box vectors in ANGSTROM. The cutoff in ANGSTROM defines how far you want to search for neigbours. Dont wonder if atoms with a dist bigger than the cutoff are present, because this program uses a grid search to speed up computation, thus distances up to sqrt(cutoff ** 3) should be encountered, but no bigger distances.

If you want to analyze from the same to the same molecule use the identical angle_dist_vectors.xyz output file as input. If you want to analyze from different molecules to certain ones, generate proper .xyz input files for normal_vector, than concatenate all the angle_dist_vectors.xyz files you receive and enter them as vectors_to_which_to_compute.xyz

vector_angle_to_distance then generates a file containing the indices from the inputted xyz files followed by the distance and angle between normal vectors. this can be converted into a density plot by the script get_point_density.py as follows:

```
get_point_density distance_to_angle.dat
distance_to_angle_density.dat 40 180 3 4 30 20 0 0
```

the expected parameters for the density plot are: get_point_density [input.dat] [output.dat] [x_range] [y_range] [col_1] [col_2] [grid size x (int)] [grid size y (int)] [x shift] [y shift] where the above default parameters should suffice. If you want to compute larger distances, be sure to increase the x_range (distance) to your maximum distance in the distance to angle file generated by vector_to_angle,dat (needs to be larger than largest value 3rd column).

this produces an output file distance_to_angle_density.dat which can be plotted by gnuplot like this for example: here 30 and 20 are your xgrid and ygrid size from get_point_density

```
set dgrid3d 30 20
set pm3d
set xlabel 'distance [A]'
set ylabel 'angle [Deg]'
set view map
splot 'distance_to_angle_density.dat' w l
```