README Documentation

Effect of Core Morphology on the Structural Asymmetry of Alkanethiol Monolayer-Protected Gold Nanoparticles

Alex K. Chew and Reid C. Van Lehn*

Department of Chemical and Biological Engineering, University of Wisconsin – Madison, Madison, WI, 53706, United States.

*send correspondence to: vanlehn@wisc.edu

Table of Contents

1. Description

The purpose of this document is to go through a step-by-step procedure in producing detailed figures and information for the DOI: https://pubs.acs.org/doi/full/10.1021/acs.jpcc.8b09323

2. Directory structure

This directory contains the following files:

- **bin**: contains all global variables and functions.
- **prep_files**: contains all preparation files (*e.g.* force fields)
- **prep_system**: contains all preparation systems (*e.g.* gold core development, self-assembly simulations, *etc.*)
- **scripts**: contains all scripts used to generate systems
- **simulations**: contains production *NPT* simulations used for this work

The next sections go through in greater-depth the description of the files.

2.1. bin

All scripts load code from the bin, representing global variables and functions. Here, is a description of each file:

- **nanoparticle_functions.sh**: contains all functions that is used throughout the work
- **nanoparticle_global_vars.sh**: contains all global variables. **Note that the user may need to change file paths in this script to correctly run all other code!**
- **nanoparticle_rc.sh**: contains all initial paths. This script essentially loads all global variable and functions. Note that part of this code needs "server_general_research_functions.sh", which was originally in another directory and copied within this bin. Similarly, "MDBuilder" and "MDDescriptor" were previously in different directories; see Section on "Other notes" for more details.
- **server_general_research_functions.sh:** contains general research functions
- **MDBuilders**: python package developed to generate systems.
- **MDDescriptors**: python package developed to analyze trajectories using MDTraj python module.
- **bashfiles**: contains all bash scripts that you would use on your desktop/server. These scripts are necessary for sending ligand topologies to the server.
- **pymol_files**: contains all PyMol files necessary to view the nanoparticle and color-code by bundling groups

2.2. prep_files

This directory contains all preparation files required to run simulations. A description for each file:

- **fragment_based_ligands**: contains fragment based ligands, which was used to generate self-assembly simulations
- **input_files**: contains all force field materials, submission scripts, and *.mdp files
- **ligands**: contains forcefield parameters for the ligands. Each ligand contains an *.itp, *.pdb, *.top, *.prm file, all of which are generated from CGENFF.

2.3. prep_system

This directory contains all self-assembly simulations around the gold core. A description of each file is:

- **prep_gold**: contains all self-assembly simulations of butanethiol around the gold core.
- **prep_ligand**: contains all energy-minimized ligands used for transfer-ligand process.

2.4. scripts

This directory contains all important scripts used to generate systems, and run analysis tools. A description of each file is:

- **analysis_full_extraction.sh**: uses both "analysis_extraction.sh" and "analysis_xtctruncate.sh" to copy trajectories from a completed simulation and move them into a "analysis" directory.
- **analysis_extraction.sh**: copies the trajectories from a specified location to an analysis directory
- **analysis_xtctruncate.sh**: truncates a specified trajectory and applies periodic boundaries to ensure all molecules are set to "whole."
- **full_prep_self_assembly_gold_ligand.sh**: main script to prepare self-assembly on gold cores with butanethiol ligands.
- **full_prep_system_planar_sam.sh**: main script to generate planar SAMs
- **full_transfer_ligand_builder.sh**: main script to exchange the self-assembled butanethiol on gold to more complex ligands (*e.g.* hexadecanethiol).
- **post_extraction.sh**: script to extract information from the trajectories (*e.g.* eccentricity) after simulations are complete.
- **prep_ligand_energy_min.sh**: script to prepare ligands by energy minimizing them, enabling them to be used as a "transfer ligand."
- **prep_ligand_full_script.sh**: full code to prepare a ligand outputted from CGENFF. Note that this code may have issues if ligands have larger than 100 atoms – yet to be debugged. It stems from gmx editconf being incapable of converting >100 atoms pdb files to gro files. Please double-check topologies when generating ligands >100 atoms. This will be fixed in the future.
- **prep_self_assembly_gold_ligand.sh**: Code to prepare a single simulation of self-assembly butanethiols around a gold core.
- **prep_self_assembly_ligands.sh**: Code to generate self-assembly ligand box
- **prep_system_planar_sam.sh**: Code to prepare planar SAMs
- **restart_gromacs_jobs.sh**: Code to check if a GROMACS simulation correctly completed
- **transfer_ligand_builder.sh**: Code to transfer a single simulation of self-assembled butanethiol ligands around the core and replace the ligands with more complex ligands (*e.g.* hexadecanethiol).

The "scripts" directory also contains the following directories:

- **analysis_scripts**: contains analysis scripts
- **post_extraction_scripts**: contains scripts for post extraction processes (*i.e.* completed simulations)
- **submit_scripts**: contains submission scripts for other servers (*e.g.* XSEDE)

3. System preparation procedures

The remainder of this document is focused on procedures used to generate ligands and create nnaoparticles from the available files.

3.1. Preparing ligands

Ligands were drawn using any molecular editor. We used SPARTAN 2010 version, but other tools such as Avogadro should suffice. The ligands were drawn with the SH head group attached (*e.g.* for butanethiol, $SHCH_2CH_2CH_3$). After the ligands were drawn, you should have a *.mol2 file, containing all important atom and bonding information of the ligand. Then, the following procedure was used to get force field information from the ligand:

- Go to the CGENFF website: https://cgenff.paramchem.org/initguess/
- Input the *.mol2 file and save the *.str file that is outputted
- Copy *.mol2 file and *.str file, then run a code to prepare the structure. For example, for butanethiol, on the command line run: prep_charmm36_molecule butanethiol BUT
- Here, butanethiol is the ligand name, and BUT is the residue name.
- Then, send the molecule to the server using: send_swarm_charmm36_ligand butanethiol BUT
- Note: the "prep_charmm36_molecule" and "send_swarm_charmm36_ligand" command is available in bin/bashfiles/home_bashrc.sh. In addition, you may need to modify these commands to send the ligands to a correct server path by changing the "NETID" and "SWARM_SERVER" variables within "home_bashrc.sh" script.
- Now, your ligand should have ran energy minimization, assuming GROMACS is correctly loaded on the server.

3.2. Generating gold cores

Spherical and hollow gold cores were generated by python codes available in bin/MDBuilder/make_gold_spherical_111.py. Note that you may need to include MDBuilder within the python PATH. For embedded-atom models (EAM) obtained from variance-constrained semigrand-canonical simulations, refer to prep_system/prep_gold/vcsgc_simulations. To generate EAM model, the following procedure was used within the prep_system/prep_gold/vcsgc_simulations/scripts folder:

- Initialize vcsgc simulations by modifying the "prep_vcsgc.sh", setting an initial diameter of a sphere (should be large ~10 nm). Then, modify the phi based on number of atoms desired. This range should be large.
- We then use the "transfer_vcsgc.sh" to target a specific diameter. This will extract the simulations from "prep_vcsgc.sh" and run a separate simulation with a smaller phi increment.
- Finally, once the simulations are complete, run the "final_extract.sh" script, which finds the optimal diameter and runs a final energy minimization.

3.3. Running self-assembly simulations

Self-assembly of butanethiol around the gold core was done by running the script "prep_self_assembly_gold_ligand.sh" within the main scripts folder. For multiple gold core diameters, we used the "full_prep_self_assembly_gold_ligand.sh" code which has these tuning parameters that can generate multiple simulations when running "bash

full_prep_self_assembly_gold_ligand.sh". Tunable parameters are "shape_type" which can be "EAM" (FGC model), "spherical" (SGC model), or "hollow" (HGC model), "diameter" which can vary from 2-6 nm, and "trial_num" which are labels for the trial numbers.

3.4. Transfer ligand simulations

Once self-assembly simulations were generated, we can run transfer ligand simulations using the "transfer_ligand_builder.sh". These simulations assume that you have already prepared the ligand and ran the "prep_ligand_full_script.sh" script, see Section on "Preparing ligands". The remainder of this document is focused on procedures used to generate ligands and create nanoparticles from the available files.

For multiple core sizes and ligand types, use the "full_transfer_ligand_builder.sh", which simply for-loops the "transfer_ligand_builder.sh" script. These simulations by default run a 50 ns production simulation for each gold morphology ("shape_type"), trial ("trial_num"), core size ("diameter"), and ligand type ("ligand_names").

4. System analysis procedure

With these simulations, we can analyze the system by in-house analysis tools generated using the MDTraj toolbox and GROMACS functions. Herein, we describe how each of the outputs in the manuscript are generated. Note that after the simulations were complete, the production trajectory was extracted using the "analysis_full_extraction.sh". This script simply copies all production trajectory, gro file, and tpr file to another directory, called "analysis." Then the analysis directories were divided into "EAM", "hollow", or "spherical" directories related to the FGC, HGC, and SGC models reported in the main text.

4.1. Number of adsorbed ligands

The number of adsorbed ligands were computed by extracting the self-assembly simulations using the "analysis_full_extraction.sh" script. Note that you will need to switch the "want_self_assembled_monolayer" logical to "True" for self-assembly extraction. After running the "analysis_full_extraction.sh" script, move all the *EAM* to "EAM", *spherical* to "spherical" and *hollow* to "hollow" directories. After running this script, we need to use the scripts>analysis_scripts>multi_traj_analysis_tool.py to analyze the trajectories as follows:

- Within scripts>analysis_scripts>multi_traj_analysis_tool.py, change the "want_self_assembly_ligands" to True. Adjust the "Main_Directory_Parent_Path" accordingly. The variable "Categories" is a list that corresponds to "EAM", "hollow", or "spherical" directories.
- Run the analysis tool by running "python multi_traj_analysis_tool.py". This will output a PICKLE directory, with the full path being: scripts>analysis_scripts>PICKLE> self_assembly_structure>"DATE".
- To extract the information, go to bin>MDDescriptors>application>nanoparticle> extract_gmx_principal.py. Review the "extract_self_assembly_structure.py" script to ensure that you have the correct file paths. Then, you can run this code and it will output a csv file with the eccentricity values.

Note that we will be using "multi_traj_analysis_tool.py" as the main workhorse to extract information into PICKLES. Python PICKLE files are great for storing and extracting information quickly.

4.2. Fraction of *trans* dihedral angles

Fraction of *trans* dihedral angles was computed as follows:

- Within scripts>analysis_scripts>multi_traj_analysis_tool.py, change "want_nanoparticle_structure" to True. Adjust the "Main_Directory_Parent_Path" accordingly. The variable "Categories" is a list that corresponds to "EAM", "hollow", or "spherical" directories.
- Run the analysis tool by running "python multi_traj_analysis_tool.py". This will output a PICKLE directory, with the full path being: scripts>analysis_scripts>PICKLE> nanoparticle_structure>"DATE".
- To extract the information, go to bin>MDDescriptors>application>nanoparticle> extract_nanoparticle_structure.py. Review the "extract_nanoparticle_structure.py" script to ensure that you have the correct file paths. Then, you can run this code and it will output a csv file with the eccentricity values.

4.3. Eccentricity

Eccentricity was computed within the scripts>post_extraction.sh script. In "post_extraction.sh", all GROMACS functions are run on post-simulated results. For eccentricity, the "job_type" variable in the code was set to "CALC_GMX_PRINCIPAL" and ran over the analysis files. The outputs of *gmx principal* were placed within the output analysis directories. Then, the ensemble average of the principal axis were computed using a python script according to the following procedure:

- Within scripts>analysis_scripts>multi_traj_analysis_tool.py, change "want_analyze_gmx_principal" to True. Adjust the "Main_Directory_Parent_Path" accordingly. The variable "Categories" is a list that corresponds to "EAM", "hollow", or "spherical" directories.
- Run the analysis tool by running "python multi_traj_analysis_tool.py". This will output a PICKLE directory, with the full path being: scripts>analysis_scripts>PICKLE>analyze_gmx_principal>"DATE".
- To extract the information, go to bin>MDDescriptors>application>nanoparticle> extract_gmx_principal.py. Review the "extract_gmx_principal.py" script to ensure that you have the correct file paths. Then, you can run this code and it will output a csv file with the eccentricity values.

4.4. Number of bundling groups

The number of bundling groups were computed by using a clustering algorithm, outlined below:

- Within scripts>analysis_scripts>multi_traj_analysis_tool.py, change "want_nanoparticle_structure" to True. Adjust the "Main_Directory_Parent_Path" accordingly. The variable "Categories" is a list that corresponds to "EAM", "hollow", or "spherical" directories.
- Run the analysis tool by running "python multi_traj_analysis_tool.py". This will output a PICKLE directory, with the full path being: scripts>analysis_scripts>PICKLE> calc_nanoparticle_bundling_groups>"DATE".
- To extract the information, go to bin>MDDescriptors>application>nanoparticle> extract_nanoparticle_bundling.py. Review the "extract_nanoparticle_bundling.py" script

to ensure that you have the correct file paths. Then, you can run this code and it will output a csv file with the number of bundles per frame.

Note that the pickle files for the bundling groups may be used in subsequent calculations.

4.5. Fraction of ligands on facet for bundled and nonbundled ligands

The fraction of ligands on facet gold atoms were computed using the following procedure:

- Within scripts>analysis_scripts>multi_traj_analysis_tool.py, change "want_sulfur_gold_coordination" to True. Adjust the "Main_Directory_Parent_Path" accordingly. The variable "Categories" is a list that corresponds to "EAM", "hollow", or "spherical" directories.
- Run the analysis tool by running "python multi_traj_analysis_tool.py". This will output a PICKLE directory, with the full path being: scripts>analysis_scripts>PICKLE> self_assembly_sulfur_gold_coordination>"DATE".
- To extract the information, go to bin>MDDescriptors>application>nanoparticle> extract_gmx_principal.py. Review the "extract_self_assembly_sulfur_gold_coordination" script to ensure that you have the correct file paths. Then, you can run this code and it will output a csv file with information regarding the gold coordination and sulfur. We use another script, "extract_nanoparticle_sulfur_gold_coordination.py", which combines the results for the fraction of ligands on facet atoms and the bundling group characterization.

4.6. Fraction of *trans* dihedrals for bundled and nonbundled ligands

The fraction of *trans* dihedral angles for bundled and nonbundled ligands were done by combining the results from Section on "Fraction of *trans* dihedral angles" and "Number of bundling groups". We use the pickle results from these two computations and combine them, shown within "extract_nanoparticle_structure.py".

4.7. SASA for bundled and nonbundled ligands

The solvent-accessible surface area (SASA) of each ligand was computed as follows:

- Within scripts>post_extraction.sh, change the "job_type" to "NO_WATER_GOLD_CENTER". This will use the available trajectory files and remove all water, while centering the gold core at the center of the box. By removing water, we improve the performance of SASA calculations. After running the "post_extraction.sh" script, you should have *.gro and *.xtc with prefix "sam_prod_10_ns_whole_no_water_center".
- Within scripts>analysis_scripts>multi_traj_analysis_tool.py, change "want_ligand_sasa" to True. Adjust the "Main_Directory_Parent_Path" accordingly. The variable "Categories" is a list that corresponds to "EAM", "hollow", or "spherical" directories.
- Run the analysis tool by running "python multi_traj_analysis_tool.py". This will output a PICKLE directory, with the full path being: scripts>analysis_scripts>PICKLE> nanoparticle_sasa>"DATE".
- To extract the information, go to bin>MDDescriptors>application>nanoparticle> extract_nanoparticle_sasa.py. Review the "extract_nanoparticle_sasa.py" script to ensure that you have the correct file paths. Then, you can run this code and it will extract the SASA and correlate the values to either bundled or nonbundled groups.

5. Visualization software

All nanoparticles were visualized with the PyMol software. All the pymol scripts are available in bin>pymol_files. These scripts will use the PICKLES generated from the analysis procedure to color-code bundling groups and gold atom coordination.

5.1. Initialization

You can initialize the visualization tools as follows:

- Load the pymol_rc_home.sh by running "@PATH/pymol_rc_home.sh", where PATH is the path to your bin>pymol_files. Note that originally, the path was "~/bin/pymol_files", but you can change it however you please. It may be easier to simply create the original path and move the pymol_files directory there because all the other code is referenced to this directory.
- Upon loading the home script, you should refer to bin>pymol_files>scripts>np_global_vars.sh for global variables that you can change. The "DESCRIPTOR_CLASS_TO_DATES" contains all analysis pickles that were used for visualization. Update these with the correct pickle date as necessary.
- You can change the "main_analysis_dir" as necessary depending on how you name the analysis directory.
- Since PyMol requires a pdb to correctly load the file, generate a *.pdb file by running scripts>post_extraction.sh with "job_type" equal to "CENTER_XTC". This should be done before in Section "SASA for bundled and nonbundled ligands". If this was complete, skip this step. We will need "sam_prod_10_ns_whole_no_water_center.pdb" within each analysis directory to visualize the nanoparticle.
- Change the "category_dir" depending on whether you want the FGC model ("EAM"), SGC model ("spherical"), or HGC model ("hollow"). Adjust the "diameter" to your desired diameter and "ligand" to your desired ligand. You can also adjust "trial_num" for different trial numbers. In this work, we ran three trials, so this can vary from 1, 2, and 3.
- Once you have completed these adjustments, run the script by running on the pymol terminal:
  > np_reload
  > reloadkeys
  > np_main
- These two commands should reload your global variables and generate a gold core.

5.2. Visualization commands

You can run the following commands to visualize different parts of your system:

- > np_draw_ligands: draws the ligands in gray
- > np_color_bundling_groups: colors bundling group based on different colors
- > np_color_sasa: colors between red and blue colors of high and low SASA values, respectively
- > np_color_bundling_groups_bundled_vs_nonbundled: colors bundling groups as either black or gray, where black indicates bundled ligand and gray indicates nonbundled ligands
- > print_png("png_name.png"): prints the png in a high quality image.

6. Other notes

These codes are heavy on python. It may be useful to have a single module folder that contains "MDDescriptors" and "MDBuilder" directory. These files were originally at: "~/bin/pythonfiles/modules". You may consider copying these modules there and then adding them to your python PATH. In Bash, you can add python path by:

> export PYTHONPATH="${HOME}/bin/pythonfiles/modules:$PYTHONPATH"

Then, you can access these scripts throughout the cluster. Best of luck!