Project 3: Molecules and Computational Chemistry

Start Assignment

Due Monday by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip, ipynb, py, and html

In this project, you will compare the results of RDKit's automatic conformer generation and quantum chemical geometry optimization calculations to experimental structures.

1. Choose a molecule that has between 10-20 heavy atoms, and for which you can obtain a crystal structure from the CCDC (https://www.ccdc.cam.ac.uk/structures/?) . Make sure you're working with an isolated molecule, not a salt or aggregate. The molecule needs to contain a few rotatable bonds so that rdkit can generate at least 5 unique conformers. Download the crystal structure.
2. Look up the SMILES string for your molecule and import it with RDKit. Visualize the 2D structure and add hydrogen atoms as necessary.
3. Generate at least 5 unique conformers.
4. Import your experimental crystal structure, and align your conformers to it. Visualize the results, and compute the RMSD.
5. For each conformer you generated, perform a geometry optimization using psi4 at the B3LYP/6-31G* level of theory, and export the optimized structures to RDKit. Keep track of the optimized energy.
6. Align the optimized structures with the experimental one, and compute the RMSD.
7. Using the optimized structure, compute the energy of the molecule embedded in a solvent environment typical of an organic crystal, which typically has a dielectric constant of ~3. For this, you can choose benzene as a solvent model.
8. Create a Pandas dataframe listing each conformer, its RDKit-generated RMSD, the gas-phase B3LYP energy and RMSD, and the solvated energy. Rank the structures based on their energy (both gas-phase and solvated), and determine whether the lowest-energy structures give better agreement with the experimental structure.

Molecules and Computational Chemistry

Criteria	Ratings					Pts
Generate conformers using rdkit	**5 pts** **Excellent** Molecule chosen is 10-20 heavy atoms, and at least 5 unique confirmers are created using the rdkit library	**3 pts** **Fair** Molecule is chosen, but is too small, or too simple to produce the correct number of isomers.	**0 pts** **No Marks** No confirmers were produced.			5 pts
Import crystal structure/align confermers	**5 pts** **Excellent** All structures are visualized with the crystal structure. Hydrogen atoms are present. RMSD computed.	**3 pts** **Fair** Some errors in plotting the structures, but partially successful in comparing conformers to the crystal structure or RMSD not computed.	**0 pts** **No Marks** Crystal structure plot with confirmers not produced.			5 pts
Geometry calculations	**10 pts** **Full Marks** Each conformer has a geometry optimization run using the correct level of theory. Optimized energies reported. Optimized structures compared to crystal structure. Energy of molecule embedded in solvent calculated.	**8 pts** **Very Good** CSV file is produced, with close to the correct parameters for the gaussian fit, with minor errors. Plots are produced for the 6 largest peaks, but with minor flaws in the presentation.	**6 pts** **Good** No CSV file or no plots are produced, but the other is high quality.	**4 pts** **Fair** Major issues with the csv file and the 6 plots.	**0 pts** **No approval** No report or plots produced	10 pts
Final Report	**5 pts** **Excellent** Pandas dataframe with the conformers, pdkit generated RMSD, gas phase B3LYP energies and RMSDs, and the solvated energy created. Structures rankedbased on their energy.	**3 pts** **Very Good** Some errors in producing the pandas dataframe, it's not ranked by energy.	**0 pts** **No Marks** No Dataframe produced			5 pts

Criteria	Ratings						Pts
Execution How well does the python code run?	**10 pts** **Excellent** Code is well-written, making use of efficient routines for fast computation. Code runs and produces all required output without errors. Any warnings are explained/handled.	**8 pts** **Very Good** Code is well-written and mostly fast/efficient. No significant errors, and any warnings are explained/handled.	**6 pts** **Good** The code gets the job done, though could be improved by making use of more efficient coding patterns. Code may contain an error that is easy to fix (e.g., cells ordered incorrectly).	**4 pts** **Fair** Code is slow/inefficient, or the code contains mistakes that cause the output to be incorrect or unintended.	**2 pts** **Poor** Code is slow/inefficient and/or cannot be successfully run from beginning to end without significant errors.	**0 pts** **No Marks** Code does not run at all and contains major errors in construction.	10 pts
Readability/Documentation Note that code comments can be in the form of inline python comments or markdown annotations in a jupyter notebook (or a combination of the two)	**15 pts** **Excellent** Code is well-organized, easy to understand, and well-commented. Variables/functions/classes have names that make clear what their purpose is, and any especially tricky areas of the code include detailed comments that explain the intended function.	**10 pts** **Good** Most parts of the code are organized and easy to read for someone familiar with the project. Variables/classes/functions are named appropriately. Comments are mostly informative for explaining what is happening.	**5 pts** **Poor** Code is difficult to follow even for an experienced python programmer. Variables/functions/classes may have inappropriate names. Comments are sparse.	**0 pts** **No Marks** Code seems obfuscated and contains almost no comments.			15 pts
							Total Points: 50