

The general workflow for simulating a system is shown in Figure XX. (1) We create the molecule we want to investigate from smiles strings, using a program called mBuild. (2) We create our simulation object by adding the number of molecules we want to simulate, n, to a simulation box. (3) Next, we apply the forcefield parameters. (4) Conduct simulation until the system has reached equilibrium. While this workflow may appear straightforward, going from a SMILES string to an equilibrated morphology is extremely nontrivial. In this study we focus on bridging the gaps in step (3).

These parameters tell the simulation engine how our molecules should behave. They are grouped into bonded and non-bonded forces. The bonded forces include the bond, angle, and dihedral parameters. The non-bonded force parameters include sigma and epsilon. Sigma is the internuclear distance at which the potential energy of two particle is zero. Epsilon is the depth of the potential energy well; this governs the attractive forces between two particles.

We have created a tool that allows you to generate a forcefield file containing all the particle parameters needed to simulate a molecule, from a smiles string. To accomplish this, we created an mBuild compound of the molecule under investigation from a smiles string. In order to create an mBuild compound of larger, more complicated molecules, fragments of the molecule would be generated using smiles strings, and then the fragments would be pieced together by creating bonds using mBuild’s force overlap functionality. We utilized mBuild compounds because we were working espaloma into the MoSDeF workflow. Once the mBuild compound is created, we needed a function that would convert the mBuild compound into an Openmm Molecule format, which is needed for espaloma to generate a molecule graph.

There is not a direct conversion from an mBuild Compound to an openmm Molecule, so we created a function that uses an rdkit molecule as an intermediate, and then convert the rdkit molecule to an openmm Molecule object. In creating this function we observed that the bond order in our molecules was not being conserved when converting from mBuild to openmm. To combat this issue we created a function that uses a random walk algorithm to determine which atoms had a full octet and which atoms were missing higher order bonds. If the octet of an atom was complete, the algorithm walked to a neighboring atom and checked the octet of the neighbor atom. If two bonded atoms had an unsatisfied octet the function would increase the bond order by one, effectively adding a double bond. The function would continue this until every atom in the molecule had satisfied octets. After the algorithm had walked around the whole molecule, if an atom had a missing octet, meaning the double bonds were added in the incorrect order (see diagram XX), the bond orders would be reset to 1 and the algorithm would start over, walking in a different pattern until all the octets were met. The molecule object with a fully satisfied octet was then fed into the espaloma framework. Within this framework, espaloma creates a molecule graph from the openmm molecule input, uses a neural network to determine the chemical environment of a particle and assign an atom type to each unique particle in the molecule. Espaloma then uses this atom type/chemical environment to assign forcefield parameters to the molecule. Espaloma writes this information into an openmm system. We then wrote a function that would extract the bond, angle, dihedral and non-bonded particle information from this openmm system and write it each type into a python dictionary. These dictionaries were then fed into an xml writer function that would generate a forcefield file with an xml format that could be applied in a hoomd-blue MD simulation. The atoms in the mbuild compound that was created at the beginning of the process would then be renamed to match the atom types assigned by espaloma and saved to a mol2 file. This step aids us in applying the correct forcefield parameters to the corresponding particles in a simulation object.