# Module **parameterize**

QMMMReBind : Quantum Mechanics – Molecular Mechanics ( *QMMM* ) forcefield *R*eparamaterisation of the *Bind*ing site for the receptor-ligand complexes

## Functions

```
def OPLS_LJ(system)

def copy_file(source, destination)

def dihedral_energy(x, k1, k2, k3, k4=0)

def dot_product(u_PA, eig_AB)

def error_function(delta_qm, delta_mm)

def error_function_boltzmann(delta_qm, delta_mm, T)

def fit_params(qm_scan_file, load_topology, system_xml, method)

def force_angle_constant(atom_A, atom_B, atom_C, bond_lengths, eigenvalues,
                         eigenvectors, coords, scaling_1, scaling_2)

def force_angle_constant_special_case(atom_A, atom_B, atom_C, bond_lengths, eigenvalues,
                                      eigenvectors, coords, scaling_1, scaling_2)

def force_constant_bond(atom_A, atom_B, eigenvalues, eigenvectors, coords)

def gen_init_guess(qm_scan_file, load_topology, system_xml)

def generate_mm_pdbs(qm_scan_file, template_pdb)

def generate_xml_from_charged_pdb_sdf(system_pdb, system_init_sdf, system_sdf,
                                      num_charge_atoms, index_charge_atom_1,
                                      charge_atom_1, system_xml)
```

This function generates an openforcefield xml file from the pdb file via SDF file and openforcefield.

```
def generate_xml_from_pdb_sdf(system_pdb, system_sdf, system_xml)
```

This function generates an openforcefield xml file from the pdb file

```
def get_dihedrals(qm_scan_file)

def get_mm_potential_energies(qm_scan_file, load_topology, system_xml)

def get_non_torsion_mm_energy(system_pdb, load_topology, system_xml)

def get_qm_energies(qm_scan_file)

def get_tor_params(qm_scan_file, template_pdb, load_topology, system_xml, method)

def get_torsional_lines(template_pdb, system_xml, qm_scan_file, load_topology, method,
                        dihedral_text_file)

def list_diff(list_1, list_2)

def list_hartree_kcal(list_)

def list_kJ_kcal(list_)

def list_to_dict(lst)

def objective_function(k_array, x, delta_qm)

def remove_mm_files(qm_scan_file)

def reverse_list(lst)

def scale_list(list_)

def search_in_file(file: str, word: str) -> list
```

Search for the given string in file and return lines containing that string along with line numbers

```
def torsiondrive_input_to_xyz(psi_input_file, xyz_file)
```

```
def u_PA_from_angles(atom_A, atom_B, atom_C, coords)

def uniq(input_)

def unit_vector_N(u_BC, u_AB)

def xyz_to_pdb(xyz_file, coords_file, template_pdb, system_pdb)
```

# Classes

```
class GuestAmberXMLAmber (charge, num_charge_atoms, charge_atom_1, index_charge_atom_1,
                          system_pdb='guest_init_II.pdb', system_mol2='guest.mol2',
                          system_in='guest.in', system_frcmod='guest.frcmod',
                          prmtop_system='guest.prmtop', inpcrd_system='guest.inpcrd',
                          system_leap='guest.leap', system_xml='guest_init.xml',
                          system_smi='guest.smi', system_sdf='guest.sdf',
                          system_init_sdf='guest_init.sdf', index_charge_atom_2=' ',
                          charge_atom_2=' ', charge_parameter_file='guest_charges.txt',
                          system_qm_pdb='guest_init_II.pdb',
                          bond_parameter_file='guest_bonds.txt',
                          angle_parameter_file='guest_angles.txt',
                          system_qm_params_file='guest_qm_params.txt',
                          reparameterised_intermediate_system_xml_file='guest_intermedia
                          te_reparameterised.xml',
                          system_xml_non_bonded_file='guest_xml_non_bonded.txt',
                          system_xml_non_bonded_reparams_file='guest_xml_non_bonded_repa
                          rams.txt',
                          reparameterised_system_xml_file='guest_reparameterised.xml',
                          non_reparameterised_system_xml_file='guest_init.xml',
                          prmtop_system_non_params='guest_non_params.prmtop',
                          inpcrd_system_non_params='guest_non_params.inpcrd',
                          prmtop_system_params='guest_params.prmtop',
                          inpcrd_system_params='guest_params.inpcrd',
                          load_topology='openmm')
```

## Methods

```
def analyze_diff_energies(self)

def generate_xml_antechamber(self)
```

This function generates an xml file from the pdb file through antechamber

`def generate_xml_from_charged_pdb_sdf(self)`

This function generates an openforcefield xml file from the pdb file via SDF file and openforcefield.

`def generate_xml_from_doubly_charged_pdb_sdf(self)`

This function generates an openforcefield xml file from the pdb file via SDF file and openforcefield.

`def generate_xml_from_pdb_sdf(self)`

This function generates an openforcefield xml file from the pdb file

`def generate_xml_from_pdb_smi(self)`

This function generates an openforcefield xml file from the pdb file

`def save_amber_params(self)`

`def write_reparameterised_system_xml(self)`

`def write_system_params(self)`

This function saves the parameters obtained from the QM log files in a text file.

```
class HostAmberXMLAmber (system_pdb='host.pdb', system_xml='host.xml',
                         sim_output='sim_output.pdb', sim_steps=1000,
                         charge_parameter_file='host_qm_surround_charges.txt',
                         system_qm_pdb='host_qm.pdb',
                         bond_parameter_file='host_qm_bonds.txt',
                         angle_parameter_file='host_qm_angles.txt',
                         system_qm_params_file='host_qm_params.txt',
                         reparameterised_intermediate_system_xml_file='host_intermediate
                         _reparameterised.xml',
                         system_xml_non_bonded_file='host_xml_non_bonded.txt',
                         system_xml_non_bonded_reparams_file='host_xml_non_bonded_repara
                         ms.txt',
                         reparameterised_system_xml_file='host_reparameterised.xml',
                         non_reparameterised_system_xml_file='host.xml',
                         prmtop_system_non_params='host_non_params.prmtop',
```

```
                        inpcrd_system_non_params='host_non_params.inpcrd',
                        prmtop_system_params='host_params.prmtop',
                        inpcrd_system_params='host_params.inpcrd',
                        load_topology='openmm')
```

## Methods

```
def analyze_diff_energies(self)

def save_amber_params(self)

def serialize_system(self)

def write_reparameterised_system_xml(self)

def write_system_params(self)
```

This function saves the parameters obtained from the QM log files in a text file.

```
class MergeHostGuestTopology (host_prmtop, guest_prmtop, host_inpcrd, guest_inpcrd,
                              system_prmtop, system_inpcrd)
```

## Methods

```
def merge_topology_files(self)
```

```
class ParameterizeGuest (vibrational_scaling, xyz_file='guest_coords.xyz',
                         coordinate_file='guest_coordinates.txt',
                         unprocessed_hessian_file='guest_unprocessed_hessian.txt',
                         bond_list_file='guest_bond_list.txt',
                         angle_list_file='guest_angle_list.txt',
                         hessian_file='guest_hessian.txt',
                         atom_names_file='guest_atom_names.txt',
                         bond_parameter_file='guest_bonds.txt',
                         angle_parameter_file='guest_angles.txt',
                         charge_parameter_file='guest_charges.txt',
                         guest_pdb='guest_init_II.pdb',
                         proper_dihedral_file='proper_dihedrals.txt')
```

## Methods

**def get_atom_names(self)**

This function saves a list of atom names from the formatted checkpoint file.

**def get_bond_angle_params(self)**

This function saves the bond and angle parameter files obtained from the formatted checkpoint file.

**def get_bond_angles(self)**

This function saves a text file of the bonds and angles from the gaussian log file.

**def get_charges(self)**

This function saves the charges in a text file obtained from the Gaussian log file.

**def get_hessian(self)**

This function extracts hessian from the unprocessed hessian and saves into a new file.

**def get_proper_dihedrals(self)**

This function saves proper dihedral angles of the guest molecule in a text file.

**def get_unprocessed_hessian(self)**

This function saves a text file of the unprocessed hessian from the formatted checkpoint file.

**def get_xyz(self)**

This function saves a xyz file from the formatted checkpoint file.

```
class ParameterizeHost (vibrational_scaling, xyz_file='host_qm_coords.xyz',
                        coordinate_file='host_qm_coordinates.txt',
                        unprocessed_hessian_file='host_qm_unprocessed_hessian.txt',
                        bond_list_file='host_qm_bond_list.txt',
                        angle_list_file='host_qm_angle_list.txt',
                        hessian_file='host_qm_hessian.txt',
                        atom_names_file='host_qm_atom_names.txt',
                        bond_parameter_file='host_qm_bonds.txt',
```

```
                    angle_parameter_file='host_qm_angles.txt',
                    charge_parameter_file='host_qm_surround_charges.txt',
                    host_qm_pdb='host_qm.pdb')
```

## Methods

**def get_atom_names(self)**

This function saves a list of atom names from the formatted checkpoint file.

**def get_bond_angle_params(self)**

This function saves the bond and angle parameter files obtained from the formatted checkpoint file.

**def get_bond_angles(self)**

This function saves a text file of the bonds and angles from the gaussian log file.

**def get_charges(self)**

This function saves the charges in a text file obtained from the Gaussian log file.

**def get_hessian(self)**

This function extracts hessian from the unprocessed hessian and saves into a new file.

**def get_unprocessed_hessian(self)**

This function saves a text file of the unprocessed hessian from the formatted checkpoint file.

**def get_xyz(self)**

This function saves a xyz file from the formatted checkpoint file.

```
class PrepareGaussianGuest (charge, multiplicity, guest_pdb='guest_init_II.pdb',
                            n_processors=12, memory=50, functional='B3LYP',
                            basis_set='6-31G', optimisation='OPT', frequency='FREQ',
                            add_keywords_I='Integral=(Grid=UltraFine)',
                            add_keywords_II='Pop(MK,ReadRadii)',
                            add_keywords_III='IOp(6/33=2,6/42=6)',
                            gauss_out_file='guest.out', fchk_out_file='guest_fchk.out')
```

## Methods

**def get_fchk(self)**

This function converts the checkpoint file file into the formatted chechkpoint file.

**def run_gaussian(self)**

This function runs the gaussian QM calculation.

**def write_input(self)**

This function prints out the commands section of the gaussian input file.

```
class PrepareGaussianHost (charge, multiplicity, host_qm_pdb='host_qm.pdb',
                           n_processors=12, memory=50, functional='B3LYP', basis_set='6-
                           31G', optimisation='OPT', frequency='FREQ',
                           add_keywords_I='Integral=(Grid=UltraFine)',
                           add_keywords_II='Pop(MK,ReadRadii)',
                           add_keywords_III='IOp(6/33=2,6/42=6)',
                           gauss_out_file='host_qm.out',
                           fchk_out_file='host_qm_fchk.out')
```

## Methods

**def get_fchk(self)**

This function converts the checkpoint file file into the formatted chechkpoint file.

**def run_gaussian(self)**

This function runs the gaussian QM calculation.

**def write_input(self)**

This function prints out the commands section of the gaussian input file.

```
class PrepareGaussianHostGuest (charge, multiplicity, guest_pdb='guest_init_II.pdb',
                                host_qm_pdb='host_qm.pdb', n_processors=12, memory=50,
                                functional='B3LYP', basis_set='6-31G', optimisation='',
```

```
                                frequency='', add_keywords_I='Integral=
                                (Grid=UltraFine)', add_keywords_II='Pop(MK,ReadRadii)',
                                add_keywords_III='IOp(6/33=2,6/42=6)',
                                gauss_system_out_file='system_qm.out',
                                fchk_system_out_file='system_qm_fchk.out',
                                host_guest_input='host_guest.com',
                                qm_guest_charge_parameter_file='guest_qm_surround_charge
                                s.txt',
                                qm_host_charge_parameter_file='host_qm_surround_charges.
                                txt',
                                qm_guest_atom_charge_parameter_file='guest_qm_atom_surro
                                und_charges.txt')
```

## Methods

**def get_fchk(self)**

This function converts the checkpoint file file into the formatted chechkpoint file.

**def get_qm_host_guest_charges(self)**

This function extracts charges and saves them separately for the host and guest

**def run_gaussian(self)**

This function runs the gaussian QM calculation.

**def write_input(self)**

This function prints out the commands section of the gaussian input file.

```
class PrepareQMMM (init_pdb, distance, num_residues, guest_resname,
                   cleaned_pdb='system.pdb', guest_init_pdb='guest_init.pdb',
                   host_pdb='host.pdb', guest_pdb='guest_init_II.pdb',
                   guest_xyz='guest_coord.txt', residue_list='residue_list.txt',
                   host_qm_atoms='host_qm.txt', host_mm_atoms='host_mm.txt',
                   host_qm_pdb='host_qm.pdb', host_mm_pdb='host_mm.pdb',
                   qm_pdb='qm.pdb', mm_pdb='mm.pdb',
                   host_mm_region_I_atoms='host_mm_region_I.txt',
                   host_mm_region_II_atoms='host_mm_region_II.txt',
                   host_mm_region_I_pdb='host_mm_region_I.pdb',
                   host_mm_region_II_pdb='host_mm_region_II.pdb')
```

A class used to segregate the QM and MM regions.

This class contains methods to remove the solvent, ions and all entities that are exclusive of receptor and the ligand. It also defines the Quantum Mechanical (QM) region and the Molecular Mechanical (MM) region based upon the distance of the ligand from the receptor and the chosen number of receptor residues. It is also assumed that the initial PDB file will have the receptor followed by the ligand.

...

## Attributes

**init_pdb** : `str`
Initial PDB file containing the receptor-ligand complex with solvent, ions, etc.

**cleaned_pdb** : `str`
Formatted PDB file containing only the receptor and the ligand. (This file will be saved in the current working directory)

**guest_init_pdb** : `str`
A separate ligand PDB file with atom numbers not beginning from 1. (This file will be saved in the current working directory)

**host_pdb** : `str`
A separate receptor PDB file with atom numbers beginning from 1.

**guest_resname** : `str`
Three letter residue ID for the ligand

**guest_pdb** : `str`
Ligand PDB file with atom numbers beginning from 1. (This file will be saved in the current working directory)

**guest_xyz** : `str`
A text file of the XYZ corordinates of the ligand. (This file will be saved in the current working directory)

**distance** : `float`
The distance required to define the QM region of the receptor. This is the distance between the atoms of the ligand and the atoms of the receptor.

**residue_list** : `str`
A text file of the residue numbers of the receptor within the proximity (as defined by the distance) from the ligand. (This file will be saved in the current working directory)

**host_qm_atoms** : `str`

A text file of the atom numbers of the receptors in the QM region. (This file will be saved in the current working directory)

**host_mm_atoms** : `str`
A text file of the atom numbers of the receptors in the MM region (all atoms except atoms in the QM region) (This file will be saved in the current working directory)

**host_qm_pdb** : `str`
PDB file for the receptor's QM region. (This file will be saved in the current working directory)

**host_mm_pdb** : `str`
PDB file for the receptor's MM region. (This file will be saved in the current working directory)

**qm_pdb** : `str`
PDB file for the QM region (receptor's QM region and the ligand). (This file will be saved in the current working directory)

**mm_pdb** : `str`
PDB file for the MM region. (This file will be saved in the current working directory)

**host_mm_region_I_atoms** : `str`
A text file of the atom numbers of the receptors in the MM region preceeding the QM region. (This file will be saved in the current working directory)

**host_mm_region_II_atoms** : `str`
A text file of the atom numbers of the receptors in the MM region following the QM region. (This file will be saved in the current working directory)

**host_mm_region_I_pdb** : `str`
PDB file of the receptor in the MM region preceeding the QM region. (This file will be saved in the current working directory)

**host_mm_region_II_pdb** : `str`
PDB file of the receptor in the MM region following the QM region. (This file will be saved in the current working directory)

**num_residues** : `int`
Number of residues required in the QM region of the receptor.

## Parameters

**init_pdb** : `str`

Initial PDB file containing the receptor-ligand complex with solvent, ions, etc.

**cleaned_pdb** : `str`
Formatted PDB file containing only the receptor and the ligand. (This file will be saved in the current working directory)

**guest_init_pdb** : `str`
A separate ligand PDB file with atom numbers not beginning from 1. (This file will be saved in the current working directory)

**host_pdb** : `str`
A separate receptor PDB file with atom numbers beginning from 1.

**guest_resname** : `str`
Three letter residue ID for the ligand.

**guest_pdb** : `str`
Ligand PDB file with atom numbers beginning from 1. (This file will be saved in the current working directory)

**guest_xyz** : `str`
A text file of the XYZ corordinates of the ligand. (This file will be saved in the current working directory)

**distance** : `float`
The distance required to define the QM region of the receptor. This is the distance between the atoms of the ligand and the atoms of the receptor.

**residue_list** : `str`
A text file of the residue numbers of the receptor within the proximity (as defined by the distance) from the ligand. (This file will be saved in the current working directory)

**host_qm_atoms** : `str`
A text file of the atom numbers of the receptors in the QM region. (This file will be saved in the current working directory)

**host_mm_atoms** : `str`
A text file of the atom numbers of the receptors in the MM region (all atoms except atoms in the QM region) (This file will be saved in the current working directory)

**host_qm_pdb** : `str`
PDB file for the receptor's QM region. (This file will be saved in the current working directory)

**host_mm_pdb** : `str`
PDB file for the receptor's MM region. (This file will be saved in the current working directory)

**qm_pdb** : `str`
PDB file for the QM region (receptor's QM region and the ligand). (This file will be saved in the current working directory)

**mm_pdb** : `str`
PDB file for the MM region. (This file will be saved in the current working directory)

**host_mm_region_I_atoms** : `str`
A text file of the atom numbers of the receptors in the MM region preceeding the QM region. (This file will be saved in the current working directory)

**host_mm_region_II_atoms** : `str`
A text file of the atom numbers of the receptors in the MM region following the QM region. (This file will be saved in the current working directory)

**host_mm_region_I_pdb** : `str`
PDB file of the receptor in the MM region preceeding the QM region. (This file will be saved in the current working directory)

**host_mm_region_II_pdb** : `str`
PDB file of the receptor in the MM region following the QM region. (This file will be saved in the current working directory)

**num_residues** : `int`
Number of residues required in the QM region of the receptor.

## Methods

**def clean_up(self)**

Reads the given PDB file, removes all entities except the receptor and ligand and saves a new pdb file.

**def create_host_guest(self)**

Saves separate receptor and ligand PDB files.

**def get_guest_coord(self)**

Saves a text file of the XYZ corordinates of the ligand.

#### def get_host_mm_region_atoms(self)

Saves a text file for the atoms of the receptor's MM region preceding the QM region and saves another text file for the atoms of the receptor's MM region folllowing the QM region.

#### def get_host_qm_mm_atoms(self)

Saves a text file of the atom numbers of the receptors in the QM region and MM region separately.

#### def get_qm_mm_regions(self)

Saves separate PDB files for the QM and MM regions. QM regions comprise the QM region of the receptor and the entire ligand where the MM region comprise the non-selected QM regions of the receptor.

#### def get_qm_resids(self)

Saves a text file of the residue numbers of the receptor within the proximity (as defined by the distance) from the ligand.

#### def realign_guest(self)

Saves a ligand PDB file with atom numbers beginning from 1.

#### def save_host_mm_regions_pdbs(self)

Saves a PDB file for the receptor's MM region preceding the QM region and saves another PDB file for the receptor's MM region folllowing the QM region.

#### def save_host_pdbs(self)

Saves a PDB file for the receptor's QM region and MM region separately.

## class RunOpenMMSims (system_prmtop, system_inpcrd, system_pdb, system_output='sim_output.pdb', sim_steps=1000)

### Methods

#### def run_openmm_prmtop_inpcrd(self)

```
    def run_openmm_prmtop_pdb(self)
```

```
class TorsionDriveParams (num_charge_atoms, index_charge_atom_1, charge_atom_1,
                          tor_dir='torsion_dir',
                          reparameterized_torsional_params_file='reparameterized_torsion
                          al_params.txt', psi_input_file='torsion_drive_input.dat',
                          xyz_file='torsion_drive_input.xyz',
                          coords_file='torsion_drive_input.txt',
                          template_pdb='guest_init_II.pdb',
                          system_pdb='torsion_drive_input.pdb',
                          system_sdf='torsion_drive_input.sdf',
                          system_xml='torsion_drive_input.xml', qm_scan_file='scan.xyz',
                          load_topology='openmm', method='L-BFGS-B',
                          dihedral_text_file='dihedrals.txt',
                          system_init_sdf='torsion_drive_input_init.sdf',
                          reparameterised_system_xml_file='guest_reparameterised.xml',
                          reparameterised_torsional_system_xml_file='guest_torsional_rep
                          arameterized.xml')
```

## Methods

```
    def write_reparams_torsion_lines(self)

    def write_reparams_torsion_lines_charged(self)

    def write_torsional_reparams(self)
```

```
class TorsionDriveSims (charge, multiplicity,
                        reparameterised_system_xml_file='guest_reparameterised.xml',
                        torsion_xml_file='guest_torsion_xml.txt',
                        xyz_file='guest_coords.xyz',
                        psi_input_file='torsion_drive_input.dat', memory=50,
                        basis_set='STO-3G', functional='BLYP', iterations=2000,
                        method_torsion_drive='native_opt',
                        system_bonds_file='guest_bonds.txt', tor_dir='torsion_dir',
                        dihedral_text_file='dihedrals.txt',
                        template_pdb='guest_init_II.pdb',
```

```
                    torsion_drive_run_file='run_command', dihedral_interval=15,
                    engine='psi4', energy_threshold=0.001)
```

## Methods

```
def create_non_H_bonded_torsion_drive_dir(self)

def create_non_H_torsion_drive_dir(self)

def create_torsion_drive_dir(self)

def run_torsion_sim(self)

def write_psi4_input(self)

def write_tor_params_txt(self)

def write_torsion_drive_run_file(self)
```

# Index

## Functions

# Classes

**GuestAmberXMLAmber**

analyze_diff_energies
generate_xml_antechamber
generate_xml_from_charged_pdb_sdf
generate_xml_from_doubly_charged_pdb_sdf
generate_xml_from_pdb_sdf
generate_xml_from_pdb_smi
save_amber_params
write_reparameterised_system_xml
write_system_params

**HostAmberXMLAmber**

analyze_diff_energies
save_amber_params
serialize_system
write_reparameterised_system_xml
write_system_params

**MergeHostGuestTopology**

merge_topology_files

**ParameterizeGuest**

get_atom_names
get_bond_angle_params
get_bond_angles
get_charges
get_hessian
get_proper_dihedrals
get_unprocessed_hessian
get_xyz

**ParameterizeHost**

get_atom_names
get_bond_angle_params
get_bond_angles
get_charges
get_hessian
get_unprocessed_hessian
get_xyz

**PrepareGaussianGuest**

get_fchk
run_gaussian

write_input

**PrepareGaussianHost**
get_fchk
run_gaussian
write_input

**PrepareGaussianHostGuest**
get_fchk
get_qm_host_guest_charges
run_gaussian
write_input

**PrepareQMMM**
clean_up
create_host_guest
get_guest_coord
get_host_mm_region_atoms
get_host_qm_mm_atoms
get_qm_mm_regions
get_qm_resids
realign_guest
save_host_mm_regions_pdbs
save_host_pdbs

**RunOpenMMSims**
run_openmm_prmtop_inpcrd
run_openmm_prmtop_pdb

**TorsionDriveParams**
write_reparams_torsion_lines
write_reparams_torsion_lines_charged
write_torsional_reparams

**TorsionDriveSims**
create_non_H_bonded_torsion_drive_dir
create_non_H_torsion_drive_dir
create_torsion_drive_dir
run_torsion_sim
write_psi4_input
write_tor_params_txt
write_torsion_drive_run_file