

NAME

OpenMMUtil

SYNOPSIS

import OpenMMUtil

DESCRIPTION

OpenMMUtil module provides the following functions:

AddWaterBox, DoAtomListsOverlap, DoesAtomListOverlapWithSystemConstraints, DoesSystemContainWater, DoesSystemUsesPeriodicBoundaryConditions, FixColumnNamesLineInDataLogFile, FreezeAtoms, GenerateReimagedRealignedTrajectoryFiles, GetAtoms, GetFormattedTotalSimulationTime, InitializeBarostat, InitializeIntegrator, InitializeReporters, InitializeSimulation, InitializeSystem, InitializeSystemGenerator, MapDataOutTypePlotToDataLogColumnNameNames, MergeSmallMoleculeWithMacromolecule, PerformAnnealing, ProcessOptionOpenMMAnnealingParameters, ProcessOptionOpenMMAtomsSelectionParameters, ProcessOptionOpenMMForcefieldParameters, ProcessOptionOpenMMIntegratorParameters, ProcessOptionOpenMMMDProtocolParameters, ProcessOptionOpenMMOutputParameters, ProcessOptionOpenMMPlatformParameters, ProcessOptionOpenMMRestartParameters, ProcessOptionOpenMMSimulationParameters, ProcessOptionOpenMMSystemParameters, ProcessOptionOpenMMWaterBoxParameters, ReadPDBFile, ReadSmallMoleculeFile, ReimageRealignTrajectory, RestrictAtoms, SetupAnnealingParameters, SetupIntegratorParameters, SetupMDProtocolParameters, SetupPlatformParameters, SetupSimulationParameters, SetupSystemGeneratorForcefieldsParameters, ValidateAndFreezeRestrictAtoms, WritePDBFile, WriteSimulationStatePDBFile

FUNCTIONS**AddWaterBox**

```
AddWaterBox(ModellerHandle, SystemGeneratorHandle, WaterBoxParamsInfo)
```

Add a water box.

The WaterBoxParamsInfo parameter is a dictionary of name and value pairs for waterbox parameters and may be generated by calling the function named ProcessOptionOpenMMWaterBoxParameters().

Arguments:

ModellerHandle (object): OpenMM modeller object.
 SystemGeneratorHandle (object): OpenMM system generator object.
 WaterBoxParamsInfo (dict): Parameter name and value pairs.

Returns:

None.

Example(s):

```
OptionsInfo["WaterBoxParams"] =
    OpenMMUtil.ProcessOptionOpenMMWaterBoxParameters("--waterBoxParams",
    Options["--waterBoxParams"])
... ..
OpenMMUtil.AddWaterBox(ModellerHandle, SystemGeneratorHandle,
    OptionsInfo["WaterBoxParams"])
```

DoAtomListsOverlap

```
DoAtomListsOverlap(AtomList1, AtomList2)
```

Check for the overlap of atoms in the specified atom lists.

Arguments:

AtomList1 (list): List of OpenMM atom objects.
 AtomList2 (list): List of OpenMM atom objects.

Returns:

bool : True - Overlap between atoms lists; Otherwise, false.

DoesAtomListOverlapWithSystemConstraints

```
DoesAtomListOverlapWithSystemConstraints(System, AtomList)
```

Check for the overlap of specified atoms with the atoms involved in system constraints.

Arguments:

System (object): OpenMM system object.
AtomList (list): List of OpenMM atom objects.

Returns:

bool : True - Overlap with system constraints; Otherwise, false.

DoesSystemContainWater

DoesSystemContainWater(Topology, WaterResidueNames=["HOH"])

Check for the presence of water residues in a system.

Arguments:

Topology (object): OpenMM modeller topology object.
WaterResidueNames (list): List of water residue names.

Returns:

bool : True - Contains water; Otherwise, false.

DoesSystemUsesPeriodicBoundaryConditions

DoesSystemUsesPeriodicBoundaryConditions(System)

Check for the use of periodic boundary conditions in a system.

Arguments:

System (object): OpenMM system object.

Returns:

bool : True - Uses periodic boundary conditions; Otherwise, false.

FixColumnNamesLineInDataLogFile

FixColumnNamesLineInDataLogFile(DataLogFile)

Fix column name line in data log file by removing the # character written by OpenMM at the start of the first line.

Arguments:

DataLogFile (str): Data log file name.

Returns:

None

FreezeAtoms

FreezeAtoms(System, AtomList)

Freeze atoms during a simulation. The specified atoms are kept completely fixed by setting their masses to zero. Their positions do not change during local energy minimization and MD simulation, and they do not contribute to the kinetic energy of the system.

Arguments:

System (object): OpenMM system object.
AtomList (list): List of OpenMM atom objects.

Returns:

None

GenerateReimagedRealignedTrajectoryFiles

GenerateReimagedRealignedTrajectoryFiles(System, Topology, TrajTopologyFile, ReimagedPDBOutfile, ReimagedTrajOutfile, OutputParamsInfo, RealignFrames=True)

Reimage and realign a trajectory file using MDTraj. The trajectory frames are reimaged before realigning to the first frame using the specified atom selection.

The trajectory file format must a valid format supported by MDTraj. No validation is performed.

The OutputParamsInfo is a dictionary of name and value pairs for output parameters and may be generated by calling the function named ProcessOptionOpenMMOutputParameters().

Arguments:

System (object): OpenMM system object.
 Topology (object): OpenMM topology object.
 TrajTopologyFile (str): Trajectory PDB topology file name.
 ReimagedPDBOutfile (str): Trajectory PDB out file name.
 ReimagedTrajOutfile (str): Trajectory out file name.
 OutputParamsInfo (dict): Parameter name and value pairs.
 RealignFrames (bool): Realign trajectory frames.

Returns:

None

GetAtoms

GetAtoms(Topology, CAlphaProteinStatus, ResidueNames, Negate)

Get a list of atoms in the specified residue names. You may set CAlphaProteinStatus flag to True to only retrieve CAlpha atoms from the residues. In addition, you may negate residue name match using Negate flag.

Arguments:

Topology (object): OpenMM topology object.
 CAlphaProteinStatus (bool): Get CAlpha atoms only.
 ResidueNames (list): List of residue names.
 Negate (bool): Negate residue name match.

Returns:

None or List of OpenMM atom objects.

GetFormattedTotalSimulationTime

GetFormattedTotalSimulationTime(StepSize, Steps)

Get formatted total simulation time with appropriate time units. parameter names and values.

Arguments:

StepSize (object): OpenMM quantity object.
 Steps (int): Number of steps.

Returns:

str: Total time.

InitializeBarostat

InitializeBarostat(ParamsInfo)

Initialize barostat.

The ParamsInfo parameter is a dictionary of name and value pairs for integrator parameters and may be generated by calling the function named ProcessOptionOpenMMIntegratorParameters().

Arguments:

ParamsInfo (dict): Parameter name and value pairs.

Returns:

Object: OpenMM barostat object.

Example(s):

```
OptionsInfo["IntegratorParams"] =
    OpenMMUtil.ProcessOptionOpenMMIntegratorParameters(
        "--integratorParams", Options["--integratorParams"],
        HydrogenMassRepartitioningStatus =
            OptionsInfo["SystemParams"]["HydrogenMassRepartitioning"])
... ..
Barostat = OpenMMUtil.InitializeBarostat(
    OptionsInfo["IntegratorParams"])
```

InitializeIntegrator

```
InitializeIntegrator(ParamsInfo, ConstraintErrorTolerance)
```

Initialize integrator.

The ParamsInfo parameter is a dictionary of name and value pairs for integrator parameters and may be generated by calling the function named ProcessOptionOpenMMIntegratorParameters().

Arguments:

ParamsInfo (dict): Parameter name and value pairs.
 ConstraintErrorTolerance (float): Distance tolerance for constraints as a fraction of the constrained distance.

Returns:

Object: OpenMM integrator object.

Example(s):

```
OptionsInfo["IntegratorParams"] =
    OpenMMUtil.ProcessOptionOpenMMIntegratorParameters(
        "--integratorParams", Options["--integratorParams"],
        HydrogenMassRepartitioningStatus =
        OptionsInfo["SystemParams"]["HydrogenMassRepartitioning"])
... ..
Integrator = OpenMMUtil.InitializeIntegrator(
    OptionsInfo["IntegratorParams"],
    OptionsInfo["SystemParams"]["ConstraintErrorTolerance"])
```

InitializeReporters

```
InitializeReporters(OutputParamsInfo, TotalSteps, DataOutAppendStatus)
```

Initialize reporters for writing data to trajectory, log, and checkpoint files along with reporting to the stdout.

The OutputParamsInfo parameter is a dictionary of name and value pairs for output parameters and may be generated by calling the function named ProcessOptionOpenMMOutputParameters().

Arguments:

OutputParamsInfo (dict): Parameter name and value pairs.
 TotalSteps (int): Total number of simulation steps.
 DataOutAppendStatus (bool): Append data to trajectory and log file.

Returns:

Object: OpenMM trajectory reporter object.
 Object: OpenMM data log file reporter object.
 Object: OpenMM stdout reporter object.
 Object: OpenMM checkpoint reporter object.

Example(s):

```
if OptionsInfo["NVTMode"]:
    ParamsDefaultInfoOverride = {"DataOutType": "Step Speed Progress
    PotentialEnergy Temperature Time Volume"}
else:
    ParamsDefaultInfoOverride = {"DataOutType": "Step Speed Progress
    PotentialEnergy Temperature Time Density"}
OptionsInfo["OutputParams"] =
    OpenMMUtil.ProcessOptionOpenMMOutputParameters("--outputParams",
    Options["--outputParams"], OptionsInfo["OutfilePrefix"],
    ParamsDefaultInfoOverride)
ProcessOutfileNames()
... ..
(TrajReporter, DataLogReporter, DataStdoutReporter, CheckpointReporter)
= OpenMMUtil.InitializeReporters(OptionsInfo["OutputParams"],
    OptionsInfo["SimulationParams"]["Steps"], OptionsInfo["DataOutAppendMode"])
```

InitializeSimulation

```
InitializeSimulation(System, Integrator, Topology, Positions, PlatformParamsInfo)
```

Initialize simulation.

The PlatformParamsInfo parameter is a dictionary of name and value pairs for platform parameters and may be generated by calling the function named ProcessOptionOpenMMPlatformParameters().

Arguments:

System (object): OpenMM system object.
 Integrator (object): OpenMM integrator object.
 Topology (object): OpenMM topology object.
 Positions (object): OpenMM Positions object.
 PlatformParamsInfo (dict): Parameter name and value pairs.

Returns:

Object: OpenMM simulation object.

Example(s):

```
ParamsDefaultInfoOverride = {"Name": Options["--platform"],
                             "Threads": 1}
OptionsInfo["PlatformParams"] =
    OpenMMUtil.ProcessOptionOpenMMPlatformParameters("--platformParams",
    Options["--platformParams"], ParamsDefaultInfoOverride)
... ..
Simulation = OpenMMUtil.InitializeSimulation(System, Integrator, Topology,
    Positions, OptionsInfo["PlatformParams"])
```

InitializeSystem

```
InitializeSystem(PDBFile, ForcefieldParamsInfo, SystemParamsInfo,
WaterBoxAdd=False, WaterBoxParamsInfo=None, SmallMolFile=None, SmallMolID="LIG")
```

Initialize OpenMM system using specified forcefields, system and water box parameters along with a small molecule file and ID.

The ForcefieldParamsInfo parameter is a dictionary of name and value pairs for system parameters and may be generated by calling the function named ProcessOptionOpenMMForcefieldParameters().

The SystemParamsInfo parameter is a dictionary of name and value pairs for system parameters and may be generated by calling the function named ProcessOptionOpenMMSystemParameters().

The WaterBoxParamsInfo parameter is a dictionary of name and value pairs for system parameters and may be generated by calling the function named ProcessOptionOpenMMWaterBoxParameters().

Arguments:

PDBFile (str): PDB file name..
 ForcefieldParamsInfo (dict): Parameter name and value pairs.
 SystemParamsInfo (dict): Parameter name and value pairs.
 WaterBoxAdd (bool): Add water box.
 WaterBoxParamsInfo (dict): Parameter name and value pairs.
 SmallMolFile (str): Small molecule file name.
 SmallMolID (str): Three letter small molecule ID.

Returns:

Object: OpenMM system object.
 Object: OpenMM topology object.
 Object: OpenMM positions object.

Example(s):

```
... ..
OptionsInfo["ForcefieldParams"] =
    OpenMMUtil.ProcessOptionOpenMMForcefieldParameters(
    "--forcefieldParams", Options["--forcefieldParams"])
... ..
OptionsInfo["SystemParams"] =
    OpenMMUtil.ProcessOptionOpenMMSystemParameters("--systemParams",
    Options["--systemParams"])
... ..
OptionsInfo["WaterBoxParams"] =
    OpenMMUtil.ProcessOptionOpenMMWaterBoxParameters(
    "--waterBoxParams", Options["--waterBoxParams"])
... ..
```

```
System, Topology, Positions = OpenMMUtil.InitializeSystem(
    OptionsInfo["Infile"], OptionsInfo["ForcefieldParams"],
    OptionsInfo["SystemParams"], OptionsInfo["WaterBox"],
    OptionsInfo["WaterBoxParams"], OptionsInfo["SmallMolFile"],
    OptionsInfo["SmallMolID"])
```

InitializeSystemGenerator

```
InitializeSystemGenerator(BiopolymerForcefield, SmallMoleculeForcefield,
    WaterForcefield, SystemParamsInfo, SmallMols, AdditionalForcefieldsList=None)
```

Initialize MMFF system generator using specified forcefields and system parameters along with a list of molecules.

The SystemParamsInfo parameter is a dictionary of name and value pairs for system parameters and may be generated by calling the function named ProcessOptionOpenMMSystemParameters().

Arguments:

```
BiopolymerForcefield (str): Biopolymer force field name.
SmallMoleculeForcefield (str): Small molecule force field name.
WaterForcefield (str): Water force field name.
SystemParamsInfo (dict): Parameter name and value pairs.
SmallMols (list): List of OpenFF toolkit molecule objects.
AdditionalForcefieldsList (list): List of any additional forcefield
    names
```

Returns:

```
Object: MMFF system generator object.
```

Example(s):

```
OptionsInfo["SystemParams"] =
    OpenMMUtil.ProcessOptionOpenMMSystemParameters("--systemParams",
    Options["--systemParams"])
... ..
SystemGeneratorHandle = OpenMMUtil.InitializeSystemGenerator(
    BiopolymerForcefield, SmallMoleculeForcefield, WaterForcefield,
    OptionsInfo["SystemParams"], SmallMols, AdditionalForcefieldsList)
```

MapDataOutTypePlotToDataLogColumnNames

```
MapDataOutTypePlotToDataLogColumnNames(DataOutTypePlotList, DataLogColNames)
```

Map data out type to be plotted to column names in data log file.

Arguments:

```
DataOutTypePlotList (list): List of data out types.
DataLogColNames (list): List of data log file column names.
```

Returns:

```
dict: Data out type and column name value pairs.
```

MergeSmallMoleculeWithMacromolecule

```
MergeSmallMoleculeWithMacromolecule(ModellerHandle, SmallMol, SmallMolID="LIG")
```

Merge small molecule with macromolecule data contained in a modeller object and assign a three letter small molecule residue name to the merged small molecule.

Arguments:

```
ModellerHandle (object): OpenMM modeller object.
SmallMol (object): OpenFF toolkit molecule object.
SmallMolID (str): Three letter residue name for small molecule.
```

Returns:

```
None
```

PerformAnnealing

```
PerformAnnealing(Simulation, Integrator, Barostat, TemperatureStart,
```

TemperatureEnd, TemperatureChange, SimulationSteps)

Perform annealing by heating or cooling the system from start to end temperature.

The temperature is increased or decreased from start to end temperature by temperature to perform annealing following by performing simulations for a specified number of steps after each temperature step.

Arguments:

System (object): OpenMM system object.
 Integrator (object): OpenMM integrator object.
 Barostat (object): OpenMM integrator object.
 TemperatureStart (float): Start temperature.
 TemperatureEnd (float): End temperature.
 TemperatureChange (int): Temperature step size to heat or cool the system from start to end temperature.
 SimulationSteps (fint): Number of simulations steps to perform after each temperature step.

Returns:

int: Total number of simulation steps.

Example(s):

```
... ..
TotalInitialSimulationSteps = OpenMMUtil.PerformAnnealing(Simulation,
    Integrator, Barostat, InitialStart, InitialEnd, InitialChange, InitialSteps)
```

ProcessOptionOpenMMAAnnealingParameters

ProcessOptionOpenMMAAnnealingParameters(ParamsOptionName, ParamsOptionValue, ParamsDefaultInfo=None)

Process parameters for annealing option and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs to setup platform.

The supported parameter names along with their default values for different platforms are shown below:

Initial heating parameters:

```
initialStart, 0.0 [ Units: kelvin ]
initialEnd, 300.0 [ Units: kelvin ]
initialChange, 5.0 [ Units: kelvin ]
initialSteps, 5000
```

```
initialEquilibrationSteps, 100000
```

Heating and cooling cycle parameters:

```
cycles, 1
```

```
cycleStart, auto [ Units: kelvin. The default value is set to
    initialEnd ]
cycleEnd, 315.0 [ Units: kelvin ]
cycleChange, 1.0 [ Units: kelvin ]
cycleSteps, 1000
```

```
cycleEquilibrationSteps, 100000
```

Final equilibration parameters:

```
finalEquilibrationSteps, 200000
```

A brief description of parameters is provided below:

Initial heating parameters:

```
initialStart: Start temperature for initial heating.
initialEnd: End temperature for initial heating.
```

`initialChange`: Temperature change for increasing temperature during initial heating.

`initialSteps`: Number of simulation steps after each heating step during initial heating

`initialEquilibrationSteps`: Number of equilibration steps after the completion of initial heating.

Heating and cooling cycles parameters:

`cycles`: Number of annealing cycles to perform. Each cycle consists of a heating and a cooling phase. The heating phase consists of the following steps: Heat system from start to end temperature using change size and perform simulation for a number of steps after each increase in temperature; Perform equilibration after the completion of heating. The cooling phase is reverse of the heating phase and cools the system from end to start temperature.

`cycleStart`: Start temperature for annealing cycle.

`cycleEnd`: End temperature for annealing cycle.

`cycleChange`: Temperature change for increasing or decreasing temperature during annealing cycle.

`cycleSteps`: Number of simulation steps after each heating and cooling step during annealing cycle.

`cycleEquilibrationSteps`: Number of equilibration steps after the completion of heating and cooling phase during a annealing cycle.

Final equilibration parameters:

`finalEquilibrationSteps`: Number of final equilibration steps after the completion of annealing cycles.

Arguments:

`ParamsOptionName` (str): Command line OpenMM annealing option name.

`ParamsOptionValues` (str): Space delimited list of parameter name and value pairs.

`ParamsDefaultInfo` (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ProcessOptionOpenMMAtomsSelectionParameters

`ProcessOptionOpenMMAtomsSelectionParameters(ParamsOptionName, ParamsOptionValue, ParamsDefaultInfo=None)`

Process parameters for selecting atoms and return a map containing processed parameter names and values.

`ParamsOptionValue` is a comma delimited list of parameter name and value pairs to select atoms.

The supported parameter names along with their default and possible values are shown below:

`selection`, none [Possible values: CAlphaProtein, Ions, Ligand, Protein, Residues, or Water]
`selectionSpec`, auto [Possible values: A space delimited list of residue names]
`negate`, no [Possible values: yes or no]

A brief description of parameters is provided below:

`selection`: Atom selection to freeze.

`selectionSpec`: A space delimited list of residue names for selecting atoms. You must specify its value during 'Ligand' and 'Protein' value for 'selection'. The default values are automatically set for 'CAAlphaProtein', 'Ions', 'Protein', and 'Water' values of 'selection' as shown below:

`CAAlphaProtein`: List of standard protein residues from pdbfixer for selecting CAlpha atoms.

`Ions`: Li Na K Rb Cs Cl Br F I

Water: HOH
 Protein: List of standard protein residues from pdbfixer.

negate: Negate atom selection match to select atoms for freezing.

In addition, you may specify an explicit space delimited list of residue names using 'selectionSpec' for any 'selection'. The specified residue names are appended to the appropriate default values during the selection of atoms for freezing.

Arguments:

ParamsOptionName (str): Command line OpenMM selection option name.
 ParamsOptionValues (str): Space delimited list of parameter name and value pairs.
 ParamsDefaultInfo (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ProcessOptionOpenMMForcefieldParameters

ProcessOptionOpenMMForcefieldParameters(ParamsOptionName, ParamsOptionValue, ParamsDefaultInfo=None)

Process parameters for biopolymer, small molecule, and water forcefields and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs for forcefields.

The supported parameter names along with their default and possible values are shown below:

biopolymer, amber14-all.xml [Possible values: Any Valid value]
 smallMolecule, OpenFF_2.2.0 [Possible values: Any Valid value]
 water, auto [Possible values: Any Valid value]

Possible biopolymer forcefield values:

amber14-all.xml, amber99sb.xml, amber99sbildn.xml, amber03.xml,
 amber10.xml
 charmm36.xml, charmm_polar_2019.xml
 amoeba2018.xml

Possible small molecule forcefield values:

openff_2.2.0, openff_2.0.0, openff_1.3.1, openff_1.2.1, openff_1.1.1,
 smirnoff99frosst
 gaff-2.11, gaff-2.1, gaff-1.81, gaff-1.8, gaff-1.4

The default water forcefield value is dependent on the type of the biopolymer forcefield as shown below:

Amber: amber14/tip3pfb.xml
 CHARMM: charmm36/water.xml or None for charmm_polar_2019.xml
 Amoebe: None (Explicit)

Possible water forcefield values:

amber14/tip3p.xml, amber14/tip3pfb.xml, amber14/spce.xml,
 amber14/tip4pew.xml, amber14/tip4pfb.xml,
 implicit/obc2.xml, implicit/GBn.xml, implicit/GBn2.xml
 charmm36/water.xml, charmm36/tip3p-pme-b.xml,
 charmm36/tip3p-pme-f.xml, charmm36/spce.xml,
 charmm36/tip4pew.xml, charmm36/tip4p2005.xml,
 charmm36/tip5p.xml, charmm36/tip5pew.xml,
 implicit/obc2.xml, implicit/GBn.xml, implicit/GBn2.xml
 amoeba2018_gk.xml (Implicit water), None (Explicit water for amoeba)

You may specify any valid forcefield name supported by OpenMM. No explicit validation is performed.

Arguments:

ParamsOptionName (str): Command line OpenMM forcefield option name.
 ParamsOptionValues (str): Space delimited list of parameter name and value pairs.
 ParamsDefaultInfo (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ProcessOptionOpenMMIntegratorParameters

```
ProcessOptionOpenMMIntegratorParameters(ParamsOptionName, ParamsOptionValue,
ParamsDefaultInfo=None, HydrogenMassRepartitioningStatus=False)
```

Process parameters for integrator option and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs to setup platform.

The supported parameter names along with their default and possible values are shown below:

```
integrator, LangevinMiddle [ Possible values: LangevinMiddle,
Langevin, NoseHoover, Brownian ]
```

```
randomSeed, auto [ Possible values: > 0 ]
```

```
frictionCoefficient, 1.0 [ Units: 1/ps ]
```

```
stepSize, auto [ Units: fs; Default value: 4 fs during yes value of
hydrogen mass repartitioning with no freezing/restraining of atoms;
otherwise, 2 fs ]
```

```
temperature, 300.0 [ Units: kelvin ]
```

```
barostat, MonteCarlo [ Possible values: MonteCarlo or
MonteCarloMembrane ]
```

```
barostatInterval, 25
```

```
pressure, 1.0 [ Units: atm ]
```

Parameters used only for MonteCarloMembraneBarostat with default values corresponding to Amber forcefields:

```
surfaceTension, 0.0 [ Units: atm*A. It is automatically converted
into OpenMM default units of atm*nm before its usage. ]
```

```
xymode, Isotropic [ Possible values: Anisotropic or Isotropic ]
```

```
zmode, Free [ Possible values: Free or Fixed ]
```

A brief description of parameters is provided below:

integrator: Type of integrator

randomSeed: Random number seed for barostat and integrator. Not supported NoseHoover integrator.

frictionCoefficient: Friction coefficient for coupling the system to the heat bath.

stepSize: Simulation time step size.

temperature: Simulation temperature.

barostat: Barostat type.

barostatInterval: Barostat interval step size, in terms of time step size, for applying Monte Carlo pressure changes during NPT simulation.

pressure: Pressure during NPT simulation.

surfaceTension: Surface tension acting on the system.

xymode: Behavior along X and Y axes. You may allow the X and Y axes to vary independently of each other or always scale them by the same amount to keep the ratio of their lengths constant.

zmode: Behavior along Z axis. You may allow the Z axis to vary independently of the other axes or keep it fixed.

Arguments:

ParamsOptionName (str): Command line OpenMM integrator option name.

ParamsOptionValues (str): Space delimited list of parameter name and value pairs.

ParamsDefaultInfo (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ProcessOptionOpenMMMDProtocolParameters

```
ProcessOptionOpenMMMDProtocolParameters(ParamsOptionName, ParamsOptionValue,
ParamsDefaultInfo=None)
```

Process parameters for MD protocol option and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs to setup platform.

The supported parameter names along with their default values for different platforms are shown below:

Phase1 - Initial heating parameters (NVT simulation):

```
phase1, yes [ Possible values: yes or no ]
phase1InitialStart, 0.0 [ Units: kelvin ]
phase1InitialEnd, 300.0 [ Units: kelvin ]
phase1InitialChange, 5.0 [ Units: kelvin ]
phase1InitialSteps, 5000
```

```
phase1InitialEquilibrationSteps, 100000
```

Phase2 - Heating and cooling cycles parameters (NVT simulation):

```
phase2, yes [ Possible values: yes or no ]
phase2Cycles, 1
phase2CycleStart, auto [ Units: kelvin. The default value is set to
    initialEnd ]
phase2CycleEnd, 315.0 [ Units: kelvin ]
phase2CycleChange, 1.0 [ Units: kelvin ]
phase2CycleSteps, 1000
```

```
phase2CycleEquilibrationSteps, 100000
```

Phase3 - NVT equilibration parameters:

```
phase3, yes [ Possible values: yes or no ]
phase3Steps, 200000
```

Phase4 - NPT equilibration parameters:

```
phase4, yes [ Possible values: yes or no ]
phase4Steps, 200000
```

Phase5 - NPT production parameters:

```
phase5, yes [ Possible values: yes or no ]
phase5Steps, 1000000
phase5StepSize, auto [ Units: fs; Default value: Same as stepSize
    parameter in integratorParams option. ]
```

A brief description of parameters is provided below:

Phase1 - Initial heating parameters (NVT simulation):

```
phase1: Execute phase1.
phase1InitialStart: Start temperature for initial heating.
phase1InitialEnd: End temperature for initial heating.
phase1InitialChange: Temperature change for increasing temperature
    during initial heating.
phase1InitialSteps: Number of simulation steps after each
    heating step during initial heating
```

```
phase1InitialEquilibrationSteps: Number of equilibration steps
    after the completion of initial heating.
```

Phase2 - Heating and cooling cycles parameters (NVT simulation):

```
phase2: Execute phase2.
phase2Cycles: Number of annealing cycles to perform. Each cycle
    consists of a heating and a cooling phase. The heating phase
    consists of the following steps: Heat system from start to
    end temperature using change size and perform simulation for a
    number of steps after each increase in temperature; Perform
    equilibration after the completion of heating. The cooling
```

phase is reverse of the heating phase and cools the system from end to start temperature.

phase2CycleStart: Start temperature for annealing cycle.
 phase2CycleEnd: End temperature for annealing cycle.
 phase2CycleChange: Temperature change for increasing or decreasing temperature during annealing cycle.
 phase2CycleSteps: Number of simulation steps after each heating and cooling step during annealing cycle.

phase2CycleEquilibrationSteps: Number of equilibration steps after the completion of heating and cooling phase during a annealing cycle.

Phase3 - NVT equilibration parameters:

phase3: Execute phase3.
 phase3Steps: Number of NVT equilibration steps.

Phase4 - NPT equilibration parameters:

phase4: Execute phase4.
 phase4Steps: Number of NPT equilibration steps.

Phase5 - NPT production parameters:

phase5: Execute phase5
 phase5Steps: Number of NPT production steps.
 phase5StepSize: Simulation time step size for NPT production.

Arguments:

ParamsOptionName (str): Command line OpenMM MD protocol option name.
 ParamsOptionValues (str): Space delimited list of parameter name and value pairs.
 ParamsDefaultInfo (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ProcessOptionOpenMMOutputParameters

ProcessOptionOpenMMOutputParameters(ParamsOptionName, ParamsOptionValue, OutfilePrefix, ParamsDefaultInfo=None)

Process parameters for output option and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs to setup platform.

The supported parameter names along with their default and possible values are shown below:

checkpoint, no [Possible values: yes or no]
 checkpointFile, auto [Default: <OutfilePrefix>.chk]
 checkpointSteps, 10000

 dataOutType, auto [Possible values: A space delimited list of valid parameter names.
 NPT simulation default: Density Step Speed Progress
 PotentialEnergy Temperature Time.
 NVT simulation default: Step Speed Progress PotentialEnergy
 Temperature Time Volume
 Other valid names: ElapsedTime RemainingTime KineticEnergy
 TotalEnergy]

 dataLog, yes [Possible values: yes or no]
 dataLogFile, auto [Default: <OutfilePrefix>.csv]
 dataLogSteps, 1000

 dataStdout, no [Possible values: yes or no]
 dataStdoutSteps, 1000

```

dataOutTypePlot, yes [ Possible values: yes or no ]
dataOutTypePlotX, auto [ Default: Time; Possible values: Step or
Time ]
dataOutTypePlotY, auto [ Possible values: A space delimited list
of valid parameter names specified for dataOutType.
NPT simulation default: Density PotentialEnergy Temperature
NVT simulation default: PotentialEnergy Temperature Volume
Other valid names: KineticEnergy TotalEnergy]

minimizationDataSteps, 100
minimizationDataStdout, no [ Possible values: yes or no ]
minimizationDataLog, no [ Possible values: yes or no ]
minimizationDataLogFile, auto [ Default:
<OutfilePrefix>_MinimizationOut.csv ]
minimizationDataOutType, auto [ Possible values: A space delimited
list of valid parameter names. Default: SystemEnergy
RestraintEnergy MaxConstraintError.
Other valid names: RestraintStrength ]

pdbOutFormat, PDB [ Possible values: PDB or CIF ]
pdbOutKeepIDs, yes [ Possible values: yes or no ]

pdbOutMinimized, no [ Possible values: yes or no ]
pdbOutEquilibrated, no [ Possible values: yes or no ]
pdbOutFinal, no [ Possible values: yes or no ]

pdbOutPhase1HeatedNVT, no [ Possible values: yes or no ]
pdbOutPhase2AnnealedNVT, no [ Possible values: yes or no ]
pdbOutPhase3EquilibratedNVT, no [ Possible values: yes or no ]
pdbOutPhase4EquilibratedNPT, no [ Possible values: yes or no ]
pdbOutPhase5ProductionNPT, no [ Possible values: yes or no ]

saveFinalStateCheckpoint, yes [ Possible values: yes or no ]
saveFinalStateCheckpointFile, auto [ Default:
<OutfilePrefix>_FinalState.chk ]
saveFinalStateXML, no [ Possible values: yes or no ]
saveFinalStateXMLFile, auto [ Default:
<OutfilePrefix>_FinalState.xml ]

traj, yes [ Possible values: yes or no ]
trajFile, auto [ Default: <OutfilePrefix>.<TrajFormat> ]
trajFormat, DCD [ Possible values: DCD or XTC ]
trajSteps, 10000

xmlSystemOut, no [ Possible values: yes or no ]
xmlSystemFile, auto [ Default: <OutfilePrefix>_System.xml ]
xmlIntegratorOut, no [ Possible values: yes or no ]
xmlIntegratorFile, auto [ Default: <OutfilePrefix>_Integrator.xml ]

```

A brief description of parameters is provided below:

```

checkpoint: Write intermediate checkpoint file.
checkpointFile: Intermediate checkpoint file name.
checkpointSteps: Frequency of writing intermediate checkpoint file.

dataOutType: Type of data to write to stdout and log file.

dataLog: Write data to log file.
dataLogFile: Data log file name.
dataLogSteps: Frequency of writing data to log file.

dataStdout: Write data to stdout.
dataStdoutSteps: Frequency of writing data to stdout.

dataOutTypePlot: Generate plots using data written to log file.
dataOutTypePlotX: Data out type to plot on X axis.
dataOutTypePlotY: Data out types to plot on Y axis. An individual plot
is generated for each pair of X and Y vaues to be plotted.

```

minimizationDataSteps: Frequency of writing data to stdout and log file.
 minimizationDataStdout: Write data to stdout.
 minimizationDataLog: Write data to log file.
 minimizationDataLogFile: Data log file name.
 minimizationDataOutType: Type of data to write to stdout and log file.

pdbOutFormat: Format of output PDB files.
 pdbOutKeepIDs: Keep existing chain and residue IDs.

pdbOutMinimized: Write PDB file after minimization.
 pdbOutEquilibrated: Write PDB file after equilibration.
 pdbOutFinal: Write final PDB file after production run.

pdbOutPhase1HeatedNVT: Write out PDB file after initial heating.
 pdbOutPhase2AnnealedNVT: Write out PDB file after heating and cooling.
 pdbOutPhase3EquilibratedNVT: Write out PDB file after equilibration.
 pdbOutPhase4EquilibratedNPT: Write out PDB file after equilibration.
 pdbOutPhase5ProductionNPT: Write out PDB file after production run.

saveFinalStateCheckpoint: Save final state checkpoint file.
 saveFinalStateCheckpointFile: Name of final state checkpoint file.
 saveFinalStateXML: Save final state XML file.
 saveFinalStateXMLFile: Name of final state XML file.

traj: Write out trajectory file.
 trajFile: Trajectory file name.
 trajFormat: Trajectory file format.
 trajSteps: Frequency of writing trajectory file.

xmlSystemOut: Write system XML file.
 xmlSystemFile: System XML file name.
 xmlIntegratorOut: Write integrator XML file.
 xmlIntegratorFile: Integrator XML file name.

Arguments:

ParamsOptionName (str): Command line OpenMM system option name.
 ParamsOptionValues (str): Space delimited list of parameter name and value pairs.
 ParamsDefaultInfo (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ProcessOptionOpenMMPlatformParameters

ProcessOptionOpenMMPlatformParameters(ParamsOptionName, ParamsOptionValue,
 ParamsDefaultInfo=None)

Process parameters for platform option and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs to setup platform.

The supported parameter names along with their default values for different platforms are shown below:

CPU:

threads, 1 [Possible value: ≥ 0 or auto. The value of 'auto' or zero implies the use of all available CPUs for threading.]

CUDA:

deviceIndex, auto [Possible values: 0, '0 1' etc.]
 deterministicForces, auto [Possible values: yes or no]
 precision, single [Possible values: single, double, or mix]
 tempDirectory, auto [Possible value: DirName]
 useBlockingSync, auto [Possible values: yes or no]
 useCpuPme, auto [Possible values: yes or no]

OpenCL:

```

deviceIndex, auto [ Possible values: 0, '0 1' etc. ]
openCLPlatformIndex, auto [ Possible value: Number]
precision, single [ Possible values: single, double, or mix ]
useCpuPme, auto [ Possible values: yes or no ]

```

A brief description of parameters is provided below:

CPU:

threads: Number of threads to use for simulation.

CUDA:

deviceIndex: Space delimited list of device indices to use for calculations.

deterministicForces: Generate reproducible results at the cost of a small decrease in performance.

precision: Number precision to use for calculations.

tempDirectory: Directory name for storing temporary files.

useBlockingSync: Control run-time synchronization between CPU and GPU.

useCpuPme: Use CPU-based PME implementation.

OpenCL:

deviceIndex: Space delimited list of device indices to use for simulation.

openCLPlatformIndex: Platform index to use for calculations.

precision: Number precision to use for calculations.

useCpuPme: Use CPU-based PME implementation.

Arguments:

ParamsOptionName (str): Command line OpenMM platform option name.

ParamsOptionValues (str): Space delimited list of parameter name and value pairs.

ParamsDefaultInfo (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ProcessOptionOpenMMRestartParameters

```

ProcessOptionOpenMMRestartParameters(ParamsOptionName, ParamsOptionValue,
OutfilePrefix, ParamsDefaultInfo=None)

```

Process parameters for restart option and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs to setup platform.

The supported parameter names along with their default and possible values are shown below:

```

finalStateFile, <OutfilePrefix>_FinalState.<chk> [ Possible values:
Valid final state checkpoint or XML filename ]
dataAppend, yes [ Possible values: yes or no]

```

A brief description of parameters is provided below:

finalStateFile: Final state checkpoint or XML file

dataAppend: Append data to existing trajectory and data log files during the restart of a simulation using a previously saved final state checkpoint or XML file.

Arguments:

ParamsOptionName (str): Command line OpenMM restart option name.

ParamsOptionValues (str): Space delimited list of parameter name and value pairs.

OutfilePrefix (str): Prefix for output files.

ParamsDefaultInfo (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ProcessOptionOpenMMSimulationParameters

```
ProcessOptionOpenMMSimulationParameters(ParamsOptionName, ParamsOptionValue,
ParamsDefaultInfo=None)
```

Process parameters for simulation option and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs to setup platform.

The supported parameter names along with their default and possible values are shown below:

```
steps, 1000000 [ Possible values: > 0 ]

minimization, yes [ Possible values: yes or no ]
minimizationMaxSteps, auto [ Possible values: >= 0. The value of
zero implies until the minimization is converged. ]
minimizationTolerance, 0.24 [ Units: kcal/mol/A. The default value
0.24, corresponds to OpenMM default of value of 10.04
kJoules/mol/nm. It is automatically converted into OpenMM
default units before its usage. ]

equilibration, yes [ Possible values: yes or no ]
equilibrationSteps, 1000 [ Possible values: > 0 ]
```

A brief description of parameters is provided below:

steps: Number of steps for production run.

equilibration: Perform equilibration before the production run.

equilibrationSteps: Number of steps for equilibration.

minimizationMaxSteps: Maximum number of minimization steps. The value of zero implies until the minimization is converged.

minimizationTolerance: Energy convergence tolerance during minimization.

minimization: Perform minimization before equilibration and production run.

Arguments:

```
ParamsOptionName (str): Command line OpenMM simulation option name.
ParamsOptionValues (str): Space delimited list of parameter name and value pairs.
ParamsDefaultInfo (dict): Default values to override for selected parameters.
```

Returns:

```
dictionary: Processed parameter name and value pairs.
```

ProcessOptionOpenMMSystemParameters

```
ProcessOptionOpenMMSystemParameters(ParamsOptionName, ParamsOptionValue,
ParamsDefaultInfo=None)
```

Process parameters for system option and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs to setup platform.

The supported parameter names along with their default and possible values are shown below:

```
constraints, BondsInvolvingHydrogens [ Possible values: None,
WaterOnly, BondsInvolvingHydrogens, AllBonds, or
AnglesInvolvingHydrogens ]
constraintErrorTolerance, 0.000001
ewaldErrorTolerance, 0.0005

nonbondedMethodPeriodic, PME [ Possible values: NoCutoff,
CutoffNonPeriodic, or PME ]
nonbondedMethodNonPeriodic, NoCutoff [ Possible values:
NoCutoff or CutoffNonPeriodic]
nonbondedCutoff, 1.0 [ Units: nm ]

hydrogenMassRepartitioning, yes [ Possible values: yes or no ]
hydrogenMass, 1.5 [ Units: amu]

removeCMMotion, yes [ Possible values: yes or no ]
```

```
rigidWater, auto [ Possible values: yes or no. Default: 'No' for
                  'None' value of constraints; Otherwise, yes ]
```

A brief description of parameters is provided below:

constraints: Type of system constraints to use for simulation. These constraints are different from freezing and restraining of any atoms in the system.

constraintErrorTolerance: Distance tolerance for constraints as a fraction of the constrained distance.

ewaldErrorTolerance: Ewald error tolerance for a periodic system.

nonbondedMethodPeriodic: Nonbonded method to use during the calculation of long range interactions for a periodic system.

nonbondedMethodNonPeriodic: Nonbonded method to use during the calculation of long range interactions for a non-periodic system.

nonbondedCutoff: Cutoff distance to use for long range interactions in both periodic non-periodic systems.

hydrogenMassRepartitioning: Use hydrogen mass repartitioning. It increases the mass of the hydrogen atoms attached to the heavy atoms and decreasing the mass of the bonded heavy atom to maintain constant system mass. This allows the use of larger integration step size (4 fs) during a simulation.

hydrogenMass: Hydrogen mass to use during repartitioning.

removeCMMotion: Remove all center of mass motion at every time step.

rigidWater: Keep water rigid during a simulation. This is determined automatically based on the value of 'constraints' parameter.

Arguments:

ParamsOptionName (str): Command line OpenMM system option name.

ParamsOptionValues (str): Space delimited list of parameter name and value pairs.

ParamsDefaultInfo (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ProcessOptionOpenMMWaterBoxParameters

```
ProcessOptionOpenMMWaterBoxParameters(ParamsOptionName, ParamsOptionValue,
ParamsDefaultInfo=None)
```

Process parameters for adding a water box option and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs for adding a water box.

The supported parameter names along with their default and possible values are shown below:

```
model, tip3p [ Possible values: tip3p, spce, tip4pew, tip5p or swm4ndp ]
mode, Padding [ Possible values: Size or Padding ]
size, None [ Possible values: xsize ysize zsize ]
padding, 1.0
shape, cube [ Possible values: cube, dodecahedron, or octahedron ]
ionPositive, Na+ [ Possible values: Li+, Na+, K+, Rb+, or Cs+ ]
ionNegative, Cl- [ Possible values: Cl-, Br-, F-, or I- ]
ionicStrength, 0.0
```

A brief description of parameters is provided below:

model: Water model to use for adding water box.

mode: Specify the size of the waterbox explicitly or calculate it automatically for a macromolecule along with adding padding around macromolecule. Possible values: Size or Padding.

size: A space delimited triplet of values corresponding to water size in nanometers. It must be specified during 'Size' value of 'mode' parameter.

padding: Padding around macromolecule in nanometers for filling box with water. It must be specified during 'Padding' value of 'mode' parameter.

ionPositive: Type of positive ion to add during the addition of a water box.

ionNegative: Type of negative ion to add during the addition of a water box.

ionicStrength: Total concentration (molar) of both positive and negative ions to add excluding the ions added to neutralize the system during the addition of a water box.

Arguments:

ParamsOptionName (str): Command line OpenMM water box option name.
 ParamsOptionValues (str): Space delimited list of parameter name and value pairs.
 ParamsDefaultInfo (dict): Default values to override for selected parameters.

Returns:

dictionary: Processed parameter name and value pairs.

ReadPDBFile

ReadPDBFile(PDBFile)

Read molecule from a PDB file.
 The supported PDB file formats are pdb and cif.

Arguments:

PDBFile (str): Name of PDB file.

Returns:

object: OpenMM PDBFile or PDBFilex object.

ReadSmallMoleculeFile

ReadSmallMoleculeFile(FileName)

Read small molecule file using OpenFF toolkit.

Arguments:

FileName (str): Small molecule file name.

Returns:

None or OpenFF toolkit molecule object.

ReimageRealignTrajectory

ReimageRealignTrajectory(Topology, TrajFile, ReimageFrames=True,
 RealignFrames=True, Selection="protein and backbone and name CA")

Reimage and realign a trajectory file using MDTraj. The trajectory frames are reimaged before realigning to the first frame using the specified atom selection.

The trajectory file format must a valid format supported by MDTraj. No validation is performed.

Arguments:

Topology (str or object): PDB file name or OpenMM topology object.
 TrajFile (str): Trajectory file name.
 ReimageFrames (bool): Reimage trajectory frames.
 RealignFrames (bool): Realign trajectory frames.
 Selection (str): MDTraj atom selection for realigning frames.

Returns:

None or object: MDTraj trajectory object.
 bool: Reimaged status.
 bool: Realigned status.

RestraintAtoms

RestraintAtoms(System, Positions, AtomList, SpringConstantInKcal)

Restraint atoms during a simulation. The motion of specified atoms is restricted by adding a harmonic force that binds them to their starting positions. The atoms are not completely fixed unlike freezing of atoms. Their motion, however, is restricted and they are not able to move far away from their starting positions during local energy minimization and MD simulation.

The SpringConstantInKcal value must be specified in the units of kcal/mol/Å². It is automatically converted into the units of kJoules/mol/nm² for OpenMM API call.

Arguments:

System (object): OpenMM system object.
 Positions (object): OpenMM positions object.
 AtomList (list): List of OpenMM atom object.
 SpringConstantInKcal (float): Spring constant value.

Returns:

None

SetupAnnealingParameters

SetupAnnealingParameters(ParamsInfo)

Setup annealing parameters for OpenMM API calls.

The ParamsInfo parameter is a dictionary of name and value pairs for annealing parameters and may be generated by calling the function named ProcessOptionOpenMMAnnealingParameters().

Arguments:

ParamsInfo (dict): Parameter name and value pairs.

Returns:

dict: Annealing parameter name and values pairs.

Example(s):

```
OptionsInfo["AnnealingParams"] =
    OpenMMUtil.ProcessOptionOpenMMAnnealingParameters(
        "--annealingParams", Options["--annealingParams"],
        ... ..
    )
AnnealingParams = SetupAnnealingParams(
    OptionsInfo["AnnealingParams"])
```

SetupIntegratorParameters

SetupIntegratorParameters(ParamsInfo)

Setup integrator parameters for OpenMM API calls.

The ParamsInfo parameter is a dictionary of name and value pairs for integrator parameters and may be generated by calling the function named ProcessOptionOpenMMIntegratorParameters().

Arguments:

ParamsInfo (dict): Parameter name and value pairs.

Returns:

dict: Integrator parameter name and values pairs.

Example(s):

```
OptionsInfo["IntegratorParams"] =
    OpenMMUtil.ProcessOptionOpenMMIntegratorParameters(
        "--integratorParams", Options["--integratorParams"],
        HydrogenMassRepartitioningStatus =
        OptionsInfo["SystemParams"]["HydrogenMassRepartitioning"])
    ... ..
IntegratorParams = SetupIntegratorParameters(
    OptionsInfo["IntegratorParams"])
```

SetupMDProtocolParameters

SetupMDProtocolParameters(ParamsInfo)

Setup MD protocol parameters for OpenMM API calls.

The ParamsInfo parameter is a dictionary of name and value pairs for annealing parameters and may be generated by calling the function named ProcessOptionOpenMMMDProtocolParameters().

Arguments:

ParamsInfo (dict): Parameter name and value pairs.

Returns:

dict: MD protocol parameter name and values pairs.

Example(s):

```
OptionsInfo["ProtocolParams"] =
    OpenMMUtil.ProcessOptionOpenMMMDProtocolParameters(
        "--protocolParams", Options["--protocolParams"],
        ... ..
    ProtocolParams = SetupMDProtocolParameters(
        OptionsInfo["ProtocolParams"])
```

SetupPlatformParameters

```
SetupPlatformParameters(ParamsInfo)
```

Setup platform parameters for OpenMM calls.

The ParamsInfo parameter is a dictionary of name and value pairs for platform parameters and may be generated by calling the function named ProcessOptionOpenMMPlatformParameters().

Arguments:

ParamsInfo (dict): Parameter name and value pairs.

Returns:

```
str: PlatformName.
dict: Platform properties parameter name and values pairs.
str: Text message describing platform.
```

Example(s):

```
ParamsDefaultInfoOverride = {"Name": Options["--platform"],
    "Threads": 1}
OptionsInfo["PlatformParams"] =
    OpenMMUtil.ProcessOptionOpenMMPlatformParameters("--platformParams",
        Options["--platformParams"], ParamsDefaultInfoOverride)
... ..
PlatformName, PlatformProperties, PlatformMsg =
    SetupPlatformParameters(PlatformParamsInfo)
```

SetupSimulationParameters

```
SetupSimulationParameters(ParamsInfo)
```

Setup simulation parameters for OpenMM API calls.

The ParamsInfo parameter is a dictionary of name and value pairs for integrator parameters and may be generated by calling the function named ProcessOptionOpenMMSimulationParameters().

Arguments:

ParamsInfo (dict): Parameter name and value pairs.

Returns:

```
dict: Integrator parameter name and values pairs.
```

Example(s):

```
OptionsInfo["SimulationParams"] =
    OpenMMUtil.ProcessOptionOpenMMSimulationParameters(
        "--simulationParams", Options["--simulationParams"])
... ..
SimulationParams = SetupSimulationParameters(
    OptionsInfo["SimulationParams"])
```

SetupSystemGeneratorForcefieldsParameters

```
SetupSystemGeneratorForcefieldsParameters(SystemParamsInfo)
```

Setup forcefield parameters for OpenMM API calls.

The SystemParamsInfo parameter is a dictionary of name and value pairs for system parameters and may be generated by calling the function named ProcessOptionOpenMMSystemParameters().

Arguments:

SystemParamsInfo (dict): Parameter name and value pairs.

Returns:

dict: Forcefield parameter name and value pairs.
 dictionary2: Periodic forcefield parameter name and value pairs.
 dictionary3: Non-periodic parameter name and value pairs.

Example(s):

```
OptionsInfo["SystemParams"] =
    OpenMMUtil.ProcessOptionOpenMMSystemParameters("--systemParams",
    Options["--systemParams"])
... ..
(ForcefieldParams, PeriodicForcefieldParams, NonPeriodicForcefieldParams)
    = SetupSystemGeneratorForcefieldsParameters(OptionsInfo["SystemParams"])
```

ValidateAndFreezeRestraintAtoms

```
ValidateAndFreezeRestraintAtoms(FreezeAtomsStatus, FreezeAtomsParamsInfo,
RestraintAtomsStatus, RestraintAtomsParamsInfo, RestraintSpringConstantInKcal,
SystemParamsInfo, System, Topology, Positions)
```

Handle freezing and restraining of atoms along with validation of atoms to be frozen/restrained.

The FreezeAtomsParamsInfo and RestraintAtomsParamsInfo parameters are dictionaries of name and value pairs for system parameters and may be generated by calling the function named ProcessOptionOpenMMAAtomsSelectionParameters().

The SystemParamsInfo parameter is a dictionary of name and value pairs for system parameters and may be generated by calling the function named ProcessOptionOpenMMSystemParameters().

Arguments:

FreezeAtomsStatus (bool): Freeze atoms.
 FreezeAtomsParamsInfo (dict): Parameter name and value pairs.
 RestraintAtomsStatus (bool): Restraint atoms.
 RestraintAtomsParamsInfo (dict): Parameter name and value pairs.
 RestraintSpringConstantInKcal (float): Restraint spring constant.
 SystemParamsInfo (dict): Parameter name and value pairs.
 Topology (object): OpenMM topology object.
 Positions (object): OpenMM positions object.

Returns:

None or List: OpenMM atom objects.
 None or List: OpenMM atom objects.

WritePDBFile

```
WritePDBFile(PDBFile, Topology, Positions, KeepIDs=True)
```

Write a PDB file.

The supported PDB file formats are pdb and cif.

Arguments:

PDBFile (str): Name of PDB file.
 Topology (object): Topology OpenMM object.
 Positions (object): Positions OpenMM object.
 KeepIDs (bool): Keep existing residue and chain IDs.

Returns:

None

WriteSimulationStatePDBFile

```
WriteSimulationStatePDBFile(Simulation, PDBFile, KeepIDs=True)
```

Write a PDB file for current simulation state.

The supported PDB file formats are pdb and cif.

Arguments:

Simulation (object): OpenMM simulation object.

PDBFile (str): Name of PDB fil.
KeepIDs (bool): Keep existing residue and chain IDs.

Returns:

None

AUTHOR

Manish Sud <msud@san.rr.com>

COPYRIGHT

Copyright (C) 2026 Manish Sud. All rights reserved.

The functionality available in this script is implemented using OpenMM, an open source molecular simulation package.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.