1 MMPBSA.py

1.1 Introduction

This section describes the use of the python script MMPBSA.py to perform Molecular Mechanics / Poisson Boltzmann (or Generalized Born) Surface Area (MM/PB(GB)SA) calculations. This is a post-processing method in which representative snapshots from an ensemble of conformations are used to calculate the free energy change between two states (typically a bound and free state of a receptor and ligand). Free energy differences are calculated by combining the so-called gas phase energy contributions that are independent of the chosen solvent model as well as solvation free energy components (both polar and non-polar) calculated from an implicit solvent model for each species. Entropy contributions to the total free energy may be added as a further refinement. The entropy calculations are currently done only in the gas phase with the *nmode* program in Amber or via the quasi-harmonic approximation in *ptraj*.

The gas phase free energy contributions are calculated by *sander* within the Amber program suite according to the force field with which the topology files were created. The solvation free energy contributions may be further decomposed into an electrostatic and hydrophobic contribution. The electrostatic portion is calculated using either the linearized Poisson Boltzmann (PB) equation or by the Generalized Born method. The PB equation is solved numerically by either the *pbsa* program included with AmberTools or by the Adaptive Poisson Boltzmann Solver (APBS) program through the iAPBS interface with Amber (for more information, see http://www.poissonboltzmann.org/apbs). The hydrophobic contribution is approximated by the LCPO method [2] implemented within *sander*.

MM/PB(GB)SA typically employs the approximation that the configurational space explored by the systems are very similar between the bound and unbound states, so every snapshot for each species is extracted from the same trajectory file, although MMPBSA.py will accept separate trajectory files for each species. Furthermore, explicit solvent and ions are stripped from the trajectory file(s) to hasten convergence by preventing solvent-solvent interactions from dominating the energy terms. A more detailed explanation of the theory can be found in Srinivasan, et. al.[1]

1.2 Installation and Testing

You must have amber9 or later to use MMPBSA.py, as it takes advantage of trajectory analysis (imin=5) that is not present in earlier versions. Morever, make sure that you have applied all bugfixes to date. The source code for MMPBSA.py is included in $AMBERHOME/src/mmpbsa_py. MMPBSA.py is installed by default in a typical Amber11 installation. If you are updating the code or re-installing for any reason, you can install MMPBSA.py by itself. You can install MMPBSA.py using the command

```
cd $AMBERHOME/src/mmpbsa_py
make install
```

or MMPBSA.py.MPI using the command

```
cd $AMBERHOME/src/mmpbsa_py
make parallel
```

Any time you download an update to the script, you must unpack the updated Python files into $AMBERHOME/src/mmpbsa_py and re-install per the instructions above. The Makefile in this directory checks for the version of Python that you have, converts the files if necessary, compiles the modules, and puts the compiled modules and MMPBSA.py into $AMBERHOME/bin. Python 2.5 or greater is necessary for proper installation of MMPBSA.py. Python version 2 is significantly different than Python version 3, and they are not compatible. If you have version 2, then no file conversion is necessary (as the script was written for version 2). However, if you have Python version 3, the utility 2to3 must be present to translate the code into Python 3 syntax (this will be done automatically if 2to3 is in your path when you run "make install").

The parallel version of the scripts, MMPBSA.py.MPI, utilizes the python module mpi4py. If mpi4py is not currently installed on your platform, then you must compile it before using MMPBSA.py.MPI. Download and installation instructions for mpi4py can be found online at http://ambermd.org/tutorials/advanced/tutorial3/py_script/compile.htm.

After you successfully install MMPBSA.py, check that everything is working in the test directory. Use the commands

```
cd $AMBERHOME/test
make test.mmpbsa_py
```

to start the test. The test will take several minutes if it works correctly, so please be patient while it completes. If DO_PARALLEL is set then this command will automatically test MMPBSA.py.MPI, as well. If you see a potential failure, check the TEST_FAILURES.diff file. Minor differences are benign.

1.3 Preparing for an MM/PB(GB)SA calculation

MM/PB(GB)SA is often a very useful tool for obtaining relative free energies of binding when comparing ligands. Perhaps its biggest advantage is that it is very computationally inexpensive compared to other free energy calculations, such as TI or FEP. Following the advice given below before any MD simulations are run will make running MMPBSA.py successfully much easier.

1.3.1 Building Topology Files

MMPBSA.py requires at least three, usually four, compatible topology files. If you plan on running MD in explicit water, you will need a solvated topology file of the entire complex, and you will always need a topology for the entire complex, one for just the receptor, and a final one for just the ligand. Moreover, they must be compatible with one another (i.e. each must

have the same charges for the same atoms, the same force field must be used for all three of the required prmtops, and they must have the same PBRadii set, see LEaP for description of pbradii). Thus, it is strongly advised that all prmtop files are created with the same script. We run through a typical example here, though leave some of the details to other sections and other tutorials. We will start with a system that is a large protein binding a small, one-residue ligand. We will assume that a docked structure has already been obtained as a PDB and that two separate PDBs have been constructed, receptor.pdb and LIG.pdb. We will also assume that a MOL2 file was created from LIG.pdb, residue name 'LIG', was built with charges already derived (either through antechamber or some other method), and an frcmod file for 'LIG' that contains all missing parameters have already been created. Furthermore, we will use the FF99SB force field for this example. A sample script file called, for instance, mmpbsa_leap.in, is shown below

```
source leaprc.ff99SB
loadAmberParams LIG.frcmod
LIG = loadMol2 LIG.mol2
receptor = loadPDB receptor.pdb
complex = combine {receptor LIG}
set default PBRadii mbondi2

saveAmberParm LIG lig.top lig.crd
saveAmberParm receptor rec.top rec.crd
saveAmberParm complex com.top com.crd

solvateOct complex TIP3PBOX 15.0
saveAmberParm complex com_solvated.top com_solvated.crd
quit
```

The above script, run with the command

```
sleap -f mmpbsa_leap.in
```

should produce four prmtop files, lig.top, rec.top, com.top, and com_solvated.top. Topology files created in this manner will make running MMPBSA.py far easier. This is, of course, the simplest case, but we briefly describe some other examples. MMPBSA.py will guess the mask for both the receptor and ligand inside the complex topology file as long as the ligand residues appear continuously in the complex topology file. Thus, for instance, if you're adding two ligands, combine the two ligands consecutively in the complex (rather than one residue at the beginning and one at the end, for instance).

1.3.2 Running Molecular Dynamics

Not many details will be given here, as MM/PB(GB)SA is a post-processing trajectory analysis technique. Molecular dynamics are run to generate an ensemble of snapshots upon which to calculate the binding energy. This technique is most effective when the structures are not correlated, which means that the simulated time between extracted snapshots should be sufficiently large to avoid such correlation.

There are two techniques that can be employed when running these simulations with respect to MMPBSA.py. The first is what's called the "single trajectory protocol" and the second of which is called the "multiple trajectory protocol". The first method will extract the snapshots for the complex, receptor, and ligand from the same trajectory. This is a faster method because it requires the simulation of only a single system, but makes the assumption that the configurational space explored by the receptor and ligand is unchanged between the bound and unbound states. The latter method eliminates this assumption at the cost of more simulations. MMPBSA.py requires a complex trajectory, but will accept a receptor and/or ligand trajectory as well. Any trajectory not given to the script will be extracted from the complex trajectory.

1.4 Running MMPBSA.py

1.4.1 The input file

The input file was designed to be as syntactically similar to other programs in Amber as possible. The input file has the same namelist structure as both sander and pmemd. The allowed namelists are &general, &gb, &pb, &alanine_scanning, and &nmode. The input variables recognized in each namelist are described below, but those in &general are typically variables that apply to all aspects of the calculation. Those in &gb are unique to Generalized Born calculations, &pb is unique to Poisson Boltzmann simulations, &alanine_scanning is unique to alanine scanning calculations, and &nmode is unique to the normal mode calculations used to approximate vibrational entropies. All of the input variables are described below according to their respective namelists. Integers and floating point variables should be typed as-is while strings should be put in either single- or double-quotes.

&general namelist variables

startframe The frame from which to begin extracting snapshots from each trajectory. This is always the first frame read. (Default = 1)

endframe The frame from which to stop extracting snapshots from each trajectory. Any number higher than the total number of frames is automatically reduced to the last frame in each trajectory. (Default = 1000000000)

interval The offset from which to choose frames from each trajectory file. For example, an interval of 2 will pull every 2nd frame beginning at startframe and ending less than or equal to endframe. (Default = 1)

receptor_mask The mask that specifies the receptor residues within the complex prmtop (NOT the solvated prmtop if there is one). The default guess is generally sufficient and will only fail if the ligand residues are not found in succession within the complex prmtop. You should not specify a receptor_mask unless you know your ligand residues are non-continuous in your prmtop. If they are continuous, but the program still complains that it cannot find a suitable mask, your topology files are likely inconsistent. It uses the "Amber mask" syntax described elsewhere in the Amber manuals. This will be replaced with the default receptor_mask if ligand_mask (below) is not also set.

ligand_mask The mask that specifies the ligand residues within the complex prmtop (NOT the solvated prmtop if there is one). The default guess is generally sufficient and will only fail when stated above. This follows the same description as the receptor_mask above.

verbose The variable that specifies how much output is printed in the output file. There are three allowed values: 0, 1, and 2. A value of 0 will simply print difference terms, 1 will print all complex, receptor, and ligand terms, and 2 will also print bonded terms if one trajectory is used. (Default = 1)

keep_files The variable that specifies which temporary files are kept. All temporary files have the prefix "_MMPBSA_" prepended to them. Allowed values are 0, 1, and 2. 0: Keep no temporary files, 1: Keep all generated trajectory files and output files created by sander, nmode, and ptraj, 2: Keep all temporary files. Temporary files are only deleted if MMPBSA.py completes successfully. (Default = 1)

strip_mdcrd The variable used to tell the MMPBSA.py whether or not to strip the mask specified by "strip_mask" from the trajectory. Allowed values are
0: Do not strip anything from the given mdcrds
1: Strip "strip_mask" from the given mdcrds (Default)

strip_mask The variable that specifies which atoms are stripped from the trajectory file if strip_mdcrd above is 1. (Default = ":WAT:Cl-:CIO:Cs+:IB:K+:Li+:MG2:Na+:Rb+") (see Advanced Options before changing)

entropy Specifies whether or not a quasi-harmonic entropy approximation is made with ptraj. Allowed values are 0: Don't. 1: Do (Default = 0)

&gb namelist variables (More thorough descriptions of each can be found in the Amber manual)

igb Generalized Born method to use. See the description in the Amber manual. Allowed values are 1, 2, 5, and 7. (Default = 5)

gbsa Option to carry out Generalized Born/Surface Area simulations. See the description in the Amber manual. Allowed values are 0, 1, and 2. (Default = 1)

saltcon Salt concentration in Molarity. (Default = 0.0)

surften Surface tension value (Default = 0.0072)

surfoff Offset to correct (by addition) the value of the non-polar contribution to the solvation free energy term (Default = 0.0)

&pb namelist variables (More thorough descriptions of each can be found in the Amber manual)

indi Internal dielectric constant (Default = 1.0)

exdi External dielectric constant (Default = 80.0)

scale Resolution of the Poisson Boltzmann grid. It is equal to the reciprocal of the grid spacing. (Default = 2.0)

linit Maximum number of iterations of the linear Poisson Boltzmann equation to try (Default = 1000)

prbrad Solvent probe radius in Angstroms. Allowed values are 1.4 and 1.6 (Default = 1.4)

istrng Ionic strength in Molarity. It is converted to mM for PBSA and kept as M for APBS. (Default = 0.0)

inp Nonpolar optimization method (Default = 1)

cavity_surften Surface tension. (Default = 0.00542 kcal/mol Angstrom^2). Unit conversion to *kJ* done automatically for APBS.

cavity_offset Offset value used to correct nonpolar free energy contribution (Default = -1.008) This is not used for APBS.

fillratio The ratio between the longest dimension of the rectangular finite-difference grid and that of the solute (Default = 4.0)

radiopt The option to set up atomic radii according to 0: the prmtop, or 1: pre-computed values (see Amber manual for more complete description). (Default = 0)

sander_apbs Option to use APBS for PB calculation instead of the built-in PBSA solver. This will work only through the iAPBS interface that creates sander.APBS. Instructions for this can be found online at the iAPBS/APBS websites. Allowed values are 0: Don't use APBS, or 1: Use sander.APBS. (Default = 0)

&alanine_scanning namelist variables

mutant_only Option to perform specified calculations only for the mutants. Allowed values are 0: Do mutant and original or 1: Do mutant only (Default = 0)

Note that all calculation details are controlled in the other namelists, though for alanine scanning to be performed, the namelist must be included (blank if desired)

&nmode namelist variables

dielc Distance-dependent dielectric constant (Default = 1.0)

drms Convergence criteria for minimized energy gradient. This value is used in the sander minimizations, but is multiplied by 10 for use in nmode. (Default = 0.0001)

maxcyc Maximum number of minimization cycles to use per snapshot in sander. (Default = 10000)

nmstartframe Frame number to begin performing nmode calculations on (Default = 1) *

nmendframe Frame number to stop performing nmode calculations on (Default = 1000000000) *

nminterval Offset from which to choose frames to perform nmode calculations on (Default = 1) *

ala_entropy Determines whether to perform nmode analysis on alanine scanning mutants or not. 0: Don't. 1: Do. (Default 1)

nmode_igb Value for Generalized Born model to be used in nmode calculations. 0: Vacuum, 1: GB model (Default 1)

nmode_istrng Ionic strength to use in nmode calculation in molarity. Can only be non-zero if nmode_igb=1.

* These variables will choose a subset of the frames chosen from the variables in the &general namelist. Thus, the "trajectory" from which snapshots will be chosen for nmode calculations will be the collection of snapshots upon which the other calculations were performed.

&decomp namelist variables (available only for Amber 11)

idecomp Energy decomposition scheme to use:
1 - Per-residue decomposition with 1-4 VDW and 1-4 EEL terms added to internal potential terms
2 - Per-residue decomposition with 1-4 VDW added to VDW and 1-4 EEL added to EEL potential terms
3 - Pair-wise decomposition with 1-4 VDW and 1-4 EEL added to internal potential terms
4 - Pair-wise decomposition with 1-4 VDW added to VDW and 1-4 EEL added to EEL potential terms
(No default. This must be specified!)

print_res Select residues from the complex prmtop to print. The receptor/ligand residues will be automatically figured out if ligand_mask and/or receptor_mask was omitted from the &general namelist! If you specify your own masks, you will need to provide your own mdin files and use the -use-mdins flag. If you do not use -use-mdins, MMPBSA.py will create template mdin files for you to edit and MMPBSA.py will terminate. You must then edit them, putting in the correct residue strings (see appropriate section of Amber manual), and re-run with the -use-mdins flag. Also note that per-residue energy changes will only be printed if the default masks are guessed by MMPBSA.py. This variable accepts semi-colon-delimited lists of residue numbers or ranges. For example: print_res = "1; 3-10; 15; 100". This will print residues 1, 3 through 10, 15, and 100 from the complex prmtop and the corresponding residues in either the ligand or receptor prmtops. (Default: print all residues). *

dec_verbose Set the level of output to print in the decomp_output file. 0: DELTA energy, total contribution only. 1: DELTA energy, total, sidechain, and backbone contributions. 2: Complex, Receptor, Ligand, and DELTA energies, total contribution only. 3: Complex,

Receptor, Ligand, and DELTA energies, total, sidechain, and backbone contributions (all data) (Default 0)

* Please note: Using idecomp=3 or 4 (pairwise) with a very large number of printed residues and a large number of frames can quickly create very, very large temporary mdout files. Large print selections will also demand a large amount of memory to analyze and compose the decomposition output file (~500 MB for 250 residues printing every pair). It is not unusual for the output file to take several minutes to compile if you print a large number of residues for a large number of frames.

Sample input files

```
Sample input file for GB and PB calculation
&general
   startframe=5, endframe=100, interval=5,
   verbose=2, keep_files=0,
/
&gb
   igb=5, saltcon=0.150,
/
&pb
   istrng=0.15, fillratio=4.0
/
------------------------------------------------------
Sample input file for Alanine scanning
&general
   verbose=2,
/
&gb
   igb=2, saltcon=0.10
/
&alanine_scanning
/
------------------------------------------------------
Sample input file with nmode analysis
&general
   startframe=5, endframe=100, interval=5,
   verbose=2, keep_files=2,
/
&gb
   igb=5, saltcon=0.150,
/
&nmode
   nmstartframe=2, nmendframe=20, nminterval=2,
   maxcyc=50000, drms=0.0001,
```

```
/
-------------------------------------------------------
Sample input file with decomposition analysis
&general
   startframe=5, endframe=100, interval=5,
/
&gb
   igb=5, saltcon=0.150
/
&decomp
   idecomp=2, dec_verbose=3,
   print_res="20; 40 - 80; 200"
/
```

A few important notes about input files. Comments are allowed by placing a # at the beginning of the line (whitespace is ignored). Variable initialization cannot span multiple lines. In-line comments (i.e. putting a # for a comment after a variable is initialized in the same line) is not allowed and will result in an input error. Variable declarations must be comma-delimited, though all whitespace is ignored (except for newline characters). Finally, all lines between namelists are ignored, so comments may be put before each namelist without using #.

1.4.2 Calling MMPBSA.py from the command-line

MMPBSA.py is invoked through the command line as follows:

```
MMPBSA.py {-O} -i input_file \
               -o output_file \
               -sp solvated_prmtop \
               -cp complex_prmtop \
               -rp receptor_prmtop \
               -lp ligand_prmtop \
               -y mdcrd1 mdcrd2 mdcrd3 ... mdcrdN \
              {-do decomp_output_file} \
              {-yr receptor_mdcrd1 ... receptor_mdcrdN}\
              {-yl ligand_mdcrd1 ... ligand_mdcrdN}\
              {-mc mutant_complex_prmtop}\
              {-mr mutant_receptor_prmtop}\
              {-ml mutant_ligand_prmtop}\
              {-slp solvated_ligand_prmtop}\
              {-srp solvated_receptor_prmtop}\
              {-make-mdins || -use-mdins || -rewrite-output}
```

Unless otherwise specified, default file names are those shown on the command-line. Do not put quotations around strings on the command line. Items shown above can only be placed on the command-line. All items in braces are optional. All others (except solvated_prmtop when strip_mdcrd=1) are mandatory unless you wish to use the default names. Optional files have no defaults.

-O If present, overwrite any existing output file.

-i Input file, default is no input file and all default values will be used.

-o Output file name (Default FINAL_RESULTS_MMPBSA.dat)

-sp Solvated complex topology file (unnecessary if strip_mdcrd=0).

-cp Complex topology file

-rp Receptor topology file

-lp Ligand topology file

-y Comma- and/or whitespace-delimited list of complex trajectory files to analyze (Default = mdcrd)

-do Decomposition output file name (Default FINAL_DECOMP_MMPBSA.dat)

-yr Comma- and/or whitespace-delimited list of receptor trajectory files to analyze

-yl Comma- and/or whitespace-delimited list of ligand trajectory files to analyze

-mc Mutant complex topology file. Default is shown if &alanine_scanning is specified

-mr Mutant receptor topology file. No default, as mutation can be in either receptor or ligand.

-ml Mutant ligand topology file. No default, as mutation can be in either receptor or ligand.

-slp Solvated ligand topology file, which is required if -yl is specified and strip_mdcrd=1

-srp Solvated receptor topology file, which is required if -yr is specified and strip_mdcrd=1

-make-mdins This option will cause MMPBSA.py to create all mdin files used by sander and exit so they can be edited (see Advanced Options)

-use-mdins This option will cause MMPBSA.py to use existing mdin files for sander rather than creating them and using those (see Advanced Options)

-rewrite-output This option will cause MMPBSA.py to use existing temporary output files to compile the final output file containing the MM/PBSA statistics. This allows you to re-print the output file without having to re-run the calculation. However, you must still provide the appropriate topology files and input file so MMPBSA.py can determine what it should be printing (and it also extracts information from these files to print to the final output files). It will compile both the regular and decomposition output files if specified. This is especially useful for changing the level of verbosity (see input sections below). See description in “Advanced Options” below for a more thorough discussion.

–help (Also invoked by -h) Display usage statement and quit

–clean (Also invoked by –clear) Remove all temporary files that MMPBSA.py creates (to clean up after a previous calculation)

The last two options should appear alone on the command-line after MMPBSA.py if they are to be used. Also, if an input file is specified along with -make-mdins, then the script will make all mdin files pertinent to the input file and quit. This is, for example, the only way to create an input file for use with sander.APBS that you wish to edit by hand (you must put sander_apbs=1 in the &pb namelist).

MMPBSA.py.MPI can be invoked similar to other parallel executables in Amber. See More information on parallel machines or clusters in the Amber manual for more details.

1.4.3 The Output File

The header of the output file will contain information about the calculation. It will show a copy of the input file as well as all files that were used in the calculation (topology files and coordinate file(s)). If the masks were not specified, it prints its best guess so that you can verify its accuracy, along with the residue name of the ligand (if it is only a single residue).

The energy and entropy contributions are broken up into their components as they are in sander and nmode or ptraj. The contributions are further broken into G_{gas} and G_{solv}. The polar and non-polar contributions are EGB (or EPB) and ESURF (or ECAVITY / ENPOLAR), respectively for GB (or PB) calculations.

By default, bonded terms are not printed for any one-trajectory simulation. They are computed and their differences calculated, however. They are not shown (nor included in the total) unless specifically asked for because they should cancel completely. A single trajectory does not produce any differences between bond lengths, angles, or dihedrals between the complex and receptor/ligand structures. Thus, when subtracted they cancel completely. This includes the BOND, ANGLE, DIHED, and 1-4 interactions. If inconsistencies are found, these values are displayed and inconsistency warnings are printed. When this occurs the results are generally useless. Of course this does not hold for the multiple trajectory protocol, and so all energy components are printed in this case.

Finally, all warnings generated during the calculation that do not result in fatal errors are printed at the bottom of the output file.

1.4.4 Temporary Files

MMPBSA.py creates working files during the execution of the script beginning with the prefix _MMPBSA_. The variable “keep_files” controls how many of these files are kept after the script finishes successfully. If the script quits in error, all files will be kept. You can clean all temporary files from a directory by running MMPBSA.py –clean described above.

If MMPBSA.py does not finish successfully, several of these files may be helpful in diagnosing the problem. For that reason, every temporary file is described below. Note that not every temporary file is generated in every simulation. At the end of each description, the lowest value of “keep_files” that will retain this file will be shown in parentheses.

_MMPBSA_gb.mdin Input file that controls the GB calculation done in *sander*. (2)

_MMPBSA_pb.mdin Input file that controls the PB calculation done in *sander*. (2)

_MMPBSA_cenptraj.in Input file that extracts requested snapshots from the given mdcrd files, centers and images the complex to correct for imaging artifacts, strips water and ions, and dumps the resulting snapshots into a temporary mdcrd file. This file is processed by *ptraj*. (2)

_MMPBSA_complex.mdcrd Trajectory file printed out by _MMPBSA_cenptraj.in that contains only those snapshots that will be processed by MMPBSA.py. (1)

_MMPBSA_ligtraj.in Input file that processes ligand trajectories the same way as _MMPBSA_cenptraj.in does above if ligand trajectories are supplied. Otherwise, it loads the _MMPBSA_complex.mdcrd, strips the receptor mask, and dumps the ligand trajectory. This file is processed by *ptraj*. (2)

_MMPBSA_ligand.mdcrd Trajectory file printed out by _MMPBSA_ligtraj.in that contains only those snapshots that will be processed by MMPBSA.py. (1)

_MMPBSA_rectraj.in Input file that processes receptor trajectories the same way as _MMPBSA_cenptraj.in does above if receptor trajectories are supplied. This file is processed by *ptraj*. (2)

_MMPBSA_receptor.mdcrd Trajectory file printed out by _MMPBSA_rectraj.in that contains only those snapshots that will be processed by MMPBSA.py (1)

_MMPBSA_complexinpcrd.in Input file that extracts the first frame from _MMPBSA_complex.mdcrd to use as a dummy inpcrd file for the GB and PB calculations. This is necessary for using imin=5 functionality in sander. This file is processed by *ptraj*. (2)

_MMPBSA_receptorinpcrd.in Same as above, but for receptor. (2)

_MMPBSA_ligandinpcrd.in Same as above, but for ligand. (2)

_MMPBSA_dummycomplex.inpcrd Dummy inpcrd file generated by _MMPBSA_complexinpcrd.in for use with imin=5 functionality in sander. (1)

_MMPBSA_dummyreceptor.inpcrd Same as above, but for the receptor. (1)

_MMPBSA_dummyligand.inpcrd Same as above, but for the ligand. (1)

_MMPBSA_complex_nm.in Input file that extracts complex snapshots and dumps them into separate restart files. This file is processed by *ptraj*. (2)

_MMPBSA_rectraj_nm.in Same as above, but for receptor. (2)

_MMPBSA_ligtraj_nm.in Same as above, but for ligand. (2)

_MMPBSA_complex_nm.inpcrd.# Inpcrd files that are the extracted snapshots by _MMPBSA_complex_nm.in. (1)

_MMPBSA_receptor_nm.inpcrd.# Inpcrd files that are extracted snapshots by _MMPBSA_rectraj_nm.in. (1)

_MMPBSA_ligand_nm.inpcrd.# Inpcrd files that are extracted snapshots by _MMPBSA_ligtraj_nm.in. (1)

_MMPBSA_ptrajentropy.in Input file that calculates the entropy via the quasi-harmonic approximation. This file is processed by *ptraj*. (2)

_MMPBSA_avgcomplex.pdb PDB file containing the average positions of all complex conformations processed by _MMPBSA_cenptraj.in. It is used as the reference for the _MMPBSA_ptrajentropy.in file above. (1)

_MMPBSA_complex_entropy.out File into which the entropy results from _MMPBSA_ptrajentropy.in analysis on the complex are dumped. (1)

_MMPBSA_receptor_entropy.out Same as above, but for the receptor. (1)

_MMPBSA_ligand_entropy.out Same as above, but for the ligand. (1)

_MMPBSA_ptraj1.out Output from running *ptraj* using _MMPBSA_cenptraj.in. (1)

_MMPBSA_ptraj2.out Output from running *ptraj* using _MMPBSA_ligtraj.in. (1)

_MMPBSA_ptraj3.out Output from running *ptraj* using _MMPBSA_rectraj.in. (1)

_MMPBSA_ptraj4.out Output from running *ptraj* using _MMPBSA_complexinpcrd.in. (1)

_MMPBSA_ptraj5.out Output from running *ptraj* using _MMPBSA_receptorinpcrd.in. (1)

_MMPBSA_ptraj6.out Output from running *ptraj* using _MMPBSA_ligandinpcrd.in. (1)

_MMPBSA_ptraj7.out Output from running *ptraj* using _MMPBSA_mutant_ligtraj.in. (1)

_MMPBSA_ptraj8.out Output from running *ptraj* using _MMPBSA_mutant_rectraj.in. (1)

_MMPBSA_ptraj9.out Output from running *ptraj* using _MMPBSA_mutant_complexinpcrd.in. (1)

_MMPBSA_ptraj10.out Output from running *ptraj* using _MMPBSA_mutant_receptorinpcrd.in. (1)

_MMPBSA_ptraj11.out Output from running *ptraj* using _MMPBSA_mutant_ligandinpcrd.in. (1)

_MMPBSA_ptraj12.out Output from running *ptraj* using _MMPBSA_complex_nm.in. (1)

_MMPBSA_ptraj13.out Output from running *ptraj* using _MMPBSA_ligtraj_nm.in. (1)

_MMPBSA_ptraj14.out Output from running *ptraj* using _MMPBSA_rectraj_nm.in (1)

_MMPBSA_ptraj15.out Output from running *ptraj* using _MMPBSA_mutant_complex_nm.in. (1)

_MMPBSA_ptraj16.out Output from running *ptraj* using _MMPBSA_mutant_ligtraj_nm.in. (1)

_MMPBSA_ptraj17.out Output from running *ptraj* using _MMPBSA_mutant_rectraj_nm.in (1)

_MMPBSA_ptraj_entropy.out Output from running *ptraj* using _MMPBSA_ptrajentropy.in. (1)

_MMPBSA_complex_gb.mdout *sander* output file containing energy components of all complex snapshots done in GB. (1)

_MMPBSA_receptor_gb.mdout *sander* output file containing energy components of all receptor snapshots done in GB. (1)

_MMPBSA_ligand_gb.mdout *sander* output file containing energy components of all ligand snapshots done in GB. (1)

_MMPBSA_complex_pb.mdout *sander* output file containing energy components of all complex snapshots done in PB. (1)

_MMPBSA_receptor_pb.mdout *sander* output file containing energy components of all receptor snapshots done in PB. (1)

_MMPBSA_ligand_pb.mdout *sander* output file containing energy components of all ligand snapshots done in PB. (1)

_MMPBSA_pbsanderoutput.junk File containing the information dumped by sander.APBS to STDOUT. (1)

_MMPBSA_ligand_nm.pdb.# Restart file of minimized ligand snapshot prepared for *nmode*. (1)

_MMPBSA_complex_nm.pdb.# Restart file of minimized complex snapshot prepared for *nmode*. (1)

_MMPBSA_receptor_nm.pdb.# Restart file of minimized receptor snapshots prepared for *nmode*. (1)

_MMPBSA_ligand_nm.out Output file from *nmode* that contains the entropy data for the ligand and for all snapshots. (1)

_MMPBSA_receptor_nm.out Output file from *nmode* that contains the entropy data for the receptor for all snapshots. (1)

_MMPBSA_complex_nm.out Output file from *nmode* that contains the entropy data for the complex for all snapshots. (1)

_MMPBSA_mutant_... These files are analogs of the files that only start with _MMPBSA_ described above, but instead refer to the mutant system.

1.4.5 Advanced Options

The default values for the various parameters as well as the inclusion of some variables over others in the general MMPBSA.py input file were chosen to cover the majority of all MM/PB(GB)SA calculations that would be attempted while maintaining maximum simplicity. However, there are situations in which MMPBSA.py may appear to be restrictive and ill-equipped to address. Attempts were made to maintain the simplicity described above while easily providing users with the ability to modify most aspects of the calculation easily and without editing the source code.

-use-mdins This flag will prevent MMPBSA.py from creating the input files that control the various calculations (_MMPBSA_gb.mdin, _MMPBSA_pb.mdin, _MMPBSA_sander_nm_min.mdin, and _MMPBSA_nmode.in). It will instead attempt to use existing input files (though they must have those names above!) in their place. In this way, the user has full control over the calculations performed, however care must be taken. The mdin files created by MMPBSA.py have been tested and are (generally) known to be consistent. Modifying certain variables (such as imin=5) may prevent the script from working, so this should only be done with care. It is recommended that users start with the existing mdin files (generated by the flag below), and add and/or modify parameters from there.

-make-mdins This flag will create all of the mdin and input files used by sander and nmode so that additional control can be granted to the user beyond the variables detailed in the input file section above. The files created are _MMPBSA_gb.mdin which controls GB calculation; _MMPBSA_pb.mdin which controls the PB calculation; _MMPBSA_sander_nm_min.mdin which controls the sander minimization of snapshots to be prepared for nmode calculations; and _MMPBSA_nmode.in which controls the nmode calculation. If no input file is specified, all files above are created with default values, and _MMPBSA_pb.mdin is created for AmberTools's *pbsa*. If you wish to create a file for sander.APBS, you must include an input file with "sander_apbs=1" specified to generate the desired input file. Note that if an input file is specified, only those mdin files pertinent to the calculation described therein will be created!

-rewrite-output This flag will compile the output files from _MMPBSA_complex_gb.mdout, _MMPBSA_receptor_gb.mdout, _MMPBSA_ligand_gb.mdout, the corresponding 'pb' files, the nmode output files, and all _MMPBSA_mutant_ versions of each. The _MMPBSA_ptraj__.out files must also exist, as those are parsed to obtain the number of frames analyzed by each method (_MMPBSA_ptraj1.out for gb, pb, and alanine scanning, and _MMPBSA_ptraj12.out for normal mode analysis). All of these files will be kept by setting keep_files=1 in the &general namelist, which is its default value. The only information gleaned from the input file is the values of verbose/dec_verbose, and which types of calculations are run (gb, pb, decomp, nmode, using sander.APBS, etc.). It uses the input file to determine which files to look for. Thus, changing startframe, endframe, interval, etc. will NOT change the subset of frames used to form the final statistics.

strip_mask This input variable allows users to control which atoms are stripped from the trajectory files associated with solvated_prmtop. In general, counterions and water molecules are stripped, and the complex is centered and imaged (so that if iwrap caused the ligand

to "jump" to the other side of the periodic box, it is replaced inside the active site). If there is a specific metal ion that you wish to include in the calculation, you can prevent ptraj from stripping this ion by NOT specifying it in *strip_mask*.

strip_mdcrd strip_mdcrd=1 is used if there are no molecules to be stripped from the initial trajectory file (rendering solvated_prmtop unnecessary). This is particularly useful if you wish to perform more complex MM/PBSA calculations. For instance, if there is a bound water molecule, or a particular bound ion that is important to include in either the receptor or ligand, then you must pre-process the trajectory file such that it includes precisely those water(s) and/or ion(s) that you wish to keep. They must also be present in the topology files such that the trajectories and topology files remain consistent.

Please send any bug reports, comments, or suggestions to mmpbsa.amber@gmail.com. Thanks!

Bibliography

[1] J. Srinivasan, T.E. Cheatham, III, P. Cieplak, P. Kollman, and D.A. Case. Continuum solvent studies of the stability of DNA, RNA, and phosphoramidate–DNA helices. *J. Am. Chem. Soc.*, 120:9401–9409, 1998.

[2] J. Weiser, P.S. Shenkin, and W.C. Still. Approximate Atomic Surfaces from Linear Combinations of Pairwise Overlaps (LCPO). *J. Computat. Chem.*, 20:217–230, 1999.