AMBER Drug Discovery Boost Package

User Guide

Laboratory for Biomolecular Simulation Research
Rutgers University

February 19, 2025

1 The AMBER Drug Discovery Boost package

The AMBER Drug Discovery Boost package consists of three components, 1) `FE-MDEngine`, 2) `FE-ToolKit`, and 3) `FE-Workflow`. `FE-MDEngine` consists of the current release version of AmberTools21 and a specialized version of AMBER20 (the pmemd package) that contains the newest technologies related to alchemical free energy methods that are not yet available in the current release version of AMBER (AMBER20). The AMBER License and "patch" code mechanism enables anyone with a current AMBER license to gain advanced access to the AMBER DD Boost package for beta testing and validation prior to the official integration of these new methods into AMBER (which occurs on a 2-year cycle). `FE-ToolKit` consists of a collection of programs that implement the latest approaches for analyzing free energy simulations, and `FE-Workflow` consists of a collection of programs that are useful for the setup of alchemical free energy simulations (relative binding free energy (RBFE), relative solvation free energy (RSFE), and absolute solvation free energy (ASFE)) using `FE-MDEngine`, and analysis of such simulations using `FE-Workflow`.

Below is a list of recent references related to the different features within AMBER DD Boost package, as of February 19, 2025.

1.

2.

3.

4.

5.

6.

2 Availability of the AMBER Drug Discovery Boost package

The AMBER DD Boost package can be accessed from GitLab repositories set up through the Laboratory for Biomolecular Simulation Research (LBSR) at Rutgers (details are provided below). As mentioned earlier, `FE-MDEngine` consists of AmberTools21 and a specialized version of AMBER20. AmberTools21 is free of charge, and can also be downloaded directly from AmberMD.org. The AMBER20 part of `FE-MDEngine` is available only to AMBER20 license holders. `FE-ToolKit` and `FE-Workflow` are available to LBSR GitLab members (beta testers) and is currently available at no cost for non-commercial use. Active Amber developers can also access `FE-MDEngine` through the Amber development GitLab repository, which mirrors `FE-MDEngine` in the LBSR GitLab repository.

In order to access the LBSR GitLab repositories housing the AMBER DD Boost package, a verified AMBER20 license holder will need an active GitLab account. A new GitLab ccount can easily be obtained at www.GitLab.com. The user first needs to send the e-mail address/username associated with the GitLab account to: Shi Zhang at sz550@rutgers.edu or Darrin York at Darrin.York@rutgers.edu in order to be added to the GitLab repositories.

Note: if you have created a new GitLab account through a social media account such as Google or Facebook, you will need to manually set up your GitLab password in order for git clone to work.

Once added, the user will receive three separate notification emails confirming that the user has been added to the following three projects:

Laboratory for Biomolecular Simulation Research / `FE-MDEngine`
Laboratory for Biomolecular Simulation Research / `FE-ToolKit`
Laboratory for Biomolecular Simulation Research / `FE-Workflow`

Upon this confirmation the user will be able to check out the packages as follows:

With ssh-key setup in GitLab (recommended):
git clone git@gitlab.com:RutgersLBSR/`FE-MDEngine`.git
git clone git@gitlab.com:RutgersLBSR/`FE-Workflow`.git
git clone git@gitlab.com:RutgersLBSR/`FE-ToolKit`.git

Without ssh-key setup in GitLab:
git clone https://gitlab.com/RutgersLBSR/`FE-MDEngine`.git
git clone https://gitlab.com/RutgersLBSR/`FE-Workflow`.git
git clone https://gitlab.com/RutgersLBSR/`FE-ToolKit`.git

> *Note*: We recommend that you create one directory, e.g. GitLab, and clone all three repositories inside that directory.

3 Installation of `FE-MDEngine`

The installation process of `FE-MDEngine` is identical to that of AMBER20, and can be installed by running the script *install_FE-MDEngine.sh* located in the `FE-MDEngine`directory. Further instructions can be found here https://ambermd.org/Installation.php

> *Note*: In order to use `FE-Workflow` for AFE simulation setup, the installation of only the serial (without CUDA) version of `FE-MDEngine` is required, while to perform the AFE calculations setup by `FE-Workflow`, the installation of all versions of `FE-MDEngine`, i.e. serial and parallel (without CUDA) and serial and parallel (with CUDA) are required.

4 Installation of `FE-ToolKit`

`FE-ToolKit` can be installed by running the *INSTALL.sh* script located inside `FE-ToolKit` directory.

5 Purpose of the `FE-Workflow`

`FE-Workflow` consists of a collection of scripts that are designed to facilitate the setup, execution, and analysis of alchemical free energy (AFE) simulations using AMBER DD Boost. Currently, `FE-Workflow` can be used to perform relative binding free energy (RBFE), relative solvation free energy (RSFE), and absolute solvation free energy (ASFE) calculations. The scripts use a simplified input file, which is described in detail later in the user-guide, that provides top-level control on various important aspects of the intended AFE simulations.
Briefly, for a given system, such as a specific protein target, or a collection of small molecules, a) a list of desired transformations can be provided, b) key simulation settings can be specified, and c) initial

configuration files (MD equilibrated parameter (parm) and coordinate (rst) files associated with the system and specified transformations) must be provided. `FE-Workflow` can then be used to generate a hierarchy of directories containing relevant parameter, coordinate, and AMBER input files and job submission scripts. `FE-Workflow` can also be used to analyze the free energy simulations using `FE-ToolKit`.

6 Contents of the `FE-Workflow` repository

The `FE-Workflow` repository contains the following folders:

- *UserGuide* - containing an user guide (this document) for setting up AFE with `FE-MDEngine` using `FE-Workflow`
- *Examples* - containing test cases for RBFE and RSFE calculations
- *bin* - containing scripts related to `FE-Workflow`

We are also in the process of putting documentation up on the Wiki site that will be updated on a regular basis - `https://gitlab.com/RutgersLBSR/alchemical_fe/-/wikis/Setup-AFE_AMBER_DD_BOOST`

7 Initial requirements of using the `FE-Workflow`

Following conditions must be met before using the `FE-Workflow` -

- AMBER should be installed and *AMBERHOME* should be defined. The AmberTools *cpptraj* and *parmed* are used by `FE-Workflow` during the setup process.
  - We recommend installing AmberTools21 that comes with `FE-MDEngine`, however a standalone installation of AmberTools21 should also work.
  - For instructions related to download and installation of AmberTools, see `https://ambermd.org/AmberTools.php`
- *bin* subdirectories within `FE-Workflow` and `FE-ToolKit` should be available in $PATH
- `FE-ToolKit` must be installed when using `FE-Workflow` in "analysis" mode.
- *python3* needs to be installed and available in $PATH
- The software *RDKit* must be installed and python3 bindings to *RDKit* should be available in $PYTHONPATH. *RDKit* is a popular open-source cheminformatics software, and is used by the atom-mapping algorithms within `FE-Workflow`.
  - *RDKit* can be installed with a package manager
    - Fedora - *sudo dnf install rdkit.x86_64 python3-rdkit.x86_64*
    - Ubuntu - *sudo apt install python-rdkit*
  - or, *RDKit* can be installed using *conda*
    - See `http://www.rdkit.org/docs/Install.html#installationfordetails`
- folder containing initial structure and parameter files

– For relative binding free energy (RBFE) calculations, this folder should contain, for each intended transformation (edge), PDB file(s) of the protein-ligand(s) complex(es) and parameter (mol2, lib, frcmod) files for the associated ligands and any nonstandard residues present in the protein-ligand PDB file(s).
For relative and absolute solvation free energy (RSFE, ASFE) calculations, this folder should contain, for each intended transformation, parameter (mol2, lib, frcmod) files for the associated ligand(s).
For further details related to input format please refer to Table 1

8 Usage of the FE-Workflow

The *setup_fe* script represents the main executable of the FE-Workflow, and can be created by running the script *makesetup_fe.sh* located in the FE-Workflow repository.

./makesetup_fe.sh

makesetup_fe.sh creates in FE-Workflow directory a bashrc file named FE-Workflow.bashrc that should be sourced before executing *setup_fe*. In the default setup, *setup_fe* is meant to be kept in the FE-Workflow/bin directory and, since FE-Workflow/bin is added to the $PATH variable, *setup_fe* should be available as a command line program. In most cases, this should be the most convenient way of using *setup_fe*. However, if needed, the location of *setup_fe* can be changed by changing the "$path" variable in *setup_fe* accordingly.

9 Input file for *setup_fe*

setup_fe expects a inputfile named *input* in the directory where *setup_fe* is executed (or, the directory pointed by the "$path" variable in *setup_fe*) that contains key settings of the alchemical free energy simulations that are going to be set up. A template input file can be generated by running the script with a flag *-h* or *-help*.

setup_fe -h

A typical input file looks like the following -

```
path_to_input=initial
system=CDK2
setupmode=0
ticalc=rbfe
stage=analysis
translist=(1h1q~1h1s 1h1q~1oi9 1oi9~1h1s)
mapmethod=2
mapinspect=0
mapnetwork=true
boxbuild=1
boxbufcom=16
boxbufaq=20
ionconc=0.15
pff=ff14SB
lff=gaff2
wm=tip4pew
```

mdboxshape=cubic
nlambda=25
lamschedule=yes
lams=(0 0.176834 0.229764 0.269379 0.302697 0.33229 0.359436 0.384886 0.40913 0.432518 0.455318 0.477748 0.5 0.522252 0.544682 0.567482 0.59087 0.615114 0.640564 0.66771 0.697303 0.730621 0.770236 0.823166 1)
protocol=unified
ntrials=3
cutoff=10
repex=true
nstlimti=5000
numexchgti=1000
hmr=false
notrajectory=true
scalpha=0.5
scbeta=1.0
gti_add_sc=5
gti_scale_beta=1
gti_cut=1
gti_cut_sc_on=8
gti_cut_sc_off=10
gti_lam_sch=1
gti_ele_sc=1
gti_vdw_sc=1
gti_cut_sc=2
gti_ele_exp=2
gti_vdw_exp=2
twostate=false
bidirection_aq=true
bidirection_com=true
partition=general-long-gpu
nnodes=1
ngpus=8
wallclock=3-00:00:00
path_to_data=data
exptdatafile=skip
bar=true
ccc=false
start=0.0
stop=100.0
check_convergence=true

Table 1 provides a detailed description of keywords that are specific to this input file for *setup_fe*. For keywords/flags that are specific to AMBER20 and AMBER-DD Boost refer to the AMBER20 reference manual and Table 2.

10 File infrastructure created by *setup_fe*

In **setup mode** (keyword *stage* set to 'setup' in input), *setup_fe* creates a folder named "system", as defined in the *input* file. The "system" folder will have two subdirectories, setup and run. The former (setup) will house the various intermediate files that were generated and used in creating the final input files, while the latter (run) will contain independent subdirectories corresponding to each entry (transformation) in the keyword *"translist"* in the *input* file. These subdirectories that will have the same naming convention as provided in the *input* file, will further contain subdirectories named com and aq for RBFE calculations, or only the subdirectory aq for RSFE and ASFE calculations. The com and aq subdirectories correspond to complex and aqueous simulations, respectively, and will contain the final merged TI parameter and coordinate input files, template submission slurm scripts, a folder named inputs containing relevant AMBER input files, and production sub-folders corresponding to each specified independent trial that will house the production simulation data.

In the workflow, the starting structures are subjected to an exhaustive equilibration protocol that consists of two broad phases. In phase I, only the endstate(s) are considered (i.e. only the λ=0 state for *1-state setup* and only the λ=0 and λ=1 states for *2-state setup*. The endstates are equilibrated thoroughly using a series of minimization, constant NVT, and constant NPT simulations with varying restraints on the solute to ensure proper equilibration. Protein-ligand simulations are subjected to a longer phase I equilibration with additional steps compared to simulations of only ligand in water or vacuum (for ASFE simulations). In phase II, from the equilibrated end-point structures, all other intermediate λ windows are generated and further equilibrated, again using a series of short minimization and constant NPT simulations to generate the starting structures for the production TI simulations.

> *Note*: The equilibration protocol, that is the order in which various equilibration steps are intended to be carried out, can be found inside the *run_alltrials.slurm* script (assigned to the *eqstage* variable), generated in the "system"/run/"com" and "system"/run/"aq" folders.

In **analysis mode** (keyword *stage* set to 'analysis' in input), *setup_fe* creates a folder named results in the directory in which *setup_fe* is executed. The results folder will contain a subdirectory data that will have a nested directory structure containing Energy and DV/DL data from the various simulations in the transformation network being analyzed. The results folder will also contain the graphmbar input file named *graphmbar.inp* for `FE-ToolKit`, a python script named *gmbar.py* that facilitates the generation of the graphmbar input file, the graphmbar output file named *graphmbar.out* generated by `FE-ToolKit`, and a simplified output file named *out* summarizing the final free energy results of the entire transformation network.

11 An example usage of *setup_fe*

Once `FE-MDEngine` and `FE-ToolKit` have been installed, and *setup_fe* has been created using the instructions above, you can test the `FE-Workflow` by creating a test working directory and copying an example input file provided in Examples subdirectory within the `FE-Workflow` repository. Go to the test directory, rename the specific *input.** to input, modify the variable *path_to_input* in input to the location of the initial subdirectory within the `FE-Workflow` repository, source the `FE-Workflow`.bashrc file in bin subdirectory within the `FE-Workflow` repository, and execute *setup_fe*.

```
mkdir -p test-run

cd test-run
```

cp path-to-**FE-Workflow**/Examples/rbfe/input.CDK2 ./input

vi input #change path_to_input=path-to-**FE-Workflow**/Examples/intial

setup_fe

Table 1. Keywords associated with the *ProFESSA* workflow

Keyword	Value	Description	Example
path_to_input	string	Path to directory that contains input files. It should contain a subdirectory *system*.	path_to_input =/home/username/afe/initial
system	string	Name of the system. A folder named system should be present in *path_to_input/*, and should contain the initial structure and parameter files.	system=CDK2
translist	list of strings	A list of desired transformations or edges. In the case of RBFE or RSFE calculations, *translist* should be a list in which each entry consists of two molnames separated by the character "~", while in the case of ASFE calculations, *translist* should be a list of molnames. Initial structure/parameter files of these molnames should be provided in *path_to_input/system* and should be named as follows: For RBFE calculations, each molname present in *translist* should have an associated *molname.pdb*, representing the receptor-ligand complex structure and [*molname_0.mol2 molname_0.lib molname_0.frcmod*], representing the ligand parameters. Parameters of additional non-standard residues, if present, can be provided as [*molname_1.mol2 molname_1.lib molname_1.frcmod*], [*molname_2.mol2 molname_2.lib molname_2.frcmod*], etc. For RSFE and ASFE calculations, each molname present in *translist* should have associated [*molname_0.mol2 molname_0.lib molname_0.frcmod*] files.	translist=(1h1q~1h1r 1h1q~1h1s)
ticalc	string	Specifies nature of calculation. Acceptable values - *rbfe*, *rsfe*, *asfe*.	ticalc=rbfe
nlambda	integer	Number of lambda windows in TI calculation. Acceptable values - positive integers.	nlambda=5
lamschedule	string	Specifies if user-defined λ schedule will be used. Acceptable values - *yes*, *no*.	lamschedule=yes
lams	list of floats	Specifies the specific λ values that will be used when *lamschedule* is set to *yes*.	lams=(0 0.25 0.5 0.75 1)
override_lambda	string	Specifies directory from where lambdas can be read.. Acceptable values - *null*, *path to directory*.	override_lambda=false
protocol	string	Protocol for running TI simulations. Acceptable value - *unified*.	protocol=unified
mapmethod	integer	Specifies the algorithm using which TS/TC regions are going to be determined. Acceptable values - *0*, *1*, *2*. *0* specifies MCS algorithm. *1* specifies MCS-E algorithm. *2* specifies MCS-E2 algorithm. For a given transformation/edge *molname1~molname2* in *translist*, an atom map file, *molname1~molname2.map.txt*, is generated in the folder *system/setup*.	mapmethod=1
mapinspect	integer	Allows manual inspection of TS/TC regions. Acceptable values - *0, 1, 2.* *0* specifies no manual inspection. *1* specifies manual inspection. *2* specifies resume setup assuming manual inspection has been completed. If *mapinspect* is set to *2*, the necessary atom map files should be present in *system/setup* folder.	mapinspect=true

Continued on next page

Table 1 – continued from previous page

Keyword	Value	Description	Example
mapnetwork	string	Specifies if network-wide consistent TS/TC regions of ligands will be generated. Acceptable values - *true*,*false*.	mapnetwork=false
boxbuild	string / integer	Specifies if and how MD boxes will be built. Acceptable values - *0*, *1*, *2*, *skip*. *0* specifies to build boxes only for "aqueous" state and not for "complex" state. *1* specifies to build boxes for both "aqueous" and "complex" states. *2* specifies to build boxes for both "aqueous" and "complex" states with identical number of water and ions. *skip* specifies to skip box building altogether. For RSFE and ASFE calculations, *boxbuild 0* and *1* are identical.	boxbuild=1
boxbufcom	integer	Specifies MD box buffer for "complex" states. Relevant only for RBFE calculations. Acceptable values - positive integers.	boxbufcom=16
boxbufaq	integer	Specifies MD box buffer for "aqueous" states. Relevant only for RBFE calculations. Acceptable values - positive integers.	boxbufaq=21
ionconc	float	Specifies the MD box ion concentration in units of mol/L (M). Acceptable values - positive real number.	ionconc=0.15
pff	string	Specifies protein forcefield. Acceptable values - *ff14SB*	pff=ff14SB
lff	string	Specifies ligand forcefield. Acceptable values - *gaff*, *gaff2*	lff=gaff2
wm	string	Specifies water model. Acceptable values - *tip4pew*, *tip3p*	wm=tip4pew
mdboxshape	string	Specifies shape of MD box. Acceptable values - *cubic*	mdboxshape=cubic
ntrials	integer	Specifies the number of independent trials of calculation. Acceptable values - positive integers.	ntrials=10
cutoff	integer	Specifies non-bonded cutoff in TI simulations. Acceptable values - positive integers.	cutoff=10
repex	string	Specifies if Hamiltonian Replica Exchange will be employed.	repex=true
nstlimti	integer	Specifies the length of production TI simulations in units of fs. Acceptable values - positive integers.	nstlimti=5000
numexchgti	integer	Specifies the number of exchanges in replica exchange TI simulations. *numexchgti* is ignored is *repex* is set to false. Acceptable values - positive integers.	numexchgti=1000
hmr	string	Specifies if Hydrogen Mass Repartitioning will be used. Acceptable values - *true*, *false*.	hmr=false
notrajectory	string	Specifies if production trajectories will be saved during TI simulations. Acceptable values - *true*, *false*.	notrajectory=true
max_dt	float	Specifies the maximum timestep (default is 0.002 for not hmr, 0.004 for hmr)	max_dt=0.002/0.004
ntwx	integer	Specifies the frequency for which TI trajectory frames will be saved	ntwx=0
ntwx_equil	integer	Specifies the frequency for which TI trajectory frames will be saved for equil runs (note - careful choice recommended)	ntwx=0
ntwr	integer	Specifies the frequency for which TI restart files will be saved	ntwr=nstlimti
ntpr	integer	Specifies the frequency for which TI energies will be saved	ntpr=nstlimti
scalpha	float	Specifies the value of *AMBER DD BOOST* keyword *scalpha* in TI simulations. Acceptable values - positive real numbers.	scalpha=0.5
scbeta	float	Specifies the value of *AMBER DD BOOST* keyword *scbeta* in TI simulations. Acceptable values - positive real numbers.	scbeta=0.5

Continued on next page

Table 1 – continued from previous page

Keyword	Value	Description	Example
gti_add_sc	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_add_sc* in TI simulations. Acceptable values - positive integers.	gti_add_sc=5
gti_scale_beta	float	Specifies the value of *AMBER DD BOOST* keyword *gti_scale_beta* in TI simulations. Acceptable values - positive real number.	gti_scale_beta=1
gti_cut	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_cut* in TI simulations. Acceptable values - positive integers.	gti_cut=1
gti_cut_sc_on	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_cut_sc_on* in TI simulations. Acceptable values - positive integers.	gti_cut_sc_on=8
gti_cut_sc_off	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_cut_sc_off* in TI simulations. Acceptable values - positive integers.	gti_cut_sc_off=10
gti_lam_sch	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_lam_sch* in TI simulations. Acceptable values - positive integers.	gti_lam_sch=1
gti_ele_sc	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_ele_sc* in TI simulations. Acceptable values - positive integers.	gti_ele_sc=1
gti_vdw_sc	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_vdw_sc* in TI simulations. Acceptable values - positive integers.	gti_vdw_sc=1
gti_cut_sc	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_cut_sc* in TI simulations. Acceptable values - positive integers.	gti_cut_sc=1
gti_ele_exp	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_ele_exp* in TI simulations. Acceptable values - positive integers.	gti_ele_exp=2
gti_vdw_exp	integer	Specifies the value of *AMBER DD BOOST* keyword *gti_vdw_exp* in TI simulations. Acceptable values - positive integers.	gti_vdw_exp=2
twostate	string	Specifies if twostate setup will be employed in TI simulations. Acceptable values - *true*, *false*.	twostate=true
bidirection_aq	string	Specifies if bidirectional setup will be used for "aqueous" state TI simulations. Applicable when *twostate* is set to *false*. Acceptable values - *true*, *false*.	bidirection_aq=false
bidirection_com	string	Specifies if bidirectional setup will be used for "complex" state TI simulations. Applicable when *twostate* is set to *false*. Acceptable values - *true*, *false*.	bidirection_aq=false
stage	string	Specifies the action of the script. Acceptable values - *setup*, *analysis*. *setup* specifies script to set up TI simulations. *analysis* specifies script to perform analysis.	stage=setup
setupmode	integer	Specifies the mode of simulation setup. Acceptable values - *0* *0* sets up regular TI simulations.	setupmode=0
equil_type	integer	Specifies whether a single equilibration (equil_type=1) or a trial-dependent equilibration (equil_type=2) will be performed. Acceptable values - *1*, *2*.	equil_type=2
source_header	string	Specifies the path to the header file that will be sourced before running the TI simulations. Acceptable values - *null*, *path to header file*.	source_header=null
partition	string	Specifies the HPC partition on which TI runs will be performed. Acceptable values - *null*, *name of HPC partition*.	partition=gpu
nnodes	integer	Specifies the number of nodes to be requested for a single set of TI simulations. Acceptable values - positive integer.	nnodes=1

Continued on next page

Table 1 – continued from previous page

Keyword	Value	Description	Example
ngpus	integer	Specifies the number of gpus per node to be requested for a single set of TI simulations. Acceptable values - positive integer.	ngpus=8
wallclock	string	Specifies the wallclock on TI jobs. Acceptable values - formatted time in hours:minutes:days.	wallclock=3-00:00:00
path_to_data	string	Specifies the path to production runs. Default path is set to *system/protocol/run*	path_to_data=CDK2/unified/run
exptdatafile	string	Specifies the name of a text file containing experimental ligand binding free energies. Acceptable values - *skip*, *filename*. The text file should have two columns corresponding to *molname* (column 1) and relative ligand binding free energy (column 2).	exptdatafile=Expt.dat
bar	string	Specifies if BAR is going to be used for analysis instead of MBAR. Acceptable values - *true*, *false*.	bar=false
ccc	string	Specifies if cycle closure corrections are to be applied during analysis. Acceptable values - *true*, *false*.	ccc=true
start	float	Specifies the percentage of data to ignore from the beginning of TI production runs. Acceptable values - float numbers ranging from 0 to 100, and less than *stop*.	start=20.0
stop	float	Specifies the percentage of data to read from the start of TI production runs. Acceptable values - float numbers ranging from 0 to 100, and greater than *start*.	stop=100.00
check_convergence	string	Specifies if check of data convergence will be carried out during analysis. Acceptable values - *true*, *false*. If *check_convergence* is set to *true*, the analysis is carried out multiple times, for a range of *start* and *stop* values.	check_convergence=true

Table 2. The following table lists the keywords that are specific to `FE-MDEngine`. Brief descriptions are provided. For additional details refer to AMBER20 Reference Manual.

Flags	Entry	Description
scalpha	*real number*	The α parameter in equations 23.5 and 23.6 in AMBER20 Manual. Default value is 0
scbeta	*real number*	The parameter β in equations 23.7 in AMBER20 Manual. Default value is 12 A
gti_add_sc	*int*	Flag to control the non-bonded interactions between the common and softcore regions, and within the softcore regions. 1. AMBER18 default 2. AMBER20 default 5. AMBER20 with torsion term scaled
gti_scale_beta	*int*	Flag to control *scbeta* behavior. 0: default, original *scbeta* behavior 1. scbeta is defined as unit-less and scaled by σ_{ij}
gti_cut	*int*	0: default in versions prior to AMBER20. 1: default, the non-bond cutoff, defined by *cutoff*, will not have effect on the internal softcore non-bonded terms.
gti_cut_sc_on	*real number*	Threshold distance for switching on of softcore smoothing. If undefined, *gti_cut_sc_on* is set to *cutoff* - 2 Å. Must be smaller than the value of *gti_cut_sc_off*.
gti_cut_sc_off	*real number*	Threshold distance for switching off of softcore smoothing. Must be smaller than the value of *gti_cut_sc_off*.
gti_lam_sch	*int*	Flag for λ-scheduling. 0: Default, λ-scheduling is disabled. 1: λ-scheduling is enabled.
gti_ele_sc	*int*	Flag for the electrostatic softcore potentials. 0: Default when *gti_lam_sch*=0, smoothstep function is not utilized. 1: SSC(2) is utilized for vdW interactions.
gti_cut_sc	*int*	Flag to determine if tail smoothing will be applied to softcore potentials. 0: Default, no tail smoothing to SC 1: add smoothing to SC-vdW, beginning at *gti_cut_sc_on* and ending at *gti_cut_sc_off*, using SSC(2). 2: add smoothing to SC-vdW and SC-elec, beginning at *gti_cut_sc_on* and ending at *gti_cut_sc_off*.
gti_ele_exp	*int*	The exponent of $r_{elec,sc}(m)$ in the softcore function; the default value is 2.
gti_vdw_exp	*int*	The exponent of $r_{vdw,sc}(n)$ in the softcore function; the default value is 6.