Workflow Tools for automated Production Free Energy Simulation Setup and Analysis (*ProFESSA*) using AMBER Drug Discovery Boost - User Guide

February 8, 2022

1 Availability of the AMBER Drug Discovery Boost package

The AMBER Drug Discovery Boost package consists of the current release version of AmberTools21, which can be downloaded directly from AmberMD.org, and the specialized version of AMBER20 (the pmemd package). The AMBER20 part of AMBER Drug Discovery Boost package is only available to AMBER20 license holders. Active Amber developers can access the package directly through the Amber development GitLab repository. For other AMBER20 license holders, the primary mode for accessing the AMBER Drug Discovery Boost package is through GitLab repository set up through the laboratory for Biomolecular Simulation Research (LBSR) at Rutgers, which mirrors the AMBER Drug Discovery Boost package in the Amber development GitLab repository.

A verified AMBER20 license holder will need a GitLab account in order to access the AMBER Drug Discovery Boost GitLab repository. A new GitLab account can easily be obtained at www.GitLab.com. The user first need to send the e-mail address/username associated with the GitLab account to: Abir Ganguly at abir.ganguly@rutgers.edu or Darrin York at Darrin.York@rutgers.edu in order to be added to the AMBER Drug Discovery Boost GitLab projects. *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 / AMBER Drug Discovery Boost
Laboratory for Biomolecular Simulation Research / Alchemical_FE
Laboratory for Biomolecular Simulation Research / FE-ToolKit

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/amber-drug-discovery-boost.git
git clone git@gitlab.com:RutgersLBSR/Alchemical_fe.git
git clone git@gitlab.com:RutgersLBSR/FE-ToolKit.git

Without ssh-key setup in GitLab:
git clone https://gitlab.com/RutgersLBSR/amber-drug-discovery-boost.git
git clone https://gitlab.com/RutgersLBSR/Alchemical_fe.git
git clone https://gitlab.com/RutgersLBSR/FE-ToolKit.git

The *Alchemical_fe* folder contains the following sub-folders:

- *Documentation* - containing documentation specific to AMBER Drug Discovery Boost
- *Tutorials* - containing tutorials for setting up alchemical free energy simulations using AMBER Drug Discovery Boost
- *Examples* - containing test cases for relative binding free energy (RBFE) and relative solvation free energy (RSFE) calculations
- *bin* - containing scripts related to workflow tools to help set up alchemical free energy simulations using Amber Drug Discovery Boost

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`

2 Installation of AMBER Drug Discovery Boost

The installation process of AMBER DD Boost is identical to that of AMBER20, and detailed instructions can be found here https://ambermd.org/Installation.php

AMBER DD Boost Installation Tip

Set the following three path variables:
export AMBER_SRC=path-to-amber-drug-discovery-boost-folder
export AMBER_BUILD=path-to-where-build-directory-will-be-kept
export AMBER_INSTALLED=path-to-where-AmberDDBoost-will-be-installed

```
mkdir -p $AMBER_BUILD $AMBER_INSTALLED

cd $AMBER_BUILD
```

install serial pmemd.cuda by issuing the following command -

```
cmake -DCMAKE_INSTALL_PREFIX=$AMBER_INSTALLED \
      -DCOMPILER=GNU \
      -DMPI=FALSE \
      -DINSTALL_TESTS=TRUE
      -DCUDA=TRUE \
      -DBUILD_GUI=FALSE \
      -DBUILD_PYTHON=TRUE \
      -DDOWNLOAD_MINICONDA=TRUE \
      -DMINICONDA_USE_PY3=TRUE \
      $AMBER_SRC
```

install parallel pmemd.cuda.MPI by setting -DMPI=TRUE and re-running the above command.

3 Installation of FE-ToolKit

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

Note: FE-ToolKit installation is recommended prior the use of the Workflow Tools. Execution of *setup_fe* (see later) without FE-ToolKit installed will result in errors. *In order to use the Workflow Tools without installing FE-Toolkit, create a fake directory named "local" inside the FE-ToolKit directory.*

4 Purpose of the Workflow Tools

Workflow Tools are a set of scripts that are designed to facilitate the setup, execution, and analysis of alchemical free energy (AFE) simulations using AMBER DD Boost. Currently, the Workflow Tools can be used to perform relative binding free energy (RBFE), relative solvation free energy (RSFE), and absolute solvation free energy (RSFE) 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. The Workflow Tools can then be used to generate a hierarchy of directories containing relevant parameter, coordinate, and AMBER input files and job submission scripts. The Workflow Tools can also be used to analyze the free energy simulations using *FE-ToolKit*.

5 Initial requirements of using the Workflow Tools

Following conditions must be met before using the Workflow Tools -

I *AMBERHOME needs to be set*

II *cpptraj*(available as part of AmberTools) needs to be installed and available in $PATH

III *parmed*(available as part of AmberTools) needs to be installed and available in $PATH

- For instructions related to download and installation of AmberTools, see https://ambermd.org/AmberTools.php

IV *python3* needs to be installed and available in $PATH

V *bin* subdirectory within *Alchemical_fe* folder needs to be available in $PATH

VI FE-ToolKit needs to be installed and *local* subdirectory within *FE-ToolKit* folder must be present.

VII *RDKit* needs to be installed and python3 bindings to *RDKit* needs to be available in $PYTHONPATH

- *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`

VIII 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

6 Usage of the Workflow Tools

The *setup_fe* script represents the main executable of the Workflow Tools, and can be created by running the script *makesetup_fe.sh* located in the *bin* subdirectory of the *Alchemical_fe* folder.

./makesetup_fe.sh

makesetup_fe.sh help set the following three variables : $pathTOWRKDIR, $pathTOWFToolKit, and $pathTOFEToolKit within *setup_fe*.

I $pathTOWRKDIR should be set to the intended working directory.

II $pathTOWFToolKit should be set to the path of the *Alchemical_fe* folder.

III $pathTOFEToolKit should be set to the path of the *FE-ToolKit* folder.

By defaults, these variables are set to the path of the *current working directory*. Alternatively, these variables can be modified manually in *setup_fe*. *setup_fe* expects a inputfile named *input* in $pathTOWRKDIR

7 Input file for *setup_fe*

setup_fe requires an input file named *input* 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*. 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=11
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
ccc_ddG=true
start=0.0
stop=100.0
check_convergence=true
showallcycles=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 *AMBER DD Boost Documentation*.

8 File infrastructure created by *setup_fe*

With the *input* file present in $pathTOWRKDIR, *setup_fe* can be executed as

./setup_fe

Setup Tip: Add $pathTOWFToolKit to your $PATH variable. Keep *setup_fe* inside $pathTOWFToolKit, and manually modify the variable *"path"* in the beginning of *setup_fe* to *`pwd`*. Then go to your working directory and type

setup_fe -h

In *"setup"* mode, *setup_fe* creates a folder named *"system"*, as defined in the *input* file in $pathTOWRKDIR. The *"system"* folder will have two subdirectories, *setup* and *run*. The folder *setup* will house the various intermediate files that were generated and used in creating the final input files. The folder *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. 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, *setup_fe* creates a folder named *results* in $pathTOWRKDIR. 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.

9 An example usage of the Workflow Tools

The following section illustrates an example application of the workflow tools to setup relative binding free energy (RBFE) calculations. Herein, we assume that the initial requirements of using the Workflow Tools described earlier in section 4 have been met, and for the sake of simplicity, the Alchemical_fe repository has been cloned inside /home/user/GitLab, where the directory GitLab has been created in /home/user separately.

- Create and configure *setup_fe*

  ```
  cd /home/user/GitLab/Alchemical_fe/bin
  export pathTOWFToolKit=/home/user/GitLab/Alchemical_fe
  export pathTOFEToolKit=/home/user/GitLab/FE-ToolKit
  ./makesetup_fe.sh
  ```

  Open *setup_fe* in a text editor and manually change the "path" variable to path=`*pwd*`

- Choose a working directory

  ```
  mkdir /home/user/rbfe
  ```

- Copy the template input file *input.CDK2* from Alchemical_fe/Examples/rbfe folder to working directory and rename to *input*

```
cp /home/user/GitLab/Alchemical_fe/Examples/rbfe/input.CDK2
/home/user/rbfe/input
```

Open /home/user/rbfe/input in text editor and manually change the "path_to_input" to /home/user/GitLab/Alchemical_fe/Examples/initial

- Go the working directory and execute *setup_fe*

```
cd /home/user/rbfe
setup_fe
```

This example illustrates the setup of RBFE calculations (*ticalc*=rbfe) of two CDK2 transformations, namely 1h1q→1h1r and 1h1r→1h1s, as indicated by the keyword *translist*. The MCS mapping algorithm is used to perform the 1-to-1 atom mapping between the ligands and identify the TS and TC regions, as indicated by the keyword *mapmethod=0*. The TI simulations will consist of 25 λ windows (*nlambda=25*), with an user-defined λ schedule specified by *lams*. The file infrastructure will be generated for 3 independent trials (*ntrials=3*), the 2-state model will be used for simulation setup (*twostate=true*), and production TI runs will be carried out using the ACES approach, as specified by the combination of the use of replica exchange and *gti_add_sc=5*. Total number of replica exchanges is set to 250,0000 (*numexchgti=250000*) and exchanges will be attempted every 20 steps (*nstlimti=20*), resulting in a total simulation time of 5 ns for every λ window.
After *setup_fe* is succesfully executed, a folder "CDK2" should be created in /home/user/rbfe/ containing two subdirectories "setup" and "unified". The "setup" folder will contain the various intermediate files that were generated and used during the setup process. The "unified" folder will contain a subdirectory "run" that will have a nested directory structure. The folder "run" will contain two subdirectories "1h1q 1h1r" and "1h1r 1h1s", corresponding to the two CDK2 transformations. Each of these folders will further contain subdirectories "com" and "aq", corresponding to files related to complex and aqueous phase simulations. The "com" and "aq" folders will contain a subdirectory "inputs" containing the various AMBER input files, subdirectories "t1", "t2", "t3", that will house the simulation output files for the three independent trials, and a slurm script named *run_alltrials.slurm* that can automate the running of the entire set of equilibration simulations followed by production simulations of all three trials in the order they are intended. This slurm script is intended to be a template for running these simulations, and should be modified as needed based on the HPC architecture where the simulations will be run eventually.

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=21
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
mapnetwork	string	Specifies if network-wide consistent TS/TC regions of ligands will be generated. Acceptable values - *true*,*false*.	mapnetwork=false

Continued on next page

Table 1 – continued from previous page

Keyword	Value	Description	Example
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
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
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

Continued on next page

Table 1 – continued from previous page

Keyword	Value	Description	Example
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
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
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

Continued on next page

Table 1 – continued from previous page

Keyword	Value	Description	Example
ccc	string	Specifies if cycle closure corrections are to be applied during analysis. Acceptable values - *true*, *false*.	ccc=true
ccc_ddG	string	Specifies if cycle closures will be applied to *"complex"* - *"aqueous"* delta delta Gs instead of *"complex"* and *"aqueous"* delta Gs, independently. Acceptable values - *true*, *false*.	ccc_ddG=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
showallcycles	string	Specifies if the output should show information on all possible cycles within the given transformation network. Acceptable values - *true*, *false*.	showallcycles=true