Reference Manual for GIPS

Dr. Tobias Huefner

UC San Diego, San Diego, CA, USA
Philipps University Marburg, Marburg, Hesse, Germany

May 2, 2021

Contents

1 Getting started

1.1 Installation

We assume you have a working conda distribution on your system, e.g. miniconda or anaconda. If not, we recommend you start by setting up a conda distribution on your machine before you go on with the installation of GIPS. There are tons resources on the internet on how to set up conda. If you have conda up and running on your machine, clone the newest version of the program from GitHub and change to the main directory:

```
git clone github.com/wutobias/gips
cd gips
```

Then, build a conda environment for gips and activate it:

```
conda create --name gips -f gips.yml
conda activate gips
```

In this environment, install GIPS

```
python setup.py install
```

Now you are ready to go. The command line executable `run_gips` is now ready and the python libraries can be used from the module `gips`.

1.2 Testing

The program GIPS comes with a small set of tests that can used to check that the program is working as it should. Download ...

2 Background

This is a brief recap of the theory background of solvent functionals available in Gips. The solvation free energy score for a molecule or parts of it is computed as the sum of solvation energy and entropy:

$$\Delta G_{solv} = \Delta H_{solv} - T\Delta S_{solv} \approx \Delta E_{solv} - T\Delta S_{solv} \tag{1}$$

$$\Delta E_{solv} = \sum_{k \in R} \Delta E_{solv}(\mathbf{r}_k) \tag{2}$$

$$-T\Delta S_{solv} = \sum_{k \in R} -T\Delta S_{solv}(\mathbf{r}_k) \tag{3}$$

The solvation energy ΔE_{solv} and entropy ΔS_{solv} parts are computed for each grid voxel i using data from GIST[1, 2]:

$$\Delta E_{solv}(\mathbf{r}_k) = \Delta E_{SW}(\mathbf{r}_k) + d(E_{WW}(\mathbf{r}_k) - E_{ww}^{(bulk)}) \tag{4}$$

$$-T\Delta S_{solv}(\mathbf{r}_k) = -T\Delta S_{trans}(\mathbf{r}_k) - T\Delta S_{orient}(\mathbf{r}_k) \tag{5}$$

Here, $\Delta E_{SW}(\mathbf{r}_i)$ and $E_{WW}(\mathbf{r}_i)$ are the solute-water and water-water interaction energy at grid voxel i. The water-water interaction energy in bulk water is $E_{ww}^{(bulk)}$ and d is factor that accounts for the double-counting of water-water

interactions (usually set to 2).[1, 2] The definition of the summation region R is somewhat customary and can be a subvolume defined by a ligand molecule or the whole space. The energies and entropies from GIST can be used directly as input for eqs. 2,3, but Gips can use the data from GIST as input for solvent functionals in order to compute $\Delta E_{solv}(\mathbf{r}_i)$ and $-T\Delta S_{solv}(\mathbf{r}_i)$. In Gips, three solvent functionals are implemented[3]:

F4 Solvent Functional The solvent energy and entropy are computed with the following equations based on a Grid G:

$$\Delta E_{solv}^{(F4)}(\mathbf{r}_k) = \rho^0 \sum_k^G V_k \Delta E(\mathbf{r}_k) e_s(\mathbf{r}_k) g_s(\mathbf{r}_k) v_s(\mathbf{r}_k) \tag{6}$$

$$-T\Delta S_{solv}^{(F4)}(\mathbf{r}_k) = -\rho^0 \sum_k^G V_k T\Delta S(\mathbf{r}_k) s_s(\mathbf{r}_k) g_s(\mathbf{r}_k) v_s(\mathbf{r}_k) \tag{7}$$

Here ρ^0 and V_k are the density of pure water and the voxel volume. The energy and entropy at voxel $\Delta E(\mathbf{r}_k)$ and $T\Delta S(\mathbf{r}_k)$ are computed from GIST directly (see eqs. 4,5). The indicator functions for entropy, energy and density are defined as follows:

$$e_s(\mathbf{r}_k) = \begin{cases} 1, & \Delta E(\mathbf{r}_k) > e_{co} \\ 0 & else \end{cases} \tag{8}$$

$$g_s(\mathbf{r}_k) = \begin{cases} 1, & g(\mathbf{r}_k) > g_{co} \\ 0 & else \end{cases} \tag{9}$$

$$s_s(\mathbf{r}_k) = \begin{cases} 1, & -T\Delta S(\mathbf{r}_k) > s_{co} \\ 0 & else \end{cases} \tag{10}$$

Here the cutoff parameters e_{co}, g_{co} and s_{co} are initially unknown quantities and must be fitted with Gips. The volume indicator function indicates if a grid voxel is in the subvolume enclosed by the solvent-accessible surface of the ligand molecule or if it is outside. On a short interval between "inside" and "outside" the volume indicator function decays off smoothly:

$$v_s(\mathbf{r}_k) = \begin{cases} 1 & d(\mathbf{r}_k) < 0 \\ exp\left(\frac{d(\mathbf{r}_k)}{s}\right) & 0 < d(\mathbf{r}_k) < c \\ 0 & d(\mathbf{r}_k) > 0 \end{cases} \tag{11}$$

Here, $d(\mathbf{r}_k)$ is the distance of grid voxel k from the surface of the ligand and s is the softness of the surface. Furthermore, c defines the boundary where the indicator function is set to 0. The parameters s and c must be predefined and are currently not explicitely optimized.

F6 Solvent Functional Similar to F4 Solvent Functional (previous paragraph) except that the raw GIST data does not enter the scoring function:

$$\Delta E_{solv}^{(F6)}(\mathbf{r}_k) = E_{aff} \sum_k^G e_s(\mathbf{r}_k) g_s(\mathbf{r}_k) v_s(\mathbf{r}_k) \tag{12}$$

$$-T\Delta S_{solv}^{(F6)}(\mathbf{r}_k) = S_{aff} \sum_k^G s_s(\mathbf{r}_k) g_s(\mathbf{r}_k) v_s(\mathbf{r}_k) \tag{13}$$

Here, the parameters E_{aff} and S_{aff} are initially unknown and will be optimized during the fit. This solvent functional corresponds to the one studied in Ref. [4].

F5 Solvent Functional Similar to the F6 Solvent Functional, except that E_{aff} and S_{aff} are merged into one parameter K_{aff}:

$$\Delta E_{solv}^{(F6)}(\mathbf{r}_k) = K_{aff} \sum_k^G e_s(\mathbf{r}_k) g_s(\mathbf{r}_k) v_s(\mathbf{r}_k) \tag{14}$$

$$-T\Delta S_{solv}^{(F6)}(\mathbf{r}_k) = K_{aff} \sum_k^G s_s(\mathbf{r}_k) g_s(\mathbf{r}_k) v_s(\mathbf{r}_k) \tag{15}$$

3 Input data for Gips

3.1 GIST data

The GIST data that is used by Gips must be generated by the user. Gips itself will not generate any GIST data. It is important that the GIST grids are located in the same place where the investigated ligands are located. Gips expects the GIST data to be formated in a csv file in a format that follows the formatting generated by the GIST implementation of Cpptraj. Files that are generated by other software may or may not be read correctly.

3.2 Structural and molecule data

Currently, Gips expects that molecules are parsed as Amber restart (.rst) and Amber topology (.prmtop) files. Furthermore, it is possible to define binding pockets on a protein or select only parts of ligands in the input molecules. This is accomplished based on Amber-style selection syntax. The selection syntax is very easy to learn and is nicely described in the reference manual of the newest Amber manual.

4 Command line options for `run_gips`

The `run_gips` program must be exectuted in one of its five run modes. Each of these run modes have different options and can do different things, as will be explained in the following. A brief explanation of each option can also be obtained through the command

```
run_gips --help
```

4.1 GIST data input file

Gips uses many GIST data files from different molecules at the same time. In order to tell Gips which GIST data and molecule files to use, the user must provide a GIST data input file. An example file can be found at the end of this section. In the following all input options that go into the input file are explained. The depth of of the indentation reflects the hierarchy of the keywords in the input file. Every `init` statement creates a new layer of data and must be closed with an `end` statement at the end of the layer.

init gistdata-rec Initialze the gistdata-rec section. This section contains GIST and structural data for the receptor protein. This is the data being used for conventional displaced solvent functionals. This section can occur multiple times and may contain multiple receptors which do not necessarily be related to each other. The latter can be useful if one wants to fit solvent functionals with multiple proteins.

title This defines the name of the receptor.

init receptor Initialize a new receptor structure. The location of the GIST data files go into this section together with structural information.

gdat This is the path where the GIST data can be found. Note the formatting of this file as explained in section 3.1. This input option can occur multiple times, if the user wants to provide GIST data computed for multiple regions in the binding site. This can be beneficial if the region of interest is large and can be broken down into multiple single regions each with a single GIST calculations. Gips will merge the multiple GIST grids into one large GIST grid. Note that all GIST grids must overlap at the borders with their neighboring GIST grids. This can be conveniently accomplised with the SplitVolume.py python script.

topo Amber topology file (.prmtop) corresponding to the system that was used to compute the molecular dynamics trajectory for the GIST calculation.

strc Amber restart file (.rst) corresponding to the structure that was used to compute the molecular dynamics trajectory for the GIST calculation. This structure should be the reference structure that was used for restraining the protein during the molecular dynamics simulation in order prevent rotation and translation.

sele With this argument the user may define the binding pocket through Amber-style selection syntax. These could be atoms and residues in the binding pocket of the protein. This atom selection is mainly used to cut down the binding site and make the calculations faster.

w All GIST data in this section will be multiplied by this weighting factor. However, before this number is actually used it will be normlized with respect to all other weighting factors, in case multiple receptor sections are provided by the user. A good reason to provide weights is when each receptor section represents structures that have different probabilities, for instance computed from a clustering analysis.

end receptor: Finish this receptor section.

end gistdata-rec Finish the gistdata-rec section.

init gistdata Initialize a section containing all the other GIST data that can be used by Gips (GIST data from ligands in solution, bound protein-ligand complexes). This section can occure multiple times, for instance one for each ligand. It is recommended to only have data from one protein per gistdata section.

title The title of this gistdata section must be unique. It will be used to reference data when splitting datasets or fragmenting molecules. The user should provide a name that can easily traced back to the molecule

dg Free energy value used for fitting the solvent functionals.

dh Enthalpy value used for fitting the solvent functionals. Only used for multi-objective parameter optimization.

ds Entropy value used for fitting the solvent functionals. Only used for multi-objective parameter optimization. Note that the program assumes that the entropy is already multiplied by temperature.

init pose Initialize section with ligand poses. This section will just contain structure and topology data, no GIST data is expected here. The ligand poses listed here are only used for displaced solvent functionals that use only GIST data from the receptor structure. The ligand must be overlaid with the protein reference structures listed in the `gistdata-rec` section. Gips will not check whether they are correctly aligned in protein binding site- this is entirely up to the user. This section can occur multiple times reflecting different binding poses.

- **topo** Amber-style topolofy file (.prmtop) of the ligand molecule
- **strc** Amber-style restart file (.rst) of the ligand molecule. This file contains the structure of the ligand and will be used for the solvent functionals.
- **sele** Subset of ligand atoms that will be used for the solvent functionals.
- **ref** Identifier linking to the reference protein structure. This references to the protein structure for which the displaced solvent functional is computed for this ligand pose. The reference name follows the scheme X.Y, where X is the number of the `gistdata-rec` section and Y is the number of `receptor` section. The numbering starts at 1 in the order the `gistdata-rec` and `receptor` sections occur in the input file.

end pose Close the pose section.

init complex This section contains GIST data and structures of the protein-ligand complex.

- **gdat** Similar to the previous definition of `gdat` in the init receptor section. However, the GIST data must be computed for the bound protein-ligand complex.
- **topo** Topology in Amber-style (.prmtop) format of the bound protein-ligand complex. Since Gips only uses the coordinates of the bound ligand, it is also sufficient to just provide a topology file of the bound ligand molecule.
- **strc** Structure in Amber-style (.rst) format of the bound protein-ligand complex. This should be the coordinates of the protein-ligand complex that was used during the MD simulations. Similarly to the `topo` parameter, Gips only needs the coordinates of the bound ligand. So this can also just be the coordinates of the ligand (must match the topology file).
- **sele** Amber-style selection for the ligand molecule as it is found in the `topo` / `strc` files.
- **w** Weight factor similar as in the `init receptor` section, except that here it is used to weight the GIST data for the protein-ligand complex.

end complex End the protein-ligand complex section

init ligand This section contains GIST data and structures of the ligand molecule in solution. This is basically analogous to the `init complex` section.

- **gdat** Works exactly the same as for the protein-ligand complex.
- **topo** Works exactly the same as for the protein-ligand complex.
- **strc** Works exactly the same as for the protein-ligand complex.
- **sele** Works exactly the same as for the protein-ligand complex.
- **w** Works exactly the same as for the protein-ligand complex.

Example GIST data input file

```
init gistdata-rec
    title apo
    init receptor
        gdat apo/gist/rprod.c0.1/gist/t49/gist1/gist.dat
        gdat apo/gist/rprod.c0.1/gist/t49/gist2/gist.dat
        topo apo/gist/prep/apo.prmtop
        strc apo/gist/rprod.c0.1/gist/t0/gist.rst7
        sele :73,77,80,122,124,125,126,209,229,230,231&!(@H=|@h=)
        w    0.5843
    end receptor
    init receptor
        gdat apo/gist/rprod.c1.1/gist/t49/gist1/gist.dat
        gdat apo/gist/rprod.c1.1/gist/t49/gist2/gist.dat
        topo apo/gist/prep/apo.prmtop
        strc apo/gist/rprod.c1.1/gist/t0/gist.rst7
        sele :73,77,80,122,124,125,126,209,229,230,231&!(@H=|@h=)
        w    0.276
    end receptor
    init receptor
        gdat apo/gist/rprod.c2.1/gist/t49/gist1/gist.dat
        gdat apo/gist/rprod.c2.1/gist/t49/gist2/gist.dat
        topo apo/gist/prep/apo.prmtop
        strc apo/gist/rprod.c2.1/gist/t0/gist.rst7
        sele :73,77,80,122,124,125,126,209,229,230,231&!(@H=|@h=)
        w    0.1396
    end receptor
end gistdata-rec
init gistdata
title 3QTV
    dg    -5.732
    dh    -5.637
    ds     0.096
    init pose
        topo 3QTV/gist1/init receptorrprod/gist/t0/gist.prmtop
        strc 3QTV/gist1/rprod/gist/t0/align.gist.c0.1.rst7
        sele :1&!(@H=|@h=)
        ref 1.1
    end pose
    init pose
        topo 3QTV/gist1/rprod/gist/t0/gist.prmtop
        strc 3QTV/gist1/rprod/gist/t0/align.gist.c1.1.rst7
```

```
            sele :1&!(@H=|@h=)
            ref 1.2
            end pose
        init pose
            topo 3QTV/gist1/rprod/gist/t0/gist.prmtop
            strc 3QTV/gist1/rprod/gist/t0/align.gist.c2.1.rst7
            sele :1&!(@H=|@h=)
            ref 1.3
        end pose
        init complex
            gdat 3QTV/gist1/rprod/gist/t49/gist1/gist.dat
            gdat 3QTV/gist1/rprod/gist/t49/gist2/gist.dat
            topo 3QTV/gist1/prep/3QTV.prmtop
            strc 3QTV/gist1/rprod/gist/t0/gist.rst7
            sele :1&!(@H=|@h=)
            w    1.0
        end complex
        init ligand
            gdat 3QTV/lig/gist1/rprod/gist/t49/gist1/gist.dat
            gdat 3QTV/lig/gist1/rprod/gist/t49/gist2/gist.dat
            topo 3QTV/lig/gist1/prep/3QTV_lig.prmtop
            strc 3QTV/lig/gist1/rprod/gist/t0/gist.rst7
            sele :1&!(@H=|@h=)
            w    1.0
        end ligand
    end gistdata
```

4.2 mode: gistfit

This is the most important run mode of the program, since it carries out the fitting of the GIST-based solvent functionals. The following command line options can be used together with this mode:

-gd, --gdat This is the path to input file. See section 4.1 for details. Either this argument or `--loadlib` are required.

-ll, --loadlib This is the path to the input library file. This contains the same information as the input file, but is much faster to read. See section 4.1 for more information. Either this argument or `--gdat` are required.

-f, --fitmode The fitting mode used for the solvent functionals. This determines which data is exactly used for the solvent functionals and how many parameters are used. The allowed values are `{0,1,3,4,5,6,7}`. The solvent functionals are also named following a general nomenclature of *X/FY*, where *X* identifies the source of the GIST data and can be either P (unbound protein), L (unbound Ligand), PL (protein-ligand complex) or PL-L (difference between protein-ligand complex and unbound ligand). The *Y* (Y=4,5,6) identifies the solvent functional introduced in section 2 on page 2. During the fit, one can employ parameters that are the same for the protein, ligand and/or protein-ligand complex or different ones for each state. The former are called global parameters (e.g. $g_{CO}^{(g)}$ for a global density cutoff parameter), the latter are called state-specfic parameters (e.g. $g_{CO}^{(PL)}$ for density cutoff parameter that is only used for the protein-ligand complex (PL)).

Note that when the command line parameter `--pairs` is used, relative differences (e.g. $\Delta\Delta G$) are computed between ligand pairs. If the command line parameter `--pairs` is not used, then all quantities are computed as differences (e.g. ΔG).
This has no default value.

fitmode=0 The classical displaced solvent functional. Here we compute each ligand score from the GIST data of the unbound protein (apo). This corresponds to the P/FY (Y=4,5,6) solvent functionals in Ref. [3]. All parameters are global parameters.

fitmode=1 This is the classical solvent functional computed using multiobjective optimization. We still use the GIST data from the unbound protein, but now ΔG can be optimized together with either ΔE or ΔS (see also parameters `--decomp_E` and `--decomp_S`). This corresponds to the P/FY (Y=4,5,6) solvent functionals in Ref. [5]. All parameters are global parameters.

fitmode=2 Not functional. Depracted.

fitmode=3 This functional computes thermodynamic quantities based on the difference of the bound and unbound states $\Delta G_{solv} = \Delta G^{(PL)}_{solv} - \Delta G^{(L)}_{solv} - \Delta G^{(P)}_{solv}$. This method will not work if no GIST data for the protein-ligand complex and the unbound ligand is provided. This corresponds to the PL-L/FY (Y=4,5,6) solvent functionals in Ref. [3]. All parameters are global parameters.

fitmode=4 This is similar to `--fitmode=3`, but with multiobjective optimization (see also parameters `--decomp_E` and `--decomp_S`). All parameters are global parameters.

fitmode=5 This is similar to `--fitmode=3`, but now all density cutoff parameter, $g^{(X)}_{CO}$ are state-specific.

fitmode=6 This is similar to `--fitmode=5`, but as multiobjective optimization.

fitmode=7 All cutoff parameters $g^{(X)}_{CO}, e^{(X)}_{CO}, s^{(X)}_{CO}$ are state-specfic and optimized using multiobjective optimization (see [3]).

-b, --boundsfile Path to the file specifying the lower and upper bounds for the parameters in the solvent functionals. The solvent functional parameters are designated as *E, S, e_co, s_co, g_co* and *C* which correspond to E_{aff}, S_{aff}, e_{CO}, s_{CO}, g_{CO} and C in the related publications[5, 3]. The bounds for solvent functionals with state-specific parameters, such as $g^{(PL)}_{CO}$ and $g^{(L)}_{CO}$, are not differentiated in the bounds file. They will be both handled with the same bounds as defined through their parent single-state parameter, such as g_{CO}. This file is optional.

```
E -2 +2
S -2 +2
e_co -10 +10
s_co -10 +10
g_co +2 +10
C -10 +10
```

-c, --cut The distance away from the ligand at which the GIST boxes are cut down. This speads up the calculation, but it is important to not make this value too small. A value of *-1*, means that no cutoff is applied. Default is *-1*.

-s, --score This selects the solvent functional ("score") introduced in section 2 on page 2. The accepted values are 4,5 or 6. This has no default value.

-sc, --scaling Energy scaling factor. See parameter d in eq. 4 on page 2. Default is 2.

-dE, --decomp_E Use free energy and energy as objectives in multiobjective optimization.

-dS, --decomp_S Use free energy and entropy as objectives in multiobjective optimization.

-r, --radiusadd Add this value (in Angstrom) to the atomic radii for the calculation of the molecular volume of the ligand. This expects two values, the first one used for the displaced solvent functionals (fitmodes 0,1), the second one is used for solvent functionals operating on GIST data from ligand or protein-ligand complex (i.e. fitmodes >1). Default is `0 3`.

-softness, --softness Softness parameter s for the molecular surface calculation, see eq. 11. Default is 1.

-softcut, --softcut Surface cutoff paramete c for the molecular surface calculation, see eq. 11. Default is 2.

-opt, --optimizer Three different choices possible: `[evolution, basinhopping, brute]`. All optimization algorithms are implemented through the python package PyGMO. The default is `evolution`.

evolution Evolutionary algorithm. For single-objective optimization this uses the self-adaptive differential evolution algorithm and NSGAII for multiobjective optimiztion.

basinhopping This employes the basinhopping algorithm. Only works for single objective optimization.

brute Bruteforce scanning algorithm. The `--niter` parameter determines how many minimization steps are performed after each scanning step (=0, no minimization).

-ni, --niter Number MC attempts in basinhopping optimization before termination or number of generations in evolution optimization. Default is 500.

-pop, --popsize Population size during differental evolution optimization. Default is 50.

-step, --stepsize Step size used in brute force optimization. Default is 0.05.

-gr, --gradient Use analytical gradient. Otherwise approximate gradient using forward finite differences. The analytical gradients are much faster.

-bo, --boundary Use quadratic penalty term (i.e. $k(x-x_0)^2$, with k being a force constant and x_0 being a boundary value defined in `--boundsfile`) for the treatment of the boundaries of the parameters. Otherwise let the optimization algorithm handle the boundaries.

-k, --kforce Use this "force constant" for the quadratic penalty term (see `--boundary`). Default is 100.

-p, --pairs If activated, compute ligand pairs and not single ligands. The computed quantities will be relative differences (e.g. $\Delta\Delta G$). Otherwise, everything will computed as absolute binding free energies.

-pf, --pairfile File that holds all the pairs that the user wants to include in the analysis. Note, only considered when pairwise treatment is activated (see `--pairs`). A pairfile has three columns, and as many rows as pairs. The first two columns are the names of the ligands as they occur in the input file (see command line option `--gdat`), the last column is a float number that can be used as an additional filter (e.g. Tanimoto similarity of the ligands). Example:

```
2ZDV 2ZC9 0.873772791024
2ZF0 2ZC9 0.898697539797
2ZF0 2ZDV 0.888571428571
2ZFF 2ZC9 0.93665158371
2ZFF 2ZDV 0.922734026746
2ZFF 2ZF0 0.956856702619
2ZFP 2ZC9 0.861209964413
2ZFP 2ZDV 0.765494137353
```

```
2ZFP 2ZF0 0.782161234991
2ZFP 2ZFF 0.811387900356
2ZGX 2ZDA 0.864436619718
```

-pc, --paircut Cutoff valued used to filter the ligand pairs provided in the pairfile (see `--pairfile`). Only pairs with a value greater than this cutoff value will be considered in the analysis. Default is 0.0 .

-ks, --ksplit During the fitting of solvent functionals, Gips uses cross-validation. With this command line argument, the dataset is randomly split into k subsets. The training is carried on the $k-1$ datasets and testing is reported for data set k. Note that then a split file is provided with `--ksplitfile`, then `--ksplit` is the index of the subset that is used as test set. The Default is 5.

-ksf, --ksplitfile This is a split file and contains predefined splits for cross validation. The last column in the file is the k, indicating which split this pair belongs to. If the user wants to fit pairs (with command line argument `--pairs`), this file must have two leading columns (one for each ligand in the pair). If the fit is carrier out for single ligands, then their is only a single leading column. No default. When providing the split file in `--mode=gistfit`, no pair (`--pairfile`) or exclude (`--exclude`) file will be read in, since all information for how to select datapoints or pairs are already included in the split file. No default. Example:

```
2ZDV 2ZC9 3
2ZF0 2ZC9 0
2ZF0 2ZDV 4
2ZFF 2ZC9 1
2ZFF 2ZDV 3
2ZFF 2ZF0 4
2ZFP 2ZC9 2
2ZFP 2ZDV 2
2ZFP 2ZF0 2
2ZFP 2ZFF 2
2ZGX 2ZDA 0
2ZO3 2ZDA 1
2ZO3 2ZGX 3
```

-sh, --shuffle If this command line argument is given, then the dependent variables (i.e. ΔG, etc ...) are shuffled before fitting (*y-scrambling*).

-ex, --exclude With this argument, the user can provide a file containing names of datapoints that will be excluded from the solvent functional calculation. The names of the datapoint can be provided one per line or seperated by whitespaces. No default.

-pre, --prefix Prefix used for any output files written to disk.

-v, --verbose Run the program with verbose output.

4.3 mode: buildlib

This mode can be used to pre-process GIST input data as provided with the `--gdat` command line argument and save it to a file on disk. This is not strictly necessary, but can save some time since the preprocessing and loading of GIST data can take some time. The command line arguments for this mode are the following:

-gd, --gdat This is the input file as described in section 4.1 on page 5.

-sl, --savelib File name of the output file. If the filename ends in ".gz", the file will be compressed using gzip.

-c, --cut This is the cut distance as described in section 4.1 on page 5.

4.4 mode: split

With this mode, the dataset can be split into random subsets in order to perform cross validation. Although these datasets can also be generated on-the-fly when running in mode `gistfit`, it might be beneficial to always have the same splittings for cross validation. The output files will be named `<--prefix>`_pair-splits.dat if the splits are generated for ligand pairs or `<--prefix>`_splits.dat if single ligand splits are being generated. Ligand pairs will be generated if a pairfile is provided.

-pf, --pairfile See section 4.1 on page 5. The user must provide either this file or the include file (`--include`) or both.

-ex, --exclude See section 4.1 on page 5.

-in, --include Provide datapoints that will be included in the data set splitting. The user must provide either this file or the pair file (`--pairfile`) or both.

-ks, --ksplit See section 4.1 on page 5.

-pc, --paircut See section 4.1 on page 5.

-pre, --prefix See section 4.1 on page 5.

-r, --radiusadd

-softness, --softness

-softcut, --softcut

4.5 mode: mapout

This mode computes hydration sites clustered on the GIST density grids in the region defined by the volume indicator function eq. 11. For each hydration site, the solvent functionals are evaluated at the position of the hydration sites. The clustering procedure works as follows:

1. Initialize list with hydration site positions *hs_xyz*.
2. Find the largest density value on the grid and append its position to *hs_xyz*.
3. Set all grid points within 1.4 Å on the density grid to 0.
4. Repeat steps 2. and 3. until no grid points are left with a value greater than the minimum value on the density grid.
5. For each hydration site in *hs_xyz*, compute the solvent functional and write out pdb files for density, energy, entropy and free energy hydration sites.

Many command line arguments from command line mode `gistfit` can be used here as well:

-parms, --parmsfile This file contains the parameters used for the solvent functional in this analysis. The parameter file must match the functional form of the solvent functional, i.e. `--fitmode`, `--score` and `--pairs` must be exactly as used for the training of the solvent functional. Gips expects the parameter file to be formatted as the *parms.out* output file from running Gips in run mode `gistfit`. The user can also just use a *parms.out* output here. If multiple solutions are provided in the parameter file, Gips will figure out on its own which parameter set is the best one. For example:

```
### Step e_co[kcal/mol] s_co[kcal/mol] g_co(Rec)[1/Ang^3] g_co(Cplx)[1/Ang^3] g_co(Lig)[1/Ang^3] C_E[k
0  9.598  7.876  9.550  2.539 -1.030  0.834 1061.163 64306617.211 14870686.069  0.268  0.204 1072.871
0  9.598  7.876  9.550  2.539 -1.030  0.834 1061.163 5214.675 1513.840  0.000  0.009 1069.145 1070.451
1  9.597  7.876  9.550  2.539 -1.030  0.834 1061.163 64306617.211 14870686.069  0.268  0.204 1072.871
1  9.597  7.876  9.550  2.539 -1.030  0.834 1061.163 5214.675 1513.840  0.000  0.009 1069.145 1070.451
2  9.598  7.876  9.550  2.539 -1.030  0.834 1061.163 64306617.211 14870686.069  0.268  0.204 1072.871
2  9.598  7.876  9.550  2.539 -1.030  0.834 1061.163 5214.675 1513.840  0.000  0.009 1069.145 1070.451
3  9.598  7.876  9.550  2.539 -1.030  0.834 1061.163 64306617.211 14870686.069  0.268  0.204 1072.871
3  9.598  7.876  9.550  2.539 -1.030  0.834 1061.163 5214.675 1513.840  0.000  0.009 1069.145 1070.451
```

-gd, --gdat See section 4.1 on page 5.

-ll, --loadlib See section 4.1 on page 5.

-f, --fitmode See section 4.1 on page 5.

-s, --score See section 4.1 on page 5.

-p, --pairs See section 4.1 on page 5.

-pf, --pairfile See section 4.1 on page 5.

-ex, --exclude See section 4.1 on page 5.

-pre, --prefix See section 4.1 on page 5.

-sc, --scaling See section 4.1 on page 5.

-pre, --prefix See section 4.1 on page 5.

-v, --verbose See section 4.1 on page 5.

4.6 mode: decompose

This run mode decomposes the ligand molecules into substructures and evaluates the solvent functionals on those.[6] Three different ways of defining substructures are supported:

1. Provide the substructures directly and Gips will search for them in the dataset.
2. Provide a substructure decomposition scheme for each ligand.
3. Determine substructures automatically through the *Recap* algorithm.[7]

Since this run mode is a way to spatially decompose solvent functionals, many options from the run modes `mapout` and `gistfit` can be applied here as well. The command line options for this run mode are as follows:

-frag, --frag_file With this file the user can directly define substructures (a.k.a. fragments). Gips will then search for these structures in the dataset and perform the analysis. The fragfile contains two columns and as many rows as substructures. The first column is a unique integer identifier for each fragment, the second column is a smiles string of the fragment. If neither `--frag_file`, nor `--map_file` are provided by the user, the program will figure the substructures on its own using the Recap algorithm. There is no default.

```
0 O=C[C@@H]1CCCCN1
1 [NH]CCCNC(N)=[NH2+]
2 O=C[C@@H](CC1CCCCC1)[NH2+]CC(=O)[O-]
3 [NH3+][C@@H](C=O)Cc1ccccc1
4 O=C[C@@H]1CCCN1
5 [NH]Cc1cccc(Cl)c1
6 [NH]Cc1ccc(C(N)=[NH2+])cc1
7 [NH]Cc1cccc(F)c1
8 Cc1cccc(C[NH])c1
9 [NH]Cc1ccccc1
```

-map, --map_file With this option, the user can provide a detailed scheme for the decomposition of the molecule. This file must have two leading columns containing the name of the ligand in the first column and integer identifiers for the substrucures of the ligand in the second column. If a ligand has no substructures to be matched, then the second column has just the entry *-1*. The following columns will have the comma seperated atomic indices for each substructure as it is found in the ligand. Each substructure corresponds to one column. There is no default.

```
1K21 0,1,2 0,1,22,23,25,28,31,34 2,4,7,10,13,15,16,19 37,38,39,41,44,47,49,52,55,58,61,64,67,68,69
2ZC9 3,4,5 0,1,2,3,4,5,6,7,8,9,10 11,12,13,14,15,16,17 18,19,20,21,22,23,24,25,26
2ZDA 3,4,6 0,1,2,3,4,5,6,7,8,9,10 11,12,13,14,15,16,17 18,19,20,21,22,23,24,25,26,27,28
2ZDV 3,4,7 0,4,6,7,8,11,12,14,16,18,20 22,23,25,26,27,30,33 36,38,39,42,44,46,47,48,50
2ZF0 3,4,8 0,1,2,3,4,5,6,7,8,9,10 11,12,13,14,15,16,17 18,19,20,21,22,23,24,25,26
2ZFF 3,4,9 0,1,2,3,4,5,6,7,8,9,10 11,12,13,14,15,16,17 18,19,20,21,22,23,24,25
2ZFP 10,4,5 0,4,6,7,8,11 15,16,18,19,20,23,26 29,31,34,35,37,39,41,42,44
2ZGX 10,4,6 0,4,6,7,8,11 15,16,18,19,20,23,26 29,31,34,35,37,39,40,41,44,47,49
2ZO3 11,4,6 0,2,4,6,8,10,11,13,15,16,17,21,22,24,26,28,30 32,33,35,36,37,40,43 46,48,51,52,54,56,57,58
3BIU 12,4,6 0,1,2,5,8,10,13,16,19 22,23,25,26,27,30,33 36,38,41,42,44,46,47,48,51,53,55
```

-parms, --parmsfile See section 4.5 on page 12.

-gd, --gdat See section 4.1 on page 5.

-ll, --loadlib See section 4.1 on page 5.

-f, --fitmode See section 4.1 on page 5.

-s, --score See section 4.1 on page 5.

-p, --pairs See section 4.1 on page 5.

-pf, --pairfile See section 4.1 on page 5.

-ex, --exclude See section 4.1 on page 5. This command line option is not recommended for this run mode since it may mess up the internal substructure library. It usually doesn't hurt to not exclude ligands/ligand pairs from this analysis, since it will only generate more data.

-pre, --prefix See section 4.1 on page 5.

-sc, --scaling See section 4.1 on page 5.

-pre, --prefix See section 4.1 on page 5.

-v, --verbose See section 4.1 on page 5.

5 Understanding the Output

5.1 Output for run mode gistfit

This run mode generates three output files (with *X* being the subset of dataset used for testing in the solvent functional as defined through the command line argument `--ksplit`). For multiobjective optimization, each file contains two rows per solution. The first row for each solution corresponds to the free energy objective and the second row to the energy or entropy objective.

k*X*.prediction.out Contains the predicted values based on the solvent functional. Each column is a ligand or ligand pair and each row is a solution to the solvent functional. The row starting with *-1*, are the reference (e.g. experimental) values. The row starting with *#Selection* indicates whether a data point is from the training (A) or test (B) set. After the row starting with *### Best result (A)* the solutions are sorted from best to worst. For multiobjective optimization, the number after the forward slash / indicates the Pareto front of this solution.

k*X*.parms.out Contains the parameters for the solvent functional. The formatting is similar to the file k*X*.prediction.out

k*X*.diff.out Contains the difference between computed and reference (e.g. experimental) values as provided in the input file. The formatting is similar to the file k*X*.prediction.out

5.2 Output for run mode mapout

This run mode generates a pdb file with hydration sites for each ligand. Since a given ligand can be combined with multiple GIST data sets, this will write out one pdb file for each GIST data set.

5.3 Output for run mode decomposition

This run mode generates two output files:

decomposition.molecules.out For each ligand/ligand pair, contains the substructures and their thermodynamics (ΔG, ΔS, ΔE). Depending on solvent functional, the output is further decomposed into contributions from ligand, apo protein and protein-ligand complex.

decomposition.fragments.out For each molecular substructure, provide statistics about its min, max and average values and in which ligand they occur.

6 Remarks

6.1 Some notes on how to generate GIST data using molecular dynamics

One very important thing to consider when generating data for GIST is that rotation and translation of the solute system (e.g. protein) must be suppressed during the MD simulation. This can be conveniently achieved through positional

restraints. The way these restraints are activated is different for each MD code. For Amber, a good overview on how to carry out MD simulations for GIST analysis can be found in this tutorial on the Amber website.

Depending on the specific problem of interest, it can be a good idea to first run vanilla MD simulations and then cluster the solute coordinates. Then run restraint MD simulations on each of the cluster representatives and carry out the GIST analysis on those. This can be helpful, if the solute has multiple conformations of interest and a only single conformation would not be a good approximation.

7 Examples

Examples can found on GitHub.

References

[1] Nguyen, C. N.; Young, T. K.; Gilson, M. K. Grid inhomogeneous solvation theory: Hydration structure and thermodynamics of the miniature receptor cucurbit[7]uril. *Journal of Chemical Physics* **2012**, *137*, 1–17.

[2] Nguyen, C.; Gilson, M. K.; Young, T. Structure and Thermodynamics of Molecular Hydration via Grid Inhomogeneous Solvation Theory. *arXiv* **2011**, 16.

[3] Hüfner-Wulsdorf, T.; Klebe, G. Protein-Ligand Complex Solvation Thermodynamics: Development, Parameterization, and Testing of GIST-Based Solvent Functionals. *Journal of Chemical Information and Modeling* **2020**, *60*, 1409–1423.

[4] Nguyen, C. N.; Cruz, A.; Gilson, M. K.; Kurtzman, T. Thermodynamics of water in an enzyme active site: Grid-based hydration analysis of coagulation factor xa. *Journal of Chemical Theory and Computation* **2014**, *10*, 2769–2780.

[5] Hüfner-Wulsdorf, T.; Klebe, G. Advancing GIST-Based Solvent Functionals through Multiobjective Optimization of Solvent Enthalpy and Entropy Scoring Terms. *Journal of Chemical Information and Modeling* **2020**, *60*, 6654–6665.

[6] Hüfner-Wulsdorf, T.; Klebe, G. Mapping Water Thermodynamics on Drug Candidates via Molecular Building Blocks: a Strategy to Improve Ligand Design and Rationalize SAR. *J. Med. Chem.* **2021**, *64*, 4662–4676.

[7] Lewell, X. Q.; Judd, D. B.; Watson, S. P.; Hann, M. M. RECAP - Retrosynthetic Combinatorial Analysis Procedure: A powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry. *Journal of Chemical Information and Computer Sciences* **1998**, *38*, 511–522.