This is the title [Article v1.3]

Firstname Middlename Surname[1*], **Firstname Middlename Familyname**[1,2†§], **Firstname Initials Surname**[2†¶], **Firstname Surname**[2*]

[1]Institution 1; [2]Institution 2

This LiveCoMS document is maintained online on GitHub at https://github.com/myaccount/homegithubrepository; to provide feedback, suggestions, or help improve it, please visit the GitHub repository and participate via the issue tracker.

This version dated June 15, 2022

Abstract Grid inhomogeneous solvation theory (GIST) is a method to compute the free energy of hydration of a compound on a 3-dimensional grid. The high spatial resolution of the GIST output, as well as the decomposition into energy and entropy contributions, allow for highly detailed analyses on both proteins and small molecules. However, this versatility also comes with a higher entry barrier for new users.

In this tutorial, we aim to guide the reader through the most common steps involved in a GIST analysis at the example of the streptavidin-biotin complex. Furthermore, we discuss the theory of GIST with a focus on practical aspects, and show several pitfalls and technical difficulties that may occur during a GIST study. We assume familiarity with molecular dynamics (MD) simulations as well as the AmberTools package.

***For correspondence:**
email1@example.com (FMS); email2@example.com (FS)

†These authors contributed equally to this work
‡These authors also contributed equally to this work

Present address: §Department, Institute, Country; ¶Department, Institute, Country

1 Introduction

(vah) Sources and better description of all these methods, more literature research, better flow

IMO, our article looks too dry. We should have something nice (e.g., an illustration of the GIST method) on the 1st or 2nd page -fwa.

Solvation thermodynamics govern any process involving solutes in a solvent. Especially the interaction with water is of utmost importance, as many biological processes occur in a aquaeous environment. The free energy of solvation thereby strongly influences real-world problems of interest such as hydrophobic effects, the binding of ligands to a biomolecule, as well as the dynamics and folding of proteins.

The computational calculation of thermodynamical solvation properties is therefore of particular interest and has been tackled by a wide array of methods. Quantum mechanical methods provide the most rigorous description of the underlying potential energy surface (PES) but suffer from high computational cost, allowing only the treatment of small systems and few solvent molecules.

Molecular mechanics based approaches describe the underlying PES through empirically fit force-fields, which reduce the computational demand of these calculations tremendously. This allows for the treatment of large biological systems such as proteins in aqueous solution. Implicit solvent methods such as MM/PBSA or MM/GBSA [1, 2] can quickly provide results for a large number of structures, but yield inaccurate results when compared to more rigorous explicit solvent methods [3]. On the other hand, the free energy of solvation can be calculated in a statistically rigorous way using explicit solvent simulations combined with alchemical methods such as free energy perturbation (FEP) [4] or thermodynamic integration(TI) [5]. However, they are unable to provide a spatial interpretation of the results. Furthermore, splitting the resulting free energy into contributions for the enthalpy and entropy is challenging [6].

Methods derived from statistical mechanics bridge this gap by providing three-dimensional resolution. Two major groups of methods that fall into this group are those based on inhomogeneous solvation theory (IST) [7] and those based on the Ornstein–Zernike (OZ) equation [8]. The most important example of OZ-based methods is the reference interaction site model (RISM) [9] as well as its extension to three dimensions (3D-RISM) [10]. Those methods compute the solvent distributions by self-consistent solution of the OZ integral equation to predict solvation thermodynamics.

On the other hand, methods based on IST generally use a molecular solvent distribution obtained from an MD simulation to compute the free energy of hydration from its enthalpic and entropic contributions. Examples of those methods include grid inhomogeneous solvation theory (GIST) [11, 12], WaterMap [13, 14], SSTMap [15], and STOW [16]. The main limitation of those methods is that they rely on an infinite expansion of the entropy in terms of the density distribution, which is truncated after the first or, sometimes, the second [17, 18] term. Nevertheless, they have been applied on a wide variety of problems involving both small molecules and biological systems.

In this tutorial, GIST is applied to calculate solution thermodynamics properties of interest.

GIST is based on IST, solving the integral equations on a grid. This approach is more general than other IST-based methods because it accounts for all solvent regions rather than focusing on high-occupancy solvation sites. By simulating the dynamics of a solvent around the solute, GIST calculates thermodynamic properties for a standard solvation process as described by Ben-Naim [19]. It allows for a full localization of entropic and enthalpic contributions on the grid around the solute. It has been shown that GIST can produce comparable accuracy to TI by applying a linear correction factor (0.6 for water) to the first-order entropy as follows [18, 20]:

$$S^{total} = S^{1st\text{-}order} \times f \tag{1}$$

Recent work has improved the calculation speed tremendously, allowing for the treatment of large biological systems such as proteases or antibodies. Furthermore, an implementation based on the particle mesh Ewald (PME) [21] method was presented, which improves the consistency between GIST and popular molecular dynamics engines. Yet further changes allow the inclusion of other rigid solvents than water or allow for the treatment of solvent mixtures and ions in solution.

While previous tutorials gave insightful pointers as to how to setup and run GIST calculations, the post-processing of such calculations was often treated less rigorously. While building on these tutorials, an updated guide to calculating solvation properties using GIST is therefore presented here to address the recent improvements to the method.

1.1 Scope

This tutorial provides an example of a GIST workflow that can be easily adapted towards different systems. In addition to a short introduction of the method and the theory behind it, we present an overview of the various different flavors of GIST methods.

Furthermore, a GIST study of the streptavidin-biotin complex is presented, which reflects on many common aspects of GIST studies. Using this test system, we present solvation analysis of a small molecule (biotin) as well as a biomolecule (streptavidin). We show how to interpret the three-dimensional contributions of solvation free energy in a binding pocket and around the ligand.

Additionally, we compute the binding contribution of the free energy of solvation, which requires accurate post-processing of the GIST outputs to avoid unfavorable summation of bias in the GIST calculation. We believe that this example study covers a wide range of applications of the GIST method.

Furthermore, we discuss several technical aspects and typical pitfalls of GIST analyses, such as the normalization of voxel values and how to deal with double counting of energy contributions. Additionally, we provide a python library to unify the analysis of GIST data produced by various versions of GIST and make the post-processing of GIST results more accessible.

After completing this tutorial, we expect the reader to be able to run their own GIST study.

2 Prerequisites

(vah) Improve flow and logical progression of prerquisites.

2.1 Background knowledge

This tutorial is aimed at users with a solid knowledge of molecular dynamics (MD) simulations. The user should be able to run MD simulations and work with their output data. Previous experience of GIST is not necessary. However, we only provide a short explanation of the theory behind GIST and point the reader towards the primary sources for more detailed information.

The presented analyses are run in Python. While no programming experience should be necessary to follow the tutorial, a basic understanding of Python is advantageous to adjust the code towards different requirements and use-cases.

We provide a Jupyter notebook which contains the presented analyses. Therefore, an understanding of Jupyter notebooks is beneficial to reproduce the analyses.

2.2 Software/system requirements

The main implementation of GIST is part of the MD analysis software `cpptraj` as part of AmberTools [22]. A recent version of `cpptraj` should be used, such as the one in AmberTools22 or an installation from GitHub (https://github.com/Amber-MD/cpptraj). The closed-source Amber simulation engine is used in this tutorial to run the necessary MD simulations.

If no Amber installation is available, the MD can be performed other simulation engines compatible with `cpptraj`. If using GROMACS, we propose to prepare the structures and topologies with AmberTools and converting them using `acpype.py`. In that way, a full Amber topology is available for GIST. (vah) is it necessary to mention GROMACS specifically? I would remove the previous two lines Also note that compressed trajectory formats such as `xtc` might bias the entropy calculation.

Furthermore, a recent Python version (>3.6) should be available with the following packages.

- `gisttools` (https://github.com/liedllab/gisttools)
- `mdtraj` (https://www.mdtraj.org/)

We also show examples using the `gistpp` program, which can be obtained from https://github.com/KurtzmanLab/Gist-Post-Processing (code and documentation).

If the reader prefers to skip the MD and GIST calculation, we also provide the GIST output files with this tutorial, such that the post-processing can be done without any expensive calculations. We also provide a Jupyter notebook [23, 24] with the presented analyses. We recommend JupyterLab to work with the notebook. Jupyter is available at https://jupyter.org/.

For the visualization of three-dimensional contributions we require molecular visualization software that can handle `OpenDX` files. We recommend using a current version of `PyMol`[25] or `VMD`[26].

3 Theory

GIST is an implementation of Inhomogeneous Solvation Theory (IST) [7] that discretizes the free energy of solvation ΔG_{solv} on a three-dimensional grid. It was first devised by Nguyen et al. [11] to overcome the limitation of IST which was commonly used for regions of high water occupancy. Its implementation in `cpptraj` was thoroughly described in [12]. Here, we only provide a short overview of the theory behind GIST. For more detailed information, we recommend one of the more recent publications on developments in GIST. [20, 27]

3.1 Solvation Entropy

The solvation entropy of a system is totality of the contribution of the solute-water entropy and the water-water entropy.

$$\Delta S_{solv} = \Delta S_{sw} + \Delta S_{ww} \tag{2}$$

Here the solvation entropy is approximated from the contribution of solute-water interaction. **STILLNEEDS REVISON + ADD CITATIONS** The quantity $k_b$represents the Boltzman constant, ρ^o is the bulk number density, $g_{sw}(\mathbf{r},\omega)$ is the solute-water pair correlation function in the solute's frame of reference, $\mathbf{r}$ is the coordinates of the oxygen atom of the water molecule, ω represents the Euler angles, and $\frac{1}{8\pi^2}$ is the volume of the orientational space, which acts as a normalization constant. In bulk and in regions where the orientational entropy is uniform, the quantity $g_{sw}(\mathbf{r},\omega)$ is unity which leads to the first-order solvation entropy of bulk to be zero. Also, this quantity reaches unity as the distance from the solute molecule increases which leads for the calculation of the solvation entropy to be an approximation of the integral around the solute atom.

$$\Delta S_{solv} \approx \Delta S_{sw} \equiv -R\frac{\rho^0}{8\pi^2}\int g_{sw}(\mathbf{r},\omega)\ln g_{sw}(\mathbf{r},\omega)d\mathbf{r}d\omega \tag{3}$$

We are inconsistent regarding R/kB; vah: Fixed. is it - tho? Papers seem to disagre

The solvation entropy is further broken down in terms of the contribution from translational and orientational terms.

$$\Delta S_{solv} = \Delta S_{trans} + \Delta S_{orient} \tag{4}$$

3.2 Entropy Calculations in Cpptraj

In `cpptraj` the calculation of solvation entropy is handled into two methods.

The first method assumes that the position (r) is independent of the orientation (ω). This approximation allows for the splitting of the contributions of the translational and the orientational entropy. The nearest neighbor (NN) approach is used to evaluate each expression where N_k is the number of water molecules found in voxel k, γ is the Euler's constant that accounts for the bias in naive entropy estimator, and g_{NN} is the nearest neighbor estimate. In the nearest neighbor estimate, a voxel is considered to be a neighboring voxel when it shares any vertices of the voxel of interest. vah: Is gamma divided by Nk or not? The formula in Franz' recent paper indicates otherwise.

$$g_{vox}(\mathbf{r},\omega) \approx g_{vox}(\mathbf{r})g_{vox}\omega) \tag{5}$$

$$S_k^{trans} \approx \frac{-R}{N_k}\left(\sum_{i=1}^{N_k}\ln g_{NN,i}(\mathbf{r}) + \gamma\right) \tag{6}$$

$$S_k^{orient} \approx \frac{-R}{N_k}\left(\sum_{i=1}^{N_k} \ln g_{NN,i}(\omega) + \gamma\right) \tag{7}$$

The second method directly calculates the solvation entropy by evaluating the six-dimensional integral (3 for position and 3 for orientation) using the nearest-neighbor approach. Unlike the first method, the nearest neighbor estimate for the sixth-order entropy considers all the water found within the same and neighboring voxels.

$$S_k \approx \frac{-R}{N_k}\left(\sum_{i=1}^{N_k} \ln g_{NN,i}(\mathbf{r},\omega) + \gamma\right) \tag{8}$$

3.3 Solvation Energy

The solvation energy is readily calculated by summing up the contributions of the water-water interaction and the water-solute interaction.

$$\Delta E_{solv} = \Delta E_{sw} + \Delta E_{ww} \tag{9}$$

The solute-solvent energy E_{sw} can be expressed in terms of the solvent density and the potential $U_{sw}(\mathbf{r},\omega)$ induced by the solute. In practice, it is evaluated as the expectation value $\langle\cdot\rangle$ of the pairwise force field energy U_{ij} between all solvent molecules in voxel k and all solute atoms.

$$\begin{aligned}\Delta E_{sw}(\mathbf{r}) &\equiv \frac{1}{8\pi^2}\int g_{sw}(\omega|\mathbf{r})\, U_{sw}(\mathbf{r},\omega)\, d\omega \\ \Delta E_{sw,k} &= \left\langle \sum_i^{solvent,k} \sum_j^{solute} U_{ij} \right\rangle\end{aligned} \tag{10}$$

To localize E_{sw} on the three-dimensional grid, every energy term is assigned to the voxel k holding the solvent, and the expectation value in Equation 10 is computed per voxel. Similar to the entropy integrals, the solute-water solvation energy integral also decays with increasing distance from the solute. Hence, the solvation energy can be approximated by local spatial integrals.

The solvent-solvent energy E_{ww} is computed in a similar manner. Again, it can be expressed in terms of density functions, but is practically computed as a sum over solvent-solvent interactions where one solvent is in voxel k. In contrast to E_{sw}, E_{ww} does not tend to zero in bulk. Therefore, a reference energy corresponding to the average energy of a water molecule in bulk needs to be subtracted.

$$\begin{aligned}\Delta E_{ww}(\mathbf{r}) &\equiv \left(\frac{1}{8\pi^2}\right)^2 \rho^o \int g_{sw}(\omega|\mathbf{r}) \\ &\times \left[g_{sw}(\mathbf{r}',\omega') - g_{ww}^o(\mathbf{r},\omega,\mathbf{r}',\omega')\right] \\ &\times U_{ww}(\mathbf{r},\omega,\mathbf{r}',\omega')\, d\omega d\mathbf{r}' d\omega' \\ \Delta E_{ww,k} &= \frac{1}{N_k}\left\langle \sum_i^{solvent,k} \sum_{j\neq i}^{solvent} U_{ij}\right\rangle - \left\langle E_{ww}^{bulk}\right\rangle\end{aligned} \tag{11}$$

When computing the total E_{ww} of the system, double-counting of interactions must be avoided. The total solvent-solvent energy of the system is as follows.

$$\Delta E_{ww} = \int \frac{\Delta E_{ww}(\mathbf{r})}{2} d\mathbf{r} \tag{12}$$

When water is replaced from a small region R of interest, such as a single water molecule, almost all interactions are with water molecules outside of this region. Therefore, there is no double counting, and the full solvent-solvent energy should be used.

$$\Delta E_{ww,R} = \int_R E_{ww}(\mathbf{r}) d\mathbf{r} \tag{13}$$

When the region of interest R comprises more than one molecule, interactions within this region are double-counted while interactions to the outside are not. To take this into account, the solvent-solvent energy between each pair of voxels $E_{ww,kl}$ is needed. While this is supported by the standard GIST implementation, it is rarely done due to the large size of the $E_{ww,kl}$ matrix.

We note that replacement of a small or medium solvent region would lead to reorganization of the surrounding solvent. However, this depends on the local environment and on *what* the molecule would be replaced by. Therefore, the effect of reorganization can not be directly included in the GIST method, but must be judged for each case individually. This is not a problem when computing the total free energy of solvation ΔG_{solv}, since no additional boundary is created. Do we need this paragraph?

3.4 PME implementation

In the original version of GIST, the energies are calculated based on the the minimum image convention. In PME-GIST, the electrostatics are calculated using the particle mesh Ewald (PME) method, which yields energies that are highly consistent with the treatment in the Amber MD engine [20]. The electrostatic and Lennard-Jones parts E_{elec} and E_{lj} are computed separately. E_{LJ} seems more intuitive to me, need to check what the GIST literature is using -vah. They use E_{lj}, but LJ in the text -fwa.

$$E_{\text{total}} = E_{\text{elec}} + E_{\text{lj}} \tag{14}$$

In PME-GIST, the system is treated as periodic, and the electrostatic energy is split into a short-range term E_{dir}, which is calculated in direct space using a distance cutoff, and a long-range term E_{rec}, which is calculated in reciprocal space. Additionally, there is a correction term E_{corr}, which corrects for the self-interaction of each solvent molecule in the reciprocal term.

$$E_{elec} = E_{dir} + E_{rec} + E_{corr} \tag{15}$$

The Lennard-Jones contribution is computed in the direct space, using a distance cutoff. The short-range contribution is the LJ energy within the set cutoff and the long-range correction term accounts for the contributions above this cutoff.

$$E_{lj} = E_{lj,\,short} + E_{corr} \tag{16}$$

4 Tutorial

Here, we aim to guide the reader through a GIST analysis of the streptavidin-biotin complex. The goal is to

1. compute the binding free energy of the complex from the individual energy contributions combined with the contribution of hydration computed using GIST.
2. visualize local contributions to ΔG_{solv} in the streptavidin binding pocket as well as around biotin.

4.1 Streptavidin/Biotin

4.2 Tutorial data

We will use the 1STP crystal structure to start our calculations. For the sake of simplicity, you can download prepared and solvated Amber topologies and structures based on 1STP by running

```
git clone git@github.com:liedllab/gist-tutorial.git
```

This will download the tutorial files (including both the manuscript and the examples in the `code` folder).

4.3 System Preparation and Equilibration

If you want to skip this section, run

```
make equilibration-targets
```

in the code folder. This assumes that you have Python and Amber set up properly and that `pmemd.cuda` is in the PATH.

When calculating solvation free energy differences between a complex and the corresponding monomers (the dissociated state), there are two options for dealing with configurational diversity in the structures. Maybe Valentin knows a citation for that? One can choose to sample the complex and dissociated states individually, or to sample only one of them and assume that there is sufficient overlap in the sampled conformations.

Here, we choose not to include any sampling of the conformational states. It is therefore important that the conformations of the monomers in the complex and dissociated states match exactly. Use `cpptraj` to produce biotin and streptavidin structures based on the complex by stripping the respective other molecule like this:

```
parm complex/solvated.parm7
trajin complex/equil/EQUIL-DONE.rst
strip ^2 parmout streptavidin/solvated.parm7
trajout streptavidin/pre-equil.ncrst
```

(Adapt for biotin as necessary.) vah: Maybe include a section on how to prepare a protein structure before equilibration. Protonation, force field and a large enough iso solvation box seem important to me. I will probably also at least mention what simulation settings we are running later on.

Use the `equilibration.py` script (or your own) to perform short NVT and NpT equilibration runs on the starting structure. Note that the barostat might interfere with restrained equilibration of the complex, since different molecules are moved relative to each other when the volume changes. But since our equilibration script also contains unrestrained pressure equilibration, we should not run into any trouble.

Then split the equilibrated structure into biotin and streptavidin parts using `cpptraj_remove_mol.sh`. After that, you can use `equilibration.py -R` to run short restrained equilibrations (where only the water is allowed to move) on the individual systems.

4.4 Running MD for GIST

Now, you can run restrained MD simulations based on the equilibrated structures for subsequent GIST analyses. We apply a harmonic restraint of 100 kcal mol^{-1} Å^{-2} to all heavy atoms to keep the system in place. We run 100 ns of simulation and store the conformations every 100 ps. A simulation length of at least 10 ns to 20 ns is recommended, to obtain statistically independent snapshots of the water.

An example Amber input file for GIST might look like this:

```
restrained 100 ns NpT
&cntrl
  ntx=5, irest=1,
  ioutfm=1,
  ntb=2, iwrap=1,
  ntr=1, restraint_wt=100.0,
    restraintmask="!@H=&!:WAT",
  ntp=1, pres0=1.0, taup=1.0,
  ntc=2, ntf=2,
  ntt=3, tempi=300.0, temp0=300.0, gamma_ln=2,
  nstlim=50000000, dt=0.002,
  ntwr=50000, ntwx=5000,
  ntpr=5000,
/
```

Note that these are very standard MD settings except for the line that defines the restraints.

Run the MD using `pmemd.cuda`:

```
pmemd.cuda \
        -O \
        -i 100ns-npt-restraint.in \
        -o md-01.out \
        -p solvated.parm7 \
        -c EQUIL-DONE.rst \
        -x md-01.nc \
        -r md-01.ncrst \
        -ref EQUIL-DONE.rst
```

4.5 Running the GIST analysis

For easier post-processing, we will use grids that contain the whole molecule. This could in theory be avoided for streptavidin and the complex, since only the surrounding of the biotin molecule (i.e., the binding pocket) is required for the analysis. We also recommend that you center the molecule in space. For each system (biotin, streptavidin, complex), produce a centered pdb file using `cpptraj`:

```
parm solvated.parm7
trajin md-01.nc 1 1 1
autoimage !(:WAT) origin
strip :WAT
trajout solute-centered.pdb
```

Then, find the extent (minimum and maximum) of the x-, y-, and z-coordinates for each system. This can be done e.g., using the `mdtraj` package in Python.

```
import mdtraj as md
pdb = md.load("solute-centered.pdb")
# mdtraj converts to nm.
# *10 to convert back to Angstrom.
min = pdb.xyz[0].min(0) * 10.
max = pdb.xyz[0].max(0) * 10.
print(*list(zip("xyz", min, max)), sep="\n")
```

Then, decide on grid dimensions for every system. For a system centered at the origin with a spacing of 0.5 Å and a buffer of, e.g., 7 Å between the solute and the grid wall, you need $4.0 \times (d+7)$ bins in the respective direction, where d is the larger of –min and +max in the respective direction. this does not work with the default gisttools values for the automatic detection of the energy reference. Better use a larger value. We note that this centered grid is usually larger than it needs to be in some directions. If you want to save some extra memory and calculation time, feel free to align the grid more precisely to the minimum and maximum coordinates, or use a smaller grid that only covers the binding pocket. You can also rotate the molecule by its principal axes to fit the cuboid grid more exactly. However, this needs to be done *before* the MD run, since rotating the trajectory damages the periodic box information.

Now, you can finally run the actual GIST analysis. Center the molecule exactly in the same way as above, to make sure that its position matches the grid parameters that you decided on. We recommend using either the PME or GPU implementation of GIST, depending on whether you have a fast GPU. A further advantage of the PME implementation is that it matches the energy calculation of the MD engine more closely.

```
parm solvated.parm7
trajin md-01.nc
autoimage !(:WAT) origin
# remove the "pme" option if you want to use the GPU
gist griddim 100 100 100 out gist.dat \
  refdens 0.03287 pme
# other settings you might want to change
# (those are the defaults)
# gridcntr 0 0 0 gridspacn 0.5 temp 300
```

4.6 Reference Values and Radial Convergence

All GIST quantities should be expressed relative to bulk. In practice, this is not problematic for the entropies, since they are computed relative to a distribution of randomly oriented molecules at the reference density. The solute-solvent energy E_{sw} is also no problem, because it naturally tends to zero in bulk. However, the solvent-solvent energy E_{ww} needs to be referenced.

The Amber manual provides reference energies for several solvents. However, the exact reference value is different when using PME-GIST, and also depends on the box size. For exact analyses such as computing the total ΔG_{solv} of a compound, it is important to compute an exact reference energy.

The most accurate method to compute a reference energy is to run a separate GIST calculations with the same energy method (PME/GPU) and a similar box size, and then compute the average solvent-solvent energy per molecule.

$$E_{ww}^{ref} = \frac{\int E_{ww} \mathrm{d}x}{\langle N_{solvent} \rangle} = \frac{\sum_{voxels} E_{ww}^{dens} V_{vox}}{\sum_{voxels} N_{solvent}/N_{frames}} \tag{17}$$

Alternatively, one can note that the solvent-solvent energy in a sufficiently large GIST grid converges to the correct bulk value at large distance to the solute. This means that the reference energy can be obtained from a large GIST grid by binning the voxels by their distance to the solute, and evaluate Equation 17 within each bin. If this value converges to a constant, this can be used as E_{ww}^{ref}.

`gisttools` contains functionality to perform this analysis easily. There is also a method `detect_reference_value` that automatically tries to find the converged value from the rdf. Although it has a simple convergence check built-in, it is always recommended to check the convergence by hand as well.

```
import gisttools as gt
import matplotlib.pyplot as plt
gist = gt.gist.load_gist_file(
    "gist.dat", struct="solute-centered.pdb")
bins, eww = gist.rdf(
    "Eww_unref_norm", bins=100, rmax=20., normalize="norm")
plt.plot(bins, eww)
eww_auto = gist.detect_reference_value().values[0]
plt.axhline(eww_auto, color='k')
plt.gca().set(
    xlabel='distance␣[A]', ylabel='E_ww␣per␣mol')
plt.show()
```

While not massively important, I don't like that we call the gist object here "gist", as that's the same name as gisttools

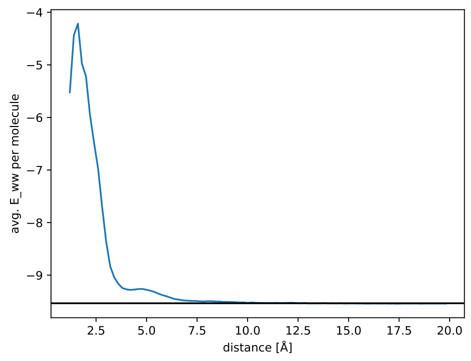

Figure 1. Convergence of E_{ww} with increasing distance to the solute. The horizontal line shows the automatically computed reference energy. Computed from the complex calculation.

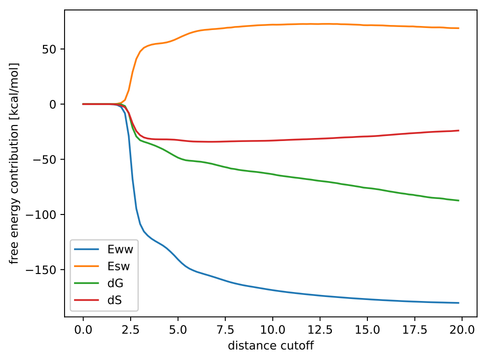

Figure 2. Convergence of ΔG_{solv} and its contributions with increasing distance to the solute. All quantities cumulative, i.e., summed up to the respective radius. Computed from the biotin calculation.

subpackage gist. "gt.gist.bla" and "gist.bla" therefore look quite similiar, but do very different things. -vah The expected output of this code is shown in Figure 1.

After choosing an appropriate reference value, we need to subtract this value from E_{ww}. Make sure to subtract the reference from the normalized (_norm) data. In gisttools, this is done simply by setting `gist.eww_ref`. In gistpp, this can be done as follows: Is there some official documentation for gistpp? We should probably link that somewhere, I feel that the code presented here is not really helpful/legible without further documentation. -vah This might become invalid with future GIST versions, depending on whether we adopt the COM as molecular center. Better use the population.

```
rho0=0.03287465
neg_eww_ref=9.533
gistpp -op multconst -i gist-g0.dx -opt const $rho0 \
    -o gist-g0-abs.dx
gistpp -op div -i gist-Eww-dens.dx -i2 gist-g0-abs.dx \
    -o gist-Eww-norm.dx
gistpp -op addconst -i gist-Eww-norm.dx \
    -opt const $neg_eww_ref -o gist-Eww-ref-norm.dx
gistpp -op mult -i gist-Eww-ref-norm.dx \
    -i2 gist-g0-abs.dx -o gist-Eww-ref-dens.dx
```

Now, add all free energy contributions to obtain the spatially resolved ΔG_{solv}. In gisttools, this is done automatically (simply use the `A_dens` and `A_norm` columns). In gistpp, use the `add` command, with the same syntax as the `div` command used above. After that, check whether your free energy contributions become negligible at high distances to the solute. In a plot like Figure 1 based on the `A_dens` column, the values should tend to zero. However, it is more informative to plot the cumulative free energy contribution against the distance to the solute, to check the convergence. It the curve flattens out, the converged value is your final ΔG_{solv}. If it does not, you might need to tweak the E_{ww} reference, or introduce a reference value for the entropy (especially with older versions of cpptraj).

```
import gisttools as gt
import matplotlib.pyplot as plt
# adapt eww_ref!
gist = gt.gist.load_gist_file("gist.dat",
    struct="solute-centered.pdb", eww_ref=-9.533)
bins, (dg, esw, eww, s) = gist.rdf(
    ["A_dens", "Esw_dens", "Eww_dens", "dTSsix_dens"],
    bins=100, rmax=20., normalize="none")
plt.plot(bins, np.cumsum(dg), label="dG")
plt.plot(bins, np.cumsum(eww), label="Eww")
plt.plot(bins, np.cumsum(esw), label="Esw")
plt.plot(bins, np.cumsum(s), label="dS")
```

The expected output of this code is shown in Figure 2. Both the solvent-solvent energy and the entropy are not perfectly flat, indicating that the reference value is not optimal. The slow energy convergence indicates that a non-optimal reference value was used, but is also complicated by the fact that a charged solute is used. In this case, both the entropy and E_{sw} can be considered converged around 10 Å. While this is not true for E_{ww}, we will see below that the energy difference upon binding converges more readily than its individual contributions. vah: Is that shown somewhere yet?

4.7 Visualizing ΔG_{solv}

Next, we want to visualize the 3-dimensional contributions to the free energy of hydration, as well as the entropy and enthalpy contributions. You can use PyMol[25] or VMD[26] to visualize the .dx files from GIST. With gistpp, you can use the file from the convergence check. Additional OpenDX files can be produced using `gistpp -op makedx`. With gisttools, you can create one using e.g., `gist.save_dx("A_dens", "A_dens.dx")`.

It might also be interesting to visualize the average energy of a water molecule at each grid voxel. This quantity is given by 2*Eww + Esw. The energy referencing leads to non-zero

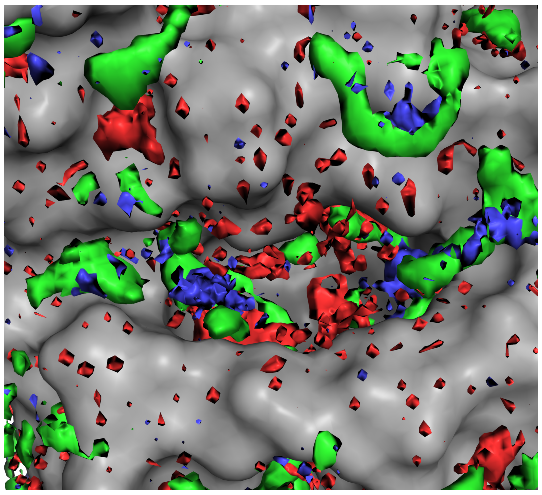

Figure 3. Free energy contributions in the streptavidin binding pocket. Red: high solvent energy (at +3 kcal/mol). Blue: low solvent energy (at -2 kcal/mol). Green: low solvent entropy $T\Delta S^{six}$ (at -2 kcal/mol).

normalized values where the population is zero. While this is irrelevant for further post-processing, it is advantageous for visualization to set those empty regions to zero. An OpenDX file can be produced using `gisttools` as follows:

```
import gisttools as gt
gist = gt.gist.load_gist_file("gist.dat",
     struct="solute-centered.pdb", eww_ref=-9.533)
gist['E_norm'] = gist['Eww_norm'] * 2 + gist['Esw_norm']
gist.loc[gist['population'] == 0, 'E_norm'] = 0
gist.save_dx('E_norm', 'gist-E-per-mol-norm.dx')
```

Visualize the free energy, the total energy, and the entropy, at several iso-levels. A minimal PyMOL input might look like this:

```
load solute-centered.pdb
show sticks, resname BTN
as surface, not resname BTN
load gist-g0.dx, g0
isosurface g0_high, g0, 2.
```

Adapt the above PyMOL script to visualize the average energy per solvent at a positive and a negative isolevel, as well as regions of negative entropy per solvent. The expected representation of the streptavidin binding pocket is shown in Figure 3.

You should find that there are several regions in the binding pocket that feature a high solvent energy and/or low solvent entropy. However, such regions also exist outside of the binding pocket, and there are also low-energy regions within the binding pocket. Therefore, the strong affinity to biotin can not be explained solely by the hydration contribution, and the direct interaction energy also needs to be taken into account.

4.8 Contribution of Hydration to Binding

In the next step, calculate the ΔG_{solv} contributions around the biotin ligand in each of the three systems (biotin, streptavidin, and complex). Note that our integration region does not fully reach into bulk, but also comprises regions that are close to the streptavidin protein. Therefore, our results for each system will depend on the exact position of the integration boundary. To avoid inconsistencies, it is important to choose exactly the same integration region for each system. Although we kept the systems rigid, the center of mass might have been shifted during pressure equilibration. Therefore, we align the complex structure to the streptavidin system and use the shifted biotin coordinates to define the integration region.

We recommend using a `Jupyter Notebook` for the following analyses, although they can also be done in a script or an interactive Python prompt. The code is not explained in detail here, but should be clear after reading the previous sections.

Load the files and double check the frame numbers and reference density.

```
import numpy as np
from gisttools.gist import load_gist_file
import matplotlib.pyplot as plt

compl = load_gist_file('complex/gist/gist.dat', \
    struct='complex/gist/solute-centered.pdb')
print(compl.n_frames, compl.rho0)
biotin = load_gist_file('biotin/gist/gist.dat', \
    struct='biotin/gist/solute-centered.pdb')
print(biotin.n_frames, biotin.rho0)
strept = load_gist_file('streptavidin/gist/gist.dat', \
    struct='streptavidin/gist/solute-centered.pdb')
print(strept.n_frames, strept.rho0)
```

Maybe add more complete print calls? E.g. print('Biotin: Frames={biotin.n_frames}, Rho={biotin.rho0} -vah) Assign reference energies and check them for plausibility.

```
biotin.eww_ref = biotin.detect_reference_value()
print("Biotin:", biotin.eww_ref)
strept.eww_ref = strept.detect_reference_value()
print("Streptavidin:", strept.eww_ref)
compl.eww_ref = compl.detect_reference_value()
print("Complex:", compl.eww_ref)
```

Subtract a reference entropy from the dTSsix columns. Is there actually a reason why this is not in gisttools itself implemented? -vah

```
def reference_entropy(gf):
    if 'dTSsix_unref_norm' not in gf.data.columns:
        gf['dTSsix_unref_norm'] = gf['dTSsix_norm']
        gf['dTSsix_unref_dens'] = gf['dTSsix_dens']
    gf['dTSsix_norm'] = gf['dTSsix_unref_norm'] \
        - gf.detect_reference_value('dTSsix_unref_dens')
    gf['dTSsix_dens'] = gf.norm2dens(gf['dTSsix_norm'])
reference_entropy(biotin)
reference_entropy(strept)
reference_entropy(compl)
```

Compute the atom positions that define the integration region. For the streptavidin integral, we align the complex structure onto streptavidin and then use the biotin positions as centers. Note that we use `mdtraj` here, since `gisttools` stores the reference structure as a `mdtraj.Trajectory` object, but any other library for post-processing of trajectories works as well.

```
col = 'dTSsix_dens'
def select(traj, sel):
    "Slice␣a␣Trajectory␣by␣selection␣mask."
    return traj.atom_slice(traj.top.select(sel))
# we multiply by 10 to convert nm to Angstrom.
compl_x = select(compl.struct, biotin_mask).xyz[0] * 10.
biotin_x = select(biotin.struct, biotin_mask).xyz[0] * 10.
aligned = compl.struct[:].superpose(strept.struct, \
    atom_indices=strept.struct.top.select(strept_mask))
aligned = select(aligned, biotin_mask)
strept_x = aligned.xyz[0] * 10.

bins, biotin_rdf = biotin.rdf( \
    col, centers=biotin_x, bins=100, rmax=24)
bins, strept_rdf = strept.rdf( \
    col, centers=strept_x, bins=100, rmax=24)
bins, compl_rdf = compl.rdf( \
    col, centers=compl_x, bins=100, rmax=24)
```

Now, subtract the monomer rdfs from the complex, and compute the integrals. If you also visualize the individual rdfs, you will notice that the difference converges much better with increasing radius than the contributions. This is because the cutoff contains regions that are close to the atom, since the integration region does not comprise the whole protein.

```
difference = compl_rdf - biotin_rdf - strept_rdf
cutoff = 12
integral = difference[bins < cutoff].sum()
print("Integral␣=␣{}".format(integral))
plt.plot(bins, np.cumsum(difference))
plt.axvline(cutoff)
plt.xlabel('distance␣to␣biotin␣[Å]')
plt.ylabel('dG␣contribution␣[kcal/mol]')
```

Compute the energy (`Eall`) and entropy (`dTSsix`) contributions separately. You will find that the energy is strongly unfavorable to binding, because we do not yet include the interaction energy between biotin and streptavidin.

Using the `energy` command in `cpptraj`, you can compute this energy difference. We recommend using PME in combination with PME-GIST, but not with GPU-GIST. An example `cpptraj` input might look like:

```
parm solvated.parm7
trajin md-01.nc 1 last 100
energy complex ^1,2 # etype pme
energy strept ^1 # etype pme
energy biotin ^2 # etype pme
go
diff = complex[total] - strept[total] - biotin[total]
writedata energy.dat diff complex[total] \
strept[total] biotin[total]
avg(complex[total])
avg(strept[total])
avg(biotin[total])
avg(diff)
```

It has been shown in literature [18, 20] that the total solvation entropy in water is best approximated by 0.6 times the first order entropy. Also compute the scaled entropy and check the effect on the binding affinity. The expected results are summarized in Table 1.

Table 1. Free energy contributions for the monomers and the dimer.

System	E(internal)	ΔE^{GIST}	ΔS^{GIST}	ΔS^{scaled}	total
complex	-1303.4	-361.2	-233.1	-139.9	-192.0
streptavidin	-1180.7	-360.8	-242.4	-145.4	-188.4
biotin	-24.0	-98.1	-35.1	-21.1	-105.4
Diff	-98.7	97.7	44.4	26.6	-27.6

Even though streptavidin-biotin is known to be a very stable complex, the solvation energy favors the dissociation. This is expected: if the molecules were not able to form any interactions, but interacted fully with the water, they would not form a complex.

In this case, we find that the energy of solvation and the internal energy of the complex more or less cancel out, indicating that binding is largely entropy-driven in this case. At first, this is surprising, since isothermal titration calorimetry (ITC) of biotin-streptavidin shows strong enthalpic binding contributions [28, 29]. However, we do not take the conformational transition of the streptavidin binding pocket into account. Prior computational studies suggest a ΔG of $-26.6\,\mathrm{kcal\,mol^{-1}}$ for the binding of biotin into the closed conformation of streptavidin [30]. This indicates that our result is in good agreement with literature. However, the conformational changes of the binding pocket would need to be described to improve the agreement with experiments. Should we include a short introduction to other solvents with GIST? What about solvent/ion mixtures? Can we somehow distinguish python code from input files optically? -vah

4.9 Further steps

In this tutorial, we obtained a value for the binding of biotin into the closed conformation of the streptavidin binging pocket. To investigate the effect of different conformations on the binding affinity, you can perform molecular dynamics simulations of the complex and/or monomers and perform GIST on multiple cluster representatives. To include the effect of the lid closing transition in the binding process, free energy calculation methods such as umbrella sampling can be used.

5 Recent developments

5.1 GIST with non-water solvents

5.2 GIST with salt-water mixtures

6 Checklists

SIMULATION SETTINGS

Each number denotes the minimum setting, with the optimum in brackets.

- ☐ Simulation time: 10 ns to 20 ns (100 ns)
- ☐ Number of analyzed frames: 10 000 (100 000)
- ☐ Restraints: typically 100 kcal mol^{-1} Å^{-2} on solute heavy atoms.

OBTAINING ABSOLUTE ΔG_{SOLV}

- ☐ Check the radial convergence (see Fig. 2)
- ☐ Choose a sufficient distance cutoff
- ☐ Choose an optimal E_{ww} reference value
- ☐ Tweak simulation length and number of frames to obtain smooth free energy contributions and unbiased (i.e., zero) bulk entropy.
- ☐ If necessary, subtract an entropy reference

Author Contributions

(Explain the contributions of the different authors here)

For a more detailed description of author contributions, see the GitHub issue tracking and changelog at https://github.com/myaccount/homegithubrepository.

Other Contributions

(Explain the contributions of any non-author contributors here) For a more detailed description of contributions from the community and others, see the GitHub issue tracking and changelog at https://github.com/myaccount/homegithubrepository.

Potentially Conflicting Interests

Declare any potentially conflicting interests here, whether or not they pose an actual conflict in your view.

Funding Information

FMS acknowledges the support of NSF grant CHE-1111111.

Author Information

ORCID:

Author 1 name: AAAA-BBBB-CCCC-DDDD
Author 2 name: EEEE-FFFF-GGGG-HHHH

References

[1] Sitkoff D, Sharp KA, Honig B. Accurate calculation of hydration free energies using macroscopic solvent models. J Phys Chem. 1994; 98(7):1978–1988. https://doi.org/10.1021/j100058a043.

[2] Kollman PA, Massova I, Reyes C, Kuhn B, Huo S, Chong L, Lee M, Lee T, Duan Y, Wang W, Donini O, Cieplak P, Srinivasan J, Case DA, Cheatham TE 3rd. Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models. Acc Chem Res. 2000; 33(12):889–897. https://doi.org/10.1021/ar000033j.

[3] Genheden S, Ryde U. The MM/PBSA and MM/GBSA methods to estimate ligand-binding affinities. Expert Opin Drug Discov. 2015; 10(5):449–461. https://doi.org/10.1517/17460441.2015.1032936.

[4] Zwanzig RW. High-temperature equation of state by a perturbation method. I. nonpolar gases. J Chem Phys. 1954; 22(8):1420–1426. https://doi.org/10.1063/1.1740409.

[5] Kirkwood JG. Statistical mechanics of fluid mixtures. Journal of Chemical Physics. 1935; 3(5):300–313. https://doi.org/10.1063/1.1749657.

[6] Peter C, Oostenbrink C, van Dorp A, van Gunsteren WF. Estimating entropies from molecular dynamics simulations. J Chem Phys. 2004; 120(6):2652–2661.

[7] Lazaridis T. Inhomogeneous fluid approach to solvation thermodynamics. 1. Theory. Journal of Physical Chemistry B. 1998; 102(18):3531–3541. https://doi.org/DOI 10.1021/jp9723574.

[8] Hansen JP, McDonald IR. Theory of simple liquids. 4 ed. San Diego, CA: Academic Press; 2013.

[9] Chandler D, Andersen HC. Optimized cluster expansions for classical fluids. II. Theory of molecular liquids. J Chem Phys. 1972; 57(5):1930–1937. https://doi.org/10.1063/1.1678513.

[10] Kovalenko A, Hirata F. Three-dimensional density profiles of water in contact with a solute of arbitrary shape: a RISM approach. Chemical Physics Letters. 1998; 290(1-3):237–244. https://doi.org/10.1016/S0009-2614(98)00471-0.

[11] Nguyen CN, Young TK, Gilson MK. Grid inhomogeneous solvation theory: Hydration structure and thermodynamics of the miniature receptor cucurbit[7]uril. Journal of Chemical Physics. 2012; 137(4):044101. https://doi.org/10.1063/1.4733951.

[12] Ramsey S, Nguyen C, Salomon-Ferrer R, Walker RC, Gilson MK, Kurtzman T. Solvation thermodynamic mapping of molecular surfaces in AmberTools: GIST. Journal of Computational Chemistry. 2016; 37(21):2029–2037. https://doi.org/10.1002/jcc.24417.

[13] Young T, Abel R, Kim B, Berne BJ, Friesner RA. Motifs for molecular recognition exploiting hydrophobic enclosure in protein–ligand binding. Proceedings of the National Academy of Sciences. 2007; 104(3):808–813. https://doi.org/10.1073/pnas.0610202104.

[14] Abel R, Young T, Farid R, Berne BJ, Friesner RA. Role of the active-site solvent in the thermodynamics of factor Xa ligand binding. J Am Chem Soc. 2008; 130(9):2817–2831. https://doi.org/10.1021/ja0771033.

[15] Haider K, Cruz A, Ramsey S, Gilson MK, Kurtzman T. Solvation structure and thermodynamic mapping (SSTMap): An open-source, flexible package for the analysis of water in molecular dynamics trajectories. J Chem Theory Comput. 2018; 14(1):418–425. https://doi.org/10.1021/acs.jctc.7b00592.

[16] **Li Z**, Lazaridis T. Computing the thermodynamic contributions of interfacial water. Methods Mol Biol. 2012; 819:393–404. https://doi.org/10.1007/978-1-61779-465-0_24.

[17] **Nguyen CN**, Kurtzman T, Gilson MK. Spatial decomposition of translational water-water correlation entropy in binding pockets. J Chem Theory Comput. 2016; 12(1):414–429. https://doi.org/10.1021/acs.jctc.5b00939.

[18] **Waibl F**, Kraml J, Hoerschinger VJ, Hofer F, Kamenik AS, Fernández-Quintero ML, Liedl KR. Grid inhomogeneous solvation theory for cross-solvation in rigid solvents. J Chem Phys. 2022; 156(20):204101. https://doi.org/10.1063/5.0087549.

[19] **Ben-Naim A**. Solvation Thermodynamics. New York, NY: Springer; 2013. https://doi.org/10.1007/978-1-4757-6550-2.

[20] **Chen L**, Cruz A, Roe DR, Simmonett AC, Wickstrom L, Deng N, Kurtzman T. Thermodynamic Decomposition of Solvation Free Energies with Particle Mesh Ewald and Long-Range Lennard-Jones Interactions in Grid Inhomogeneous Solvation Theory. Journal of Chemical Theory and Computation. 2021; 17(5):2714–2724. https://doi.org/10.1021/acs.jctc.0c01185.

[21] **Darden T**, York D, Pedersen L. Particle mesh Ewald: An N·log(N) method for Ewald sums in large systems. J Chem Phys. 1993; 98(12):10089–10092. https://doi.org/10.1063/1.464397.

[22] **Case DA**, Aktulga HM, Belfon K, Ben-Shalom IY, Berryman JT, Brozell SR, Cerutti DS, Cheatham TEI, Cisneros GA, Cruzeiro VWD, Darden TA, Duke RE, Giambasu G, Gilson MK, Gohlke H, Goetz AW, Harris R, Izadi S, Izmailov SA, Kasavajhala K, et al., Amber 2022. University of California, San Francisco; 2022.

[23] **Kluyver T**, Ragan-Kelley B, Pérez B, Granger B, Bussonnier M, Frederic J, Kelley K, Hamrick J, Grout J, Corlay S, Ivanov P, Avila D, Abdalla S, Willing C, Jupyter Development Team. Jupyter Notebooks—a publishing format for reproducible computational workflows. In: Loizides F, Schmidt B, editors. *Positioning and Power in Academic Publishing: Players, Agents and Agendas* IOS Press BV; 2016. p. 87–90. https://doi.org/10.3233/978-1-61499-649-1-87.

[24] **Granger BE**, Perez F. Jupyter: Thinking and storytelling with code and data. Comput Sci Eng. 2021; 23(2):7–14. https://doi.org/10.1109/MCSE.2021.3059263.

[25] **Schrodinger L**. The PyMOL Molecular Graphics System, Version 1.8; 2015.

[26] **Humphrey W**, Dalke A, Schulten K. VMD – Visual Molecular Dynamics. Journal of Molecular Graphics. 1996; 14:33–38.

[27] **Kraml J**, Hofer F, Kamenik AS, Waibl F, Kahler U, Schauperl M, Liedl KR. Solvation Thermodynamics in Different Solvents: Water–Chloroform Partition Coefficients from Grid Inhomogeneous Solvation Theory. Journal of Chemical Information and Modeling. 2020; 60(8):3843–3853. https://doi.org/10.1021/acs.jcim.0c00289.

[28] **Mpye KL**, Gildenhuys S, Mosebi S. The effects of temperature on streptavidin-biotin binding using affinity isothermal titration calorimetry. Aims Biophysics. 2020; 7(4):236–247. https://doi.org/10.3934/biophy.2020018.

[29] **Hyre DE**, Le Trong I, Merritt EA, Eccleston JF, Green NM, Stenkamp RE, Stayton PS. Cooperative hydrogen bond interactions in the streptavidin-biotin system. Protein Science. 2006; 15(3):459–467. https://doi.org/10.1110/ps.051970306.

[30] **Bansal N**, Zheng Z, Song LF, Pei J, Merz KM Jr. The role of the active site flap in streptavidin/biotin complex formation. J Am Chem Soc. 2018; 140(16):5434–5446.