Quantifying Spatially Resolved Hydration Thermodynamics Using Grid Inhomogeneous Solvation Theory [Article v1.0]

Valentin J. Egger-Hoerschinger[1†], Franz Waibl[1†], Vjay Molino[2], Helmut Carter[3], Monica L. Fernández-Quintero[1], Steven Ramsey[7], Daniel R. Roe[5], Klaus R. Liedl[1*], Michael K. Gilson[4*], Tom Kurtzman[6,7*]

[1]Department of General, Inorganic and Theoretical Chemistry, University of Innsbruck, Austria; [2]Ph.D. Program in Biochemistry, The Graduate Center of the City University of New York, New York, USA 10016; [3]Ph.D. Program in Biology, The Graduate Center of the City University of New York, New York, USA 10016; [4]Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California, San Diego, USA; [5]Laboratory of Computational Biology, National Heart, Lung, and Blood Institute, National Institutes of Health, Bethesda, Maryland, USA; [6]Ph.D. Programs in Biochemistry, Chemistry, and Biology, The Graduate Center of the City University of New York, New York, USA 10016; [7]Department of Chemistry, Lehman College, The City University of New York, Bronx, New York, USA 10468

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

This version dated August 14, 2025

Abstract Grid Inhomogeneous Solvation Theory (GIST) is a method to compute the free energy of solvation of a solute molecule on a three-dimensional grid based on sampling from molecular dynamics (MD) simulations. The high spatial resolution of the GIST output, as well as the decomposition into energy and entropy contributions, allow for highly detailed analyses of solvation around both proteins and small molecules. However, this versatility also comes with a significant entry barrier for new users.

In this tutorial, we aim to guide the reader through the most common steps involved in a GIST analysis using the streptavidin-biotin complex as a demonstrative system. To this end, Jupyter notebooks and a Python package (gisttools) are provided to simplify the analysis. Furthermore, we discuss the theory of GIST with a focus on practical aspects. We highlight potential pitfalls and provide strategies to avoid technical difficulties. This tutorial assumes familiarity with molecular dynamics simulations and the AmberTools package.

***For correspondence:**
klaus.liedl@uibk.ac.at (KRL); mgilson@health.ucsd.edu (MKG); thomas.kurtzman@lehman.cuny.edu (TK)

[†]These authors contributed equally to this work

Received: 12 July 2024
Accepted: 07 July 2025

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

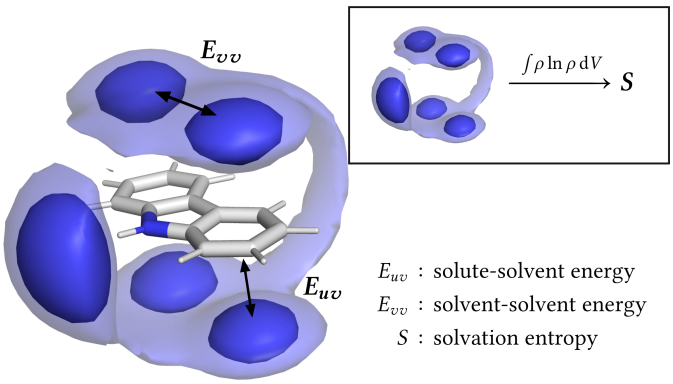


Figure 1. Schematic showing regions of high water density around a carbazole solute molecule. Based on an MD simulation of solvent around the solute molecule, the energy is computed from the intermolecular interactions of a force field, while the entropy is approximated from the 6-dimensional orientational and translational solvent distribution ρ around the solute molecule. Note that while this depiction focuses on regions of high solvent density, regions with lower density can also contain meaningful contributions to the overall energy and entropy.

1 Introduction

Solvation thermodynamics play a central role in many biological and chemical processes, particularly those involving aqueous solutions, hydrophobic effects, and partitioning between environments. Water is especially important, as most biological processes occur in aqueous conditions. The thermodynamics of solvation govern aqueous solubility and hydrophobic effects, which in turn influence numerous biological functions including ligand binding, protein folding, and the physical properties of cell membranes. Grid inhomogeneous solvation theory (GIST) [1] is a method to compute the free energy of solvation, with applications in the context of drug design and biophysical property description. It computes the solvent density from an MD trajectory and estimates localized entropic and energetic contributions on a grid around the solute molecule. Since these contributions are spatially resolved, GIST improves interpretability and offers more detailed insights into the solvation of the investigated solute. A schematic of this is shown in Figure 1. The energetic contributions are computed from the intermolecular interactions of the solvent with the solute as well as between solvent molecules. As they are derived from the force field, pairwise additivity of the underlying intermolecular interactions is assumed. The entropy is computed from orientational and translational distributions of the solvent around the solute molecule, in a first-order approximation. The calculation of higher-order entropy terms is computationally demanding but was realized in recent works [2, 3]. However, it has been found that solvation free energies of small molecule solutes can be computed at high accuracy using linear scaling factors for the higher-order terms. A common approximation is to scale the first-order entropy, for example, $\Delta S = 0.6\,\Delta S^{\text{1st-order}}$ for water [4, 5]. Since its initial development, GIST has been used to describe the thermodynamics of water displacement in protein binding pockets [2, 6–8]. It has been used for a large-scale investigation of the hydration properties of proteins found in SARS-CoV-2 [9]. Furthermore, recent works used GIST to describe biophysical properties of serine proteases [10], macrocycles [11], or antibody fragments [12].

GIST has been implemented on the GPU, substantially improving its performance for large grids [10]. Furthermore, an implementation based on the particle mesh Ewald (PME) [13] method improves both the efficiency and the agreement with popular molecular dynamics engines [4]. Other extensions of GIST allow the inclusion of rigid solvents other than water [5, 14] or the use of salt-water mixtures as solvents [3].

In this tutorial, we aim to give the reader an introduction to GIST and its application to calculate solution thermodynamics properties of interest. While previous publications [6] and tutorials [15] treated how to set up and run GIST calculations in detail, the post-processing was often treated in less detail. While building on these tutorials, we provide an updated guide to calculating solvation properties using GIST to address recent improvements. Furthermore, we aim to establish best practices for the analysis and interpretation of GIST calculations.

1.1 Comparison to other methods

A wide range of methods has been devised to compute the free energy of hydration. Data-driven methods include QSPR and UNIFAC [16, 17]. Polarizable continuum models (PCMs) [18, 19] treat the solvent as a homogeneous phase that reacts to the electrostatic potential of the solute molecule [20]. Methods such as MM/PBSA or MM/GBSA [21, 22] combine an implicit treatment of solvent electrostatics with structural sampling of the solute molecule. They can quickly provide results for a large number of structures [23].

On the other hand, explicit solvation methods model the solvent using individual molecules. They are generally more accurate than implicit methods, but require a statistical sampling of the solvent conformations [24, 25]. Molecular dynamics (MD) simulations provide an accurate way of modeling solvent packing in confined areas [26] and general hydrophobic effects [27].

The free energy of solvation can be calculated in a statistically rigorous way using alchemical methods [24, 28, 29] such as free energy perturbation (FEP) [30] or thermodynamic integration (TI) [31]. However, these methods are unable to provide a spatial interpretation of the results. Furthermore,

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

splitting a free energy into energetic and entropic contributions is challenging [32].

Methods derived from statistical mechanics bridge this gap by providing spatially resolved information. It is important to note that while the total solvation free energy is a well-defined thermodynamic quantity, its spatial decomposition is not unique and depends on the theoretical framework[33]. These methods most commonly are based on either the Ornstein–Zernike (OZ) equation, or [34] inhomogeneous fluid solvation theory (IST) [35]. The most important example of OZ-based methods is the reference interaction site model (RISM) [36] as well as its extension to three dimensions (3D-RISM) [37]. These methods compute the atomic solvent distributions by self-consistently solving the OZ integral equation to predict solvation thermodynamics.

Methods based on IST use a molecular solvent distribution obtained from an MD simulation to compute the free energy of hydration from its energetic and entropic contributions, providing easier interpretability of the resulting distributions. Examples include grid inhomogeneous solvation theory (GIST) [1, 6], WaterMap [38, 39], SSTMap [40], STOW [41] and Solvaware [42]. The main limitation of these methods is that the entropy is approximated through an infinite expansion in terms of the density distribution. Within GIST and other more recent methods, these entropy contributions are calculated via a k-nearest neighbor (kNN) approach. Due to the high computational effort required to converge the higher order terms, the entropy expansion is usually truncated after the first or, sometimes, the second [2, 5] term, thereby introducing an error.

While there are approximations available to deal with these problems, most notably the mutual information expansion (MIE) and Maximum Information Spanning Tree (MIST) approaches, they are not currently implemented in GIST. Recent kNN implementations to calculate solvation entropy further improve phase space sampling by relabeling the solvent molecules in the simulation, treating the solvent molecules as distinguishable [43–46]. For a recent review of kNN methods to calculate entropy in a molecular context, see [47]. Furthermore, IST does not consider the internal degrees of freedom of the solvent. Therefore, IST-based approaches currently can only be used with sufficiently rigid solvents, although a recent paper provides the theory needed to generalize to flexible solvent molecules [48]. Nevertheless, they have been applied to a wide variety of problems involving both small molecules and biological systems as solutes.

GIST is an implementation of IST on a three-dimensional grid. Unlike the related method, WaterMap [38, 39], it provides solvation thermodynamics on an entire grid, rather than only high-density clusters in a binding pocket. This implies a higher versatility, but also puts more responsibility on the user in terms of correct post-processing. Since the density is computed from an explicit solvent MD simulation, GIST provides an accurate picture of solvation at an atomistic level, even for demanding systems such as drug binding sites in proteins. While all MD simulations require a large computational cost, GIST is more efficient than end-state approaches such as TI [31] or FEP [30], with applications to large systems such as serine proteases [10] or antibody fragments [12]. On the other hand, the first-order approximation means that the solvation entropy is less accurately described than in TI or FEP [3, 4].

Similar to the self-consistent integral theories, GIST only considers a single solute conformation and multiple computations of representative structures are necessary to describe ensemble properties.

Aside from the previously mentioned methods, there is a large variety of other methods to compute energetic and entropic contributions to the solvation free energy, as recently reviewed [49, 50]. Approaches like SZMAP [51] trade the accuracy of all-atom MD simulation for speed by using a faster probe-based approach. SPAM ("maps" in reverse) is more akin to WaterMap, investigating distinct hydration sites based on high local water densities [52]. However, it approximates the water entropies from the water interaction energy distributions. Another group of methods builds on the cell theory for liquids [53], which derives independent effective potentials for each molecule in its neighbors' mean field. Conceptually close to GIST is Grid Cell Theory (GCT), which also adopts a grid-based approach but differs in the assumptions made in its entropy calculations [54]. More recently, Multiscale Cell Correlation (MCC) has been developed, generalizing and pushing the method from simple liquids up to large biomolecules [55–58].

1.2 Scope

The purpose of this tutorial is to help users run GIST calculations and interpret the results. The tutorial is structured around two examples: a GIST calculation for the small molecule biotin, and a GIST study of biotin-streptavidin binding. Biotin is used as a simple example to introduce the basics of GIST calculations, whereas the full biotin-streptavidin analysis highlights details and considerations relevant to more advanced GIST studies. 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 contribution of solvation free energy to binding, which requires accurate post-processing of the GIST outputs to avoid unfavorable summation of biases. In total, we aim to provide a workflow that can be easily adapted to different systems, an introduction to the

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

method and the theory behind it, and an overview of the different implementations of GIST.
Furthermore, we discuss several technical aspects of GIST studies, such as the normalization of voxel values. Additionally, we provide a Python library (`gisttools`) to unify the analysis of GIST data produced by various versions of GIST and to 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

2.1 Background knowledge

This tutorial is aimed at users with a solid knowledge of molecular dynamics (MD) simulations.
Users should be able to run MD simulations and analyze the resulting trajectories. The MD simulations presented in this tutorial are run with the Amber simulation engine and analyzed using AmberTools `cpptraj`. For a conceptual overview of running MD simulations with Amber, we recommend reading the Amber manual (https://ambermd.org/Manuals.php) [59] and the Amber tutorials (https://ambermd.org/tutorials).
Previous experience with GIST is not necessary, although we recommend reading the works referenced in section 7.
The presented analyses are run in Python. While no programming experience is necessary to follow the tutorial, a basic understanding of Python is advantageous to adjust the code towards different requirements and use-cases.

2.2 Software/system requirements

The main implementation of GIST is part of the MD analysis software `cpptraj` released with AmberTools [59, 60]. A recent version of `cpptraj` should be used. We recommend using at least version 6.24, which was released with AmberTools24 [60]. An overview of how to work with `cpptraj` is provided in the program's documentation [61], AMBER manual [59], AMBER tutorials [62] and on the AMBER-hub website [63].

The Amber simulation engine is used in this tutorial to run the MD simulations. While Amber is free for non-commercial use, it might not always be available in industrial settings. In that case, other MD engines can be used as long as topologies and trajectories can be produced in a format supported by `cpptraj`. If using GROMACS [64, 65], we propose to prepare the topology with AmberTools [60] and convert it using `acpype.py` [66]. Also, note that compressed trajectory formats such as `xtc` might bias the entropy calculation due to the loss of precision for the atom positions. For further reference, the influence of compression on energy calculations was discussed in a recent publication [67].

Furthermore, a recent Python version (3.6 or newer) should be installed, along with the following packages:

- `gisttools` (https://github.com/liedllab/gisttools, $\geq$ 0.2)
- `mdtraj` (https://www.mdtraj.org/, $\geq$ 1.9.7) [68]
- `matplotlib` (https://matplotlib.org/, $\geq$ 3.7.0) [69]
- `numpy` (https://numpy.org/, $\geq$ 1.23.5) [70]
- `pandas` (https://pandas.pydata.org/, $\geq$ 1.5.3) [71**?**]

If the reader prefers to skip the MD and GIST calculations, GIST output files are provided with this tutorial, such that the post-processing can be done without any expensive calculations. Additionally, the MD simulation files and raw GIST output files can be found at https://researchdata.uibk.ac.at//records/4mbrd-67m83, to reproduce the results shown in this tutorial. We additionally provide a Jupyter notebook [72, 73] with the presented analyses. Most of the visualizations and results in this tutorial can also be obtained using the `gistpp` program, which is available from https://github.com/KurtzmanLab/Gist-Post-Processing (code and documentation). `Gistpp` can directly manipulate `OpenDX` files from the GIST output.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

3 Running GIST

To run a GIST analysis, the following are required:

- `cpptraj`.
- A topology of the solute molecule of interest in a solvent box.
- An MD trajectory of that system.

Note that the solute should not move significantly for a GIST calculation, since IST does not take the solute degrees of freedom into account. This is typically achieved by applying restraints on the solute heavy atoms during the MD simulation. GIST analysis can be divided in three steps: defining a region of interest, running GIST using `cpptraj`, and post-processing and analysis. [6]

Identify and define the region of interest

In order to run a GIST analysis, one must define the region where solvent properties will be calculated. This region is defined by a rectangular grid with given position, size, and voxel spacing. The default grid spacing of 0.5 Å provides reasonable discretization for many use-cases. While integrals of GIST quantities over the grid are independent of the voxel size, provided sufficient sampling, individual values converge slower at high resolutions. The grid is defined by parameters of the `gist` command in `cpptraj`. The grid center is defined by its center coordinates `gridcntr`, the voxel number in x, y, and z direction by `griddim`, and the spacing (voxel side-length) in Å by `gridspacn`:

Cpptraj

```
gist gridcntr <x> <y> <z> griddim <Nx> <Ny> <Nz> gridspacn <val>
```

If the solute is already centered at the origin (0, 0, 0), the grid center does not need to be specified. As such, the GIST calculation should usually be preceded by centering the solute molecule in the unit cell and re-imaging to account for periodic boundary conditions. Often, the region of interest will be defined as the surroundings of a molecule, for example a ligand in the binding pocket, or an entire protein. We recommend using GIST grids with a similar size to the simulation box, as the GIST calculation will usually not be the time limiting step and a region of interest can then be chosen freely in post-analysis. However, if computational resources and disk space are limited, it might be beneficial to reduce the grid size. The minimum recommended grid size depends on the solvent and its interactions with the solute. Generally, a grid encompassing two to three solvent layers around the solute is sufficient. For water, this corresponds to a buffer zone of around 10 Å around the solute. Integrating GIST free energies in a 10 Å volume surrounding neutral small molecule solvents is sufficient to estimate solvation free energies [4, 5] however, larger grids might be necessary for strongly charged solutes.

Using `cpptraj`'s `bounds` command, boundaries can be output to a file. For example, here is the command for a boundary of 15 Å around the ligand LIG:

Cpptraj

```
bounds :LIG dx 0.5 offset 30 name Grid out bounds.dat
```

The atom mask `:LIG` specifies the residue named LIG to build the boundaries around, with dx and offset providing the actual dimensions of the boundaries, i.e. an offset of 30 times 0.5Å. Make sure that the `bounds` command is supplied with the exact same trajectory as that used in the GIST calculation. Any preprocessing steps need to match between the `bounds` command and the `gist` calculation. A helper script called `findcentroid.py` for the same purpose is supplied with this tutorial, and can be used as follows:

Command-line

```
python findcentroid.py -i structure.pdb
```

Run GIST using cpptraj

GIST is implemented in `cpptraj` which requires the topology and the trajectory of the MD simulation to be analyzed.

The command needed to run GIST in cpptraj can be written in an input file (e.g. gist.in) for convenience and reproducibility. The basic command, assuming a grid centered at coordinates (x=10.5 Å, y=20.1 Å, z=30.1 Å) and a grid size of $30 \times 30 \times 30$ voxels with 0.5 Å spacing, is:

Cpptraj

```
gist gridcntr 10.5 20.1 30.1 griddim 30 30 30 \
       gridspacn 0.50 out gist.dat
```

This command tells `cpptraj` to run a GIST calculation using the specified region and print the grid data in a file called `gist.dat`. Additionally, .dx files for some default thermodynamic densities are generated as well.

To use OpenMP, MPI or GPU-accelerated (using CUDA) versions of cpptraj, a different executable name such as `cpptraj.OMP`, `cpptraj.MPI` or `cpptraj.cuda` is often used. Therefore, make sure to use the right compilation and executable, to get the full benefit of the various improvements to GIST as implemented in cpptraj. The `gist` action has numerous options to modify the GIST calculation or its output. A full list is provided at the end of this tutorial or through the `cpptraj` manual. The most important options are summarized in table 1.

In GIST, the calculated properties are derived by referencing against the bulk solvent. It is therefore important to set the reference number density ρ_0 according to the solvent model used in your simulation. The solvent-solvent energy E_{ww} also needs to be referenced, but this is done in the post-processing stage. The calculation of these properties and the

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

Table 1. The main options for GIST in `cpptraj`.

Command	Options	Explanation	Default
gridcntr	<x> <y> <z>	Coordinates of the grid center in Å	0 0 0
griddim	<nx> <ny> <nz>	Number of voxels per axis	40 40 40
gridspacn	<dval>	Grid spacing / voxel dimensions in Å	0.5
refdens	<rdval>	Reference density of the bulk water in molecules/Å^3.	0.0334
temp	<tval>	System temperature in Kelvin	300

need for referencing is discussed in-depth in section 7. Densities and reference energies of typical water models can be found in the `cpptraj` manual. We also provide the densities for water models commonly used with AMBER force fields in Table 2. These densities were calculated from 100 ns MD simulations at 1 bar and 300 K using the Berendsen barostat and the Langevin thermostat. The water boxes were prepared with `tleap` to a size of 100 Å x 100 Å x 100 Å before pressure equilibration.

Because of the different sizes of the boxes, the reference energies used in this tutorial may not align with those specified in the Amber manual. In general, it is advisable to conduct your own reference calculations, especially for systems that are either very large or very small. The reference values can be extracted from unrestrained MD trajectories of pure bulk solvent by calculating the average potential energy per solvent molecule ($E_{ww,norm}$) and the average solvent density ρ_0:

$$E_{ww,norm} = \frac{\bar{E}_{pot,\,total}}{N_{WAT}} \tag{1}$$

$$\rho_0 = \frac{N_{WAT}}{\bar{V}_{box}} \tag{2}$$

It is recommended that the size of the reference solvent box closely matches that of the production system. While the impact on the density is not as pronounced, deviations from the production settings can lead to differences of up to 0.005 kcal/mol in the energy per solvent molecule ($E_{ww,norm}$) for water boxes. These discrepancies, when summing over large grid regions, have the potential to introduce significant errors due to incorrect referencing.

Postprocessing and Analyzing GIST results

Cpptraj outputs grid quantities in OpenDX (.dx) file format which can be visualized in PyMOL or VMD. Aside from thermodynamic quantities, GIST also calculates the number density of each atomic element in the solvent. Here is a list of the most important quantities calculated by GIST:

Table 2. Reference number density ρ_0 and mean solvent-solvent energy $E_{ww,norm}$ for various water models from NPT simulations of N_{WAT} water molecules at 300 K and 1bar.

Water Model	ρ_0 /Å^{-3}	$E_{ww,norm}$ /kcal · mol^{-1}	N_{WAT}
TIP3P [74]	0.03287	-9.5398	31795
TIP3PFB [75]	0.03321	-11.7214	32502
TIP4P [76]	0.03316	-9.8583	31218
TIP4PFB [75]	0.03330	-11.8724	31653
TIP4P/Ew [77]	0.03323	-11.0541	31218
TIP5P [78]	0.03285	-9.5989	31382
SPC/E [79]	0.03333	-11.1334	31795
SPC/Eb [80]	0.03358	-11.6966	31795
OPC [81]	0.03330	-12.2456	31842
OPC3 [82]	0.03323	-11.6994	32251

- [gX] For every element in the main solvent, the number density of atoms found in the voxel, in units of the bulk density. If the same element occurs multiple times, the bulk density is scaled such that the expectation value for the bulk is unity. For water, `gO` and `gH` are produced.
- [Esw] Mean solute-water interaction energy.
- [Eww] Mean water-water interaction energy.
- [PME] (only if PME was used) water PME energy.
- [dTStrans] First order translational entropy.
- [dTSorient] First order orientational entropy.
- [dTSsix] First order entropy for the six-dimensional combined orientation and translation space.
- [dipole] Magnitude of mean dipole moment (polarization).

Several dx files are written by default, such as the energy, entropy, and atomic number densities. All calculated quantities are written to the output file (default `gist-output.dat`) and can be used to generate DX files for other quantities. A full list of all properties in the output is provided at the end of the tutorial in section 8. The output file is organized by voxel and sorted according to their x, y, and z coordinates, describing the center point of the voxel. Energy and entropy quantities are reported in two different ways. The `_dens` quantities are normalized by the voxel volume and can be used to compute integrals. The `_norm` quantities are instead normalized by the solvent number density in the respective voxel averaged over all considered frames, and are useful to visualize local water properties. Converting a `_norm` quantity X^k_{norm} in voxel k to a `_dens` quantity X^k_{dens}, works as follows:

$$X^k_{dens} = \frac{X^k_{norm}}{V^k} \cdot \frac{N^k_{solvent}}{N_{frames}} \tag{3}$$

with V^k, the volume of voxel k, $N^k_{solvent}$, the number of solvent molecules in k over all simulation frames, and N_{frames}, the number of frames.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

4 Tutorial

Here, we aim to guide the reader through a GIST analysis of the streptavidin-biotin complex. The tutorial is structured as follows:

1. Set up and run GIST calculations for a small molecule solute (biotin) and a protein (streptavidin) binding pocket.
2. Compute biotin's free energy of solvation using GIST.
3. Visualize local contributions to ΔA_{solv} in the streptavidin binding pocket and around biotin.
4. Compute the hydration contribution of the biotin, streptavidin and complex to the binding. Combined with the biotin–streptavidin interaction energy, the binding free energy can be estimated.
5. Explore advantages and disadvantages of the GIST method when used to calculate ΔA_{solv}

In the first part, we show how to run GIST calculations for small molecule solutes and interpret the output. This provides an example of a simple GIST workflow, without too many details. Afterward, we apply this to a protein binding pocket and estimate a free energy of ligand binding using an end-states approach based on GIST. In addition, we discuss all steps shown in the first example in further detail.

4.1 Streptavidin/Biotin

The streptavidin-biotin complex binding is one of the strongest known non-covalent interactions involving a small molecule. Therefore, the combination of streptavidin and biotin is used routinely as a validation case [83], and the interaction between them has been investigated in detail [84].

In nature, streptavidin occurs as a tetramer, and binds biotin with a binding constant of $\sim 10^{14}\,\mathrm{M}^{-1}$ [83]. Streptavidin has a so-called flap region, which closes the binding pocket after biotin binding. Here, we will investigate the binding of biotin to a closed streptavidin monomer for simplicity. The binding to the closed conformation of the tetramer has been estimated to have a free energy of $-26.6\,\mathrm{kcal\,mol}^{-1}$ [85].

4.2 Tutorial data

We will use the crystal structure of streptavidin (PDB code: 1STP [86]) to start our calculations. In section 4.3, we will use the biotin ligand extracted from the crystal structure. For the second part in section 4.4, we will use the whole crystal structure. Alternatively, prepared and solvated Amber topologies based on 1STP can be downloaded by running

Command-line

```
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. Additionally, the original MD trajectory files and raw GIST output files are provided at https://researchdata.uibk.ac.at//records/4mbrd-67m83.

4.3 Running GIST for Biotin

In this example, we will be running a simple GIST calculation of the biotin solute molecule to get familiar with the usage of GIST in `cpptraj`.

4.3.1 System preparation

First, extract biotin from the crystal structure and generate a topology and coordinate file compatible with Amber or a simulation package of your choice. The solute molecule should be solvated with a sufficiently large solvent box. For periodic boundary conditions, the size of the solvent box should be large enough for the solvent to behave bulk-like so as to reduce artifacts from solvent-mediated interactions between the periodic images. This corresponds to a distance between solute and box boundary of at least 12 Å for water [4, 5] though we recommend 15 Å or higher. Carefully check the employed force field, protonation state and partial charges. A good starting point for the parametrization and solvation of small molecules is the Amber tutorial 4b (http://ambermd.org/tutorials/basic/tutorial4b/).

4.3.2 Equilibration

Equilibrate your solvated biotin in an NVT or NPT ensemble. Even when running the simulation in NVT, it is advisable to include an NPT equilibration step to adjust the box size beforehand. Use enough equilibration time to allow the solvent to relax. A well-tested equilibration protocol can be found in the literature [87].

4.3.3 MD simulation

The next step is to run an MD simulation based on the equilibrated structure of biotin to generate solvent configurations for subsequent GIST analysis. The density expansion in IST relies upon the Percus source particle [88] which has a rigid solute. Motion of the solute has the effect of smoothing out the water density distributions leading to overestimates of the solvent entropy. The more solute motion, the greater the overestimate. The MD simulations can be run with constraints that prevent the motion of the solute. In practice, however, the solute heavy atoms can be restrained so that the solute does not move significantly during the MD simulation. To minimize the entropy overestimate, we recommend that RMSD of the solute during simulations be 0.1 Å or lower. In AMBER, the harmonic restraint potential takes the form $U = k_r \cdot (r - r_0)^2$ for each cartesian coordinate of each re-

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

strained atom, where r_0 is its reference position. The force constant k_r should be chosen strong enough to prevent significant deviation. Here, we apply a harmonic restraint of 100 kcal mol^{-1} Å^{-2} to all heavy atoms. We recommend at least 2.5 kcal mol^{-1} Å^{-2} restraints.

The longer the simulations, the better the convergence of GIST energy and entropy estimates with the entropy values converging more slowly. An analysis of the convergence of the entropy in GIST can be found in Ramsey et al. [6]. High precision energy and entropy values are necessary for calculating integrated GIST quantities (see Section 4.4.5) as small errors on a per-molecule basis can lead to large errors when summing over hundreds or thousands of water molecules. For qualitative mapping of solvation thermodynamic quantities, however, significantly shorter simulations generally suffice. For integrated quantities, we recommend 100 ns trajectories and, for mapping, trajectories of 5 ns to 10 ns are generally sufficient. We note that storing solvent configurations too frequently from MD trajectories can lead to systematic bias in the nearest neighbors entropy estimates (see Huggins [89]). We recommend that configurations be stored no more frequently than every 2 ps. An example AMBER input file for GIST might look like this:

Amber input

```
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,
/
```

For a full explanation of the input parameters, please refer to the Amber manual [59]. Note the restraintmask, which specifies that all atoms except for hydrogens and water should be restrained, i.e. all heavy atoms of the solute. This file will run a 100 ns NPT MD simulation while keeping all atoms except for hydrogens and waters restrained. Note that these are very standard MD settings except for the restraints. Run the MD using `pmemd.cuda`:

Command-line

```
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
```

4.3.4 GIST analysis

To start our GIST analysis of biotin, we will first have to decide on the parameters defining the GIST grid during the analysis. We will need a solute structure without water for the analysis of our future GIST results. Since we ran our MD simulation with periodic boundaries, we always need to re-image all molecules to the original box using cpptraj's `autoimage` function. We also recommend that you center the biotin solute molecule to the coordinate origin, as this simplifies the definition of our grid origin later on:

Cpptraj

```
# Load the topology ('parm') and coordinates of the first frame
parm solvated.parm7
trajin md-01.nc 1 1 1
# Reimage and center the solute (not water residues) to the origin
autoimage !(:WAT) origin
# Strip all water molecules and write out a pdb file
strip :WAT
trajout biotin-centered.pdb
run
```

In this code snippet, the first two lines load the topology and coordinates of the first frame of our MD simulation. In the third line, we re-image the coordinates to the first box and center the biotin at the origin. Then, we strip all water solvent molecules in the fourth line and write out a pdb file in the fifth line. Finally, we start the analysis with the "run" command.

To decide on a grid size, find the extent (minimum and maximum) of the x-, y-, and z-coordinates of the molecule. To get bulk-like water properties at the grid border, a buffer distance of 15 Å is reasonable for biotin. The grid parameters can be directly calculated using `cpptraj's bound` command:

Cpptraj

```
# Load the topology ('parm') and coordinates of the first frame
parm solvated.parm7
trajin md-01.nc 1 1 1
# Reimage and center the solute (not water residues) to the origin
autoimage !(:WAT) origin
# Calculate the grid bounds with a buffer of 15Å
# and a voxel size of 0.5Å
bounds !(:WAT) dx 0.5 offset 30 name Grid out bounds.dat
run
```

First, we load the topology and coordinates of the first frame of our MD simulation (1 1 1 meaning start=1, stop=1, and stride=1). Then, we re-image the coordinates and center the solute at the origin. Here, dx 0.5 sets a voxel spacing of 0.5 Å for the boundary determination, and offset 30 adds a buffer of 30 voxels (i.e., 30 × 0.5 = 15) to each dimension. The atom mask !(:WAT) specifies that the grid should be calculated around all atoms that are not water residues. Note that the offset is given in terms of the number of additional voxels, not in length dimensions. By varying the atom mask in the `bound` command, it is possible to center the grid on a region of interest, similar to the `findcentroid.py` script.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

Now, you can run the actual GIST analysis. Re-image and center the solute molecule as above, to verify that its position matches the grid parameters you decided on. We recommend using either the GPU implementation or MPI-accelerated PME implementation of GIST. While the GPU version is faster when limited CPU resources are available, the PME implementation matches the energy of the Amber MD engine more closely. The MPI implementation [90] scales well over multiple CPU cores and can be combined with either the GPU or PME implementations for maximum efficiency. A speed comparison of the various parallelization implementations is shown in [90]. The reason for the many different implementations is that GIST is a very computationally expensive calculation, and historically, the CPU version was the only one available. The GPU version improved the speed of GIST calculations significantly, allowing for grids encompassing whole biomolecules for the first time. The PME version came after with the intention to match the energy of the AMBER MD engine more closely to compare with other free energy methods. Most recently, the MPI parallelization provided another significant speed up for all implementations.

Cpptraj

```
# Load the topology ('parm') and coordinates of the full trajectory
parm solvated.parm7
trajin md-01.nc
# Reimage and center the solute (not water residues) to the origin
autoimage !(:WAT) origin
# Run GIST with a grid centered at the origin
# and a grid size of 100x100x100 voxels with 0.5\AA spacing
# The 'pme' option enables PME-GIST calculations.
gist griddim 100 100 100 out gist.dat \
refdens 0.03287 pme
```

4.3.5 Working with the GIST results

Running the GIST analysis above generates an output data file (gist.dat) containing all calculated quantities for each voxel in the grid. Additionally as outlined in section 3, various quantities are written to OpenDX files. To interact with your GIST data, we recommend the gisttools library for python3. Open the GIST output with gisttools:

Python

```
from gisttools.gist import load_gist_file
import matplotlib.pyplot as plt
# Load the GIST data, setting the reference water-water energy
gist = load_gist_file(
'gist.dat', struct='solute-centered.pdb',
eww_ref=-9.539)
# Print the number of frames and the reference density
print(f'n_frames = {gist.n_frames}')
print(f'rho0 = {gist.rho0}')
```

The gist object now provides access to all information from the output file. If the number of frames of the simulation (n_frames) and the reference density (rho0) are not provided, they are automatically detected from the GIST data. We will discuss a way to similarly detect the reference energy of the solvent-solvent energy Eww in the advanced section.

The GIST data can be accessed and modified from the gist object in various ways. Gisttools provides access to single or multiple voxels through a `.loc` property similar to the pandas library. GIST quantity data for all voxels can be accessed by directly indexing the gist object, either by providing a single name or a list of names. Indexing with `.loc` works by providing a voxel index (either an index number or list of numbers) and optionally a column index (column name or list of names). Indices start from 0 and indexing will return a single entry, pandas series or pandas dataframe depending on the query:

Python

```
# GIST objects contain pandas dataframes
gist.loc[0, 'population'] # single value
gist.loc[0] # Series
gist.loc[[0,1], ['x', 'y', 'z']] # DataFrame
gist['dTSsix_norm'].head() # Series
gist[['Eww_norm', 'Esw_norm']].head() # DataFrame
```

Gisttools also calculates the total energy per voxel (E_{all}) as a sum of E_{ww} and E_{sw}, and the free energy of solvation (A) by subtracting $T\Delta S_{six}$ from E_{all}.
Information about the grid is stored in the grid property of the gist object and structural data (supplied as a pdb of the solute molecule) is stored as an `mdtraj` trajectory in the struct property:

Python

```
gist.grid # Grid object
gist.struct # mdtraj.Trajectory object
```

Gisttools also provides OpenDX writing capabilities to visualize the calculated or modified GIST data. For visualization, you'll most likely want to write out the density-normalized GIST quantities (X_dens):

Python

```
# Write the column A_dens (the free solvation energy) to a dx file
gist.save_dx('A_dens','A.dx')
```

Section 6 discusses the visualization of OpenDX files.

4.4 Applying GIST for the Streptavidin-Biotin complex

In this section we will apply GIST to the full complex and discuss each step of the process in more detail to outline potential pitfalls. We will afterward introduce advanced analysis workflows which allow you to modify and better visualize the resulting data.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

4.4.1 System Preparation

If you want to skip this section, you can run

Command-line

```
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.

The choice of initial structure is crucial to the GIST results. When starting from an experimental structure, care has to be taken during the preparation step. Make sure to choose a proper protonation state for all titratable moieties. You will need to solvate your structure in a solvent box, which should be large enough to obtain unbiased long-range electrostatics and avoid solvent-mediated interactions of periodic images. A good criterion for validating the box size is to check whether the solvent on the outside of the box behaves bulk-like according to the GIST output. Note that, due to the restraints and finite sampling time, some solvent sites might be inaccessible to the bulk solvent. To investigate such positions, make sure that the number of explicitly placed solvent molecules in such sites matches with experiment or your best guess. For cases where the exact occupation of such sites is unknown, it may be helpful to include grand-canonical Monte Carlo (GCMC) or nonequilibrium candidate Monte Carlo (NCMC) sampling steps in the equilibration or production stage [91, 92]. It is good practice to check your restrained MD simulation or solvent populations obtained by GIST to confirm whether all relevant sites were properly solvated.

4.4.2 Equilibration

Use the `equilibration.py` script (or your own procedure) to perform short NVT and NPT equilibration runs on the complex structure. Note that the equilibration script contains unrestrained NPT steps, because restraining two molecules simultaneously might interfere with the pressure equilibration. Then, use `cpptraj` to produce biotin and streptavidin structures based on the complex by stripping the respective other molecule like this:

Cpptraj

```
# Load complex topology and coordinates after equilibration
parm complex/solvated.parm7
trajin complex/equil/EQUIL-DONE.rst
# Assuming biotin is residue 2 in the complex topology:
# Remove biotin (^2) from the complex, save topology and coordinates
strip ^2 parmout streptavidin/solvated.parm7
trajout streptavidin/pre-equil.ncrst
```

Adapt the above input file for biotin by stripping the molecule 1 instead of 2 and replacing the path names in the `strip` and `trajout` commands. Alternatively, we provide a small script named `cpptraj_remove_mol.sh` for this procedure. After that, use `equilibration.py -R` to run short, restrained equilibrations (where only the water is allowed to move) on the individual systems. By using restraints, we can ensure that the monomer conformations match the complex.

4.4.3 GIST analysis of Streptavidin-Biotin

To create input `bash` scripts for the MD simulations and GIST analysis, run the following commands in the `code` folder:

Command-line

```
make gist-md-inputs
make gist-inputs
```

After running the MD simulations as described in section 4.3.3, proceed with the GIST analysis. For easier post-processing, here we use grids that contain the whole solute molecule and the surrounding solvent. If one is only interested in solvent in the binding site, smaller grids encompassing only the binding site could be used for streptavidin and the complex. While this saves calculation time for the GIST analysis, it complicates the post-processing. Since the GIST analysis is usually less time-demanding than the MD simulation, we recommend using larger grids. Furthermore, we recommend that you center the solute molecule to the coordinate origin to simplify the analysis. Calculate the GIST grid parameters as outlined in the section 4.3 for the complex and streptavidin and run the GIST analysis.

For larger systems, like whole proteins, you can usually get away with a smaller buffer region for the GIST grid compared to small molecule solutes. To test whether your grid is large enough, look at solvent properties close to the grid edges, e.g. the solvent-solvent energies. If these correspond to the respective bulk properties, it can be assumed that your grid is large enough to capture all perturbations to the solvent caused by the solute. Note that a different box size or number of solvent molecules can skew the bulk properties slightly compared to the reference values provided above. In that case, checking for convergence of the solvent properties near the grid borders is sufficient. You might want to modify your reference values according to your results, as described in the next section.

4.4.4 Reference Values and Radial Convergence

All GIST quantities should be expressed relative to the bulk as outlined in section 7. This is automatically done by `cpptraj` for the entropy, using the bulk density specified by `refdens`. The solute-solvent energy E_{sw} is zero in neat water as there is no solute. However, the solvent-solvent energy E_{ww} needs to be referenced.

The Amber manual provides reference densities and energies for several solvents. Furthermore, a large variety of reference values calculated with GPU-GIST are listed in Table 2. However, the exact energy values are different when using

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

PME-GIST, and they also depend on the box size. For quantitative analyses, it is recommended to compute an exact reference energy. The most accurate method to compute a reference energy is to run a GIST calculation of the pure solvent using the same energy method (PME/GPU), a similar box size, at the same temperature and pressure and then compute the average solvent-solvent energy per solvent molecule.

$$E_{ww}^{ref} = \frac{\int E_{ww} \mathrm{d}x}{\langle N_{solvent} \rangle} = \frac{\sum_{voxels} E_{ww}^{dens} V_{vox}}{\sum_{voxels} N_{solvent}/N_{frames}} \quad (4)$$

Alternatively, the solvent-solvent energy in a large GIST grid converges to the bulk value at sufficient distance to the solute molecule. Therefore, the reference energy can be obtained by binning the voxels by their distance to the solute molecule, and evaluate Equation 4 within each bin. If this value converges with increasing distance from the solute, it can be used as $E_{ww,ref}^{ref}$.

`gisttools` contains functions to facilitate this analysis. There is also a method `detect_reference_value` that automatically tries to find the converged value. Although it has a simple convergence check built-in, it is always recommended to check the convergence by hand. A handy way to check this is to calculate the radial distribution of this GIST property, i.e. a histogram in relation to the distance to the solute. Because of the conceptual similarity to classical radial distribution functions, describing the density of distances between two particle types, this function is exposed to the user as the `rdf` method.

Python

```
from gisttools.gist import load_gist_file
import matplotlib.pyplot as plt
# Load the GIST data without Eww set
gist = load_gist_file(
    'gist.dat', struct='solute-centered.pdb')
# Run an rdf calculation to show the unref. Eww
bins, eww = gist.rdf(
    'Eww_unref_norm', bins=100, rmax=20.,
    normalize='norm')
plt.plot(bins, eww)
# Run the Eww autodetection
eww_auto = gist.detect_reference_value()
plt.axhline(eww_auto, color='k', linestyle='--')
plt.gca().set(
    xlabel='Distance [$\AA$]',
        ylabel='Avg. E$_{ww}$ per molecule [kcal/mol]',
        xlim=(1, 15))
plt.show()
```

The expected output of this code is shown in Figure 2. 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`:

Python

```
gist.eww_ref = gist.detect_reference_value()
```

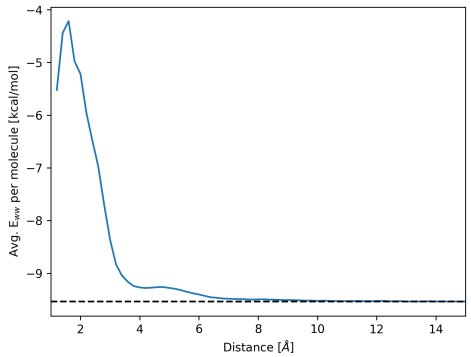


Figure 2. Convergence of E_{ww} in the complex calculation with increasing distance to the solute molecule (streptavidin-biotin). The horizontal line shows the automatically computed reference energy.

Now, sum all free energy contributions to obtain the spatially resolved ΔA_{solv}. In `gisttools`, this is done automatically and can be accessed using the `A_dens` and `A_norm` data rows.

$$\Delta A_{solv}(\mathbf{r}) = \Delta E_{ww}(\mathbf{r}) + \Delta E_{sw}(\mathbf{r}) - T\Delta S^{six}(\mathbf{r}) \quad (5)$$

After that, check whether your free energy contributions become negligible at large distances from the solute molecule. In a plot like Figure 2 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 molecule, to check the convergence (see Figure 3). If the curve flattens out, the converged value is your final ΔA_{solv}. If the curve diverges even for large integration distances, it depends on which of the contributing parts diverges.

Python

```
from gisttools.gist import load_gist_file
import matplotlib.pyplot as plt
import numpy as np
# adapt eww_ref according to the used solvent model!
gist = load_gist_file('gist.dat',
       struct='solute-centered.pdb', eww_ref=-9.5398)
# Calculate rdfs for multiple columns
bins, (da, esw, eww, s) = gist.multiple_rdfs(
       ['A_dens', 'Esw_dens', 'Eww_dens', 'dTSsix_dens'],
       bins=100, rmax=20., normalize='none')
# Plot the cumulative sum of the free energy contributions
# Note the sign change for the entropy (plots -TdS instead of TdS)
plt.plot(bins, np.cumsum(eww), label=r'$\Delta$E$_{ww}$')
plt.plot(bins, np.cumsum(esw), label=r'$\Delta$E$_{sw}$')
plt.plot(bins, np.cumsum(da), label=r'$\Delta$A')
plt.plot(bins, np.cumsum(-s), label=r'-T$\Delta$S')
plt.legend()
plt.xlabel(r"Distance Cutoff [$\AA$]")
plt.ylabel("Free Energy Contribution [kcal/mol]")
```

The expected output of this code is shown in Figure 3. Here we see two effects: First, that the solute-solvent energy E_{sw} converges only slowly, due to the charged nature of biotin.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

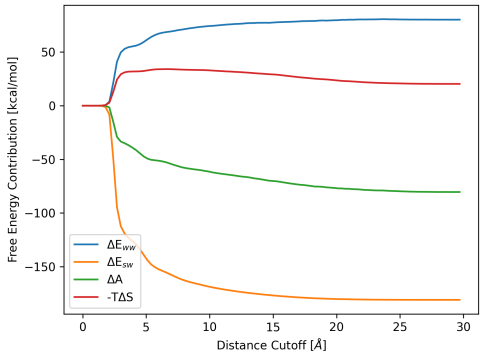


Figure 3. Convergence of ΔA_{solv} and its contributions with increasing distance to the solute molecule. All quantities are cumulative, i.e., summed up to the respective radius. Computed from the biotin calculation.

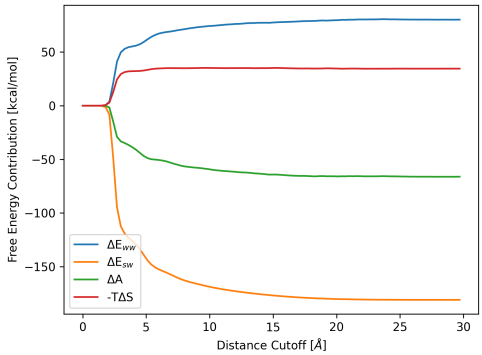


Figure 4. Convergence of ΔA_{solv} and its contributions with increasing distance to the solute molecule. The entropy was referenced, so it converges to zero for voxels far away from the solute.

Second, that the solvent entropy does not converge to a fixed value, even at large distances. Most often, this occurs due to a systematic bias in the nearest neighbor entropy calculation, which converges to the correct value from above [4, 89]. This can be fixed by either calculating the bias per water molecule (as was done in [4]) or referencing the entropy in the same manner as the $E_{ww,norm}$:

Python

```
def reference_entropy(gf):
'''
Reference the entropy to zero at large distances.
'''
if 'dTSsix_unref_norm' not in gf.data.columns:
       gf['dTSsix_unref_norm'] = gf['dTSsix_norm']
       gf['dTSsix_unref_dens'] = gf['dTSsix_dens']
refval = gf.detect_reference_value('dTSsix_unref_dens')
gf['dTSsix_norm'] = gf.get_referenced('dTSsix_unref_norm', refval)
gf['dTSsix_dens'] = gf.get_referenced('dTSsix_unref_dens', refval)

reference_entropy(biotin)
```

When we recreate the plot from Figure 3 with the referenced entropy in Figure 4, we see that the divergence is removed.

While E_{ww} converges well here when using the tabulated $E_{ww,norm}$, if it diverges, one might need to tweak the E_{ww} reference by either using the automatic referencing, using a different tabulated $E_{ww,norm}$ value or running a pure bulk solvent reference calculation. The most common cause of strong divergence of E_{ww} is that the reference value is set to a value of a different solvent model.

4.4.5 Visualizing the Solvation Free Energy (ΔA_{solv})

Next, we visualize the energetic contributions to the free energy of solvation. You can use PyMOL[93] or VMD[94] to visualize the .dx files from GIST. 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 solvent molecule at each grid voxel. This quantity is given by $2E_{ww}+E_{sw}$. The energy referencing leads to non-zero 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:

Python

```
import gisttools as gt
# Load the GIST data, make sure to set the correct reference energy
gist = gt.gist.load_gist_file('gist.dat',
    struct='solute-centered.pdb', eww_ref=-9.5398)
# Calculate the total energy without double counting
gist['E_norm'] = gist['Eww_norm'] * 2 + gist['Esw_norm']
# Set the energy to zero for all voxels with no population
gist.loc[gist['population'] == 0, 'E_norm'] = 0
# Save the new energy to a dx file
gist.save_dx('E_norm', 'gist-E-per-mol-norm.dx')
```

The OpenDX files can then be imported into your molecular viewer of choice. A visualization of energetic contributions to the free energy of solvation of the streptavidin binding pocket is shown in Figure 5. The streptavidin binding pocket is largely hydrophobic, leading to energetically unfavorable water in the middle and right of the cavity. The green regions on the left of the cavity are where biotin forms hydrogen bond interactions with the protein (green). Note that this is "norm" data, so not density-weighted but rather comparing voxels on a per-molecule basis. The surfaces are therefore disjointed since even voxels seldomly sampled by water are shown. PyMOL input scripts to generate such visualizations are available in the GitHub repository for this tutorial.

4.4.6 Contribution of Hydration to Binding

In the next step, we calculate the ΔA_{solv} contributions around biotin in each system (biotin, streptavidin, and complex). Note that our integration region does not fully reach into bulk, and some of the boundary regions are close to strep-

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

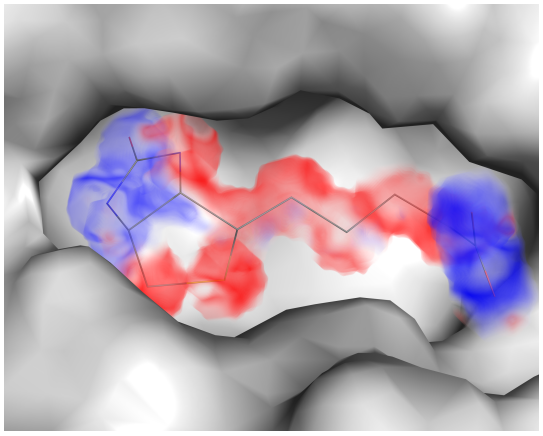


Figure 5. Water energy using "norm" data represented as volume in the streptavidin binding pocket overlaid with biotin. Red: high solvent energy (from +3.5 to +2.5 kcal/mol/molecule). Blue: low solvent energy (from -3.5 to -2.5 kcal/mol/molecule). The energy density is based on the apo structure without double-counting and is shown only within 1 Å of biotin.

tavidin. 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 streptavidin and use the shifted biotin coordinates to define the integration region. We recommend using a `Jupyter Notebook` for the following analyses. Load the files and double-check the frame numbers and reference density.

Python

```
import numpy as np
from gisttools.gist import load_gist_file
import matplotlib.pyplot as plt
# Load the GIST data, setting no reference energy
# since we will calculate it later
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)
```

Assign reference energies and check them for plausibility.

Python

```
# Set the reference density for the solvent by automatic detection
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.

Python

```
reference_entropy(biotin)
reference_entropy(strept)
reference_entropy(compl)
```

Next, compute the atom positions that define the integration region. For the streptavidin integral, we align the complex structure to streptavidin and then use the biotin positions as centers. Note that we use `mdtraj` here, since `gisttools` stores the reference structure as an `mdtraj Trajectory` object. Here, we compute the histogram of the contributions of single properties to assess the convergence.

Python

```
import mdtraj as md
col = 'dTSsix_dens'
def select(traj, sel):
    # Slice a Trajectory by selection mask.
    return traj.atom_slice(traj.top.select(sel))
# Define atom masks in mdtraj syntax for biotin and streptavidin
biotin_mask = 'resname BTN and not type H'
strept_mask = 'not resname BTN and not resname WAT and not type H'
# select the biotin atoms, we multiply by 10 to convert nm to Å.
compl_x = select(compl.struct, biotin_mask).xyz[0] * 10.
biotin_x = select(biotin.struct, biotin_mask).xyz[0] * 10.
# align the complex to streptavidin
# and select the streptavidin atoms
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.

# Now we can calculate the radial distribution functions
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 histograms from the complex, and compute the sum of your property within some cutoff distance to the solute. If you visualize the individual histograms, you will notice that the difference converges much better with increasing radius than the single contributions.

Python

```
# Endstate approach: complex - monomers
difference = compl_rdf - biotin_rdf - strept_rdf
# Set integration cutoff
cutoff = 12
# Integrate the difference
integral = difference[bins < cutoff].sum()
print('Integral = {}'.format(integral))
# Plot the cumulative sum of the difference
plt.plot(bins, np.cumsum(difference))
plt.axvline(cutoff)
plt.xlabel('distance to biotin [Å]')
plt.ylabel('$\Delta A$ contribution [kcal/mol]')
```


https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

Finally, you can also plot the difference between the complex and monomer contributions against the distance to biotin, using the complex coordinates for the holo structure:

Python

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4))
cutoff = 12
# In the first plot, we show the cumulative sum of the contributions
ax1.plot(bins, np.cumsum(biotin_rdf), label='biotin')
ax1.plot(
    bins, np.cumsum(strept_rdf), label='streptavidin')
ax1.plot(bins, np.cumsum(compl_rdf), label='complex')
ax1.legend()
ax1.axvline(cutoff, color='k', linestyle='--')
ax1.set_xlabel(r'Distance to Biotin [Å]')
ax1.set_ylabel(r'$\Delta A$ contributions [Å]')
ax1.grid()

# In the second plot, we show the difference of the cumulative sums
difference = compl_rdf - biotin_rdf - strept_rdf
ax2.plot(bins, np.cumsum(difference))
ax2.axvline(cutoff, color='k', linestyle='--')
ax2.set_xlabel(r'Distance to Biotin [Å]')
ax2.set_ylabel(r'$\Delta \Delta A$ [kcal/mol]')
ax2.grid()
plt.show()
```

The expected result is shown in Figure 6.

Next, compute the energy (`Eall`) and entropy (`dTSsix`) contributions separately. To make this easier, gisttools includes a function to integrate in a radius around atom centers:

Python

```
import pandas as pd
# Integrate the energy and entropy contributions
centers = {'biotin': biotin_x,
    'strept': strept_x,
    'compl': compl_x}
gist_objs = {
        'compl': compl,
        'strept': strept,
        'biotin': biotin,
    }
cols = ['Eall_dens','dTSsix_dens']
rmax = 12
results = {
    name: gist.integrate_around(
    cols, rmax, centers[name])
    for name, gist in gist_objs.items()}
print(pd.DataFrame(results).T)
```

You will find that the energy disfavors binding; we do not yet include the interaction energy between biotin and streptavidin. Using the `energy` command in `cpptraj`, you can compute this interaction energy. We recommend using PME in combination with PME-GIST, but not with GPU-GIST to be in line with the different ways the energy is calculated in the two GIST methods.

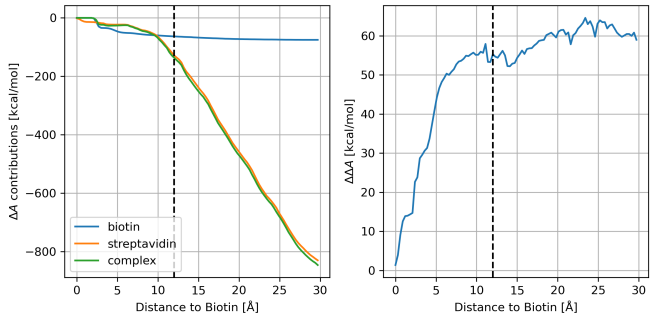


Figure 6. Left: ΔA_{solv} contributions around the location of biotin in the binding pocket, evaluated in the biotin, streptavidin, and complex systems. Each line represents a cumulative sum plotted against the distance to biotin. Right: The hydration contribution to binding, evaluated as the difference between the lines in the left panel. The vertical dashed line is at 12 Å and represents the chosen integration cutoff.

An example `cpptraj` is then:

Cpptraj

```
# Load the topology and trajectory of the complex
parm solvated.parm7
trajin md-01.nc 1 last 100
# Calculate the interaction energy of the complex (^1,2)
# and the monomers (^1 for streptavidin, ^2 for biotin)
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 [4, 5] that the solvation entropy in water is best approximated by 0.6 times the first order entropy provided by GIST. So we also compute the scaled entropy and check the effect on the binding affinity. The expected results are summarized in Table 3.

Table 3. Free energy contributions for the monomers and the dimer in kcal·mol^{-1}. "Diff" is calculated as $E^{\text{internal}} + \Delta E^{\text{GIST}} - T\Delta S^{\text{scaled}}$. "total" is calculated as "complex"-"streptavidin" - "biotin".

System	$E^{internal}$	ΔE^{GIST}	$T\Delta S^{GIST}$	$T\Delta S^{scaled}$	Diff
complex	-1303.4	-361.2	-233.1	-139.9	-1524.7
streptavidin	-1180.7	-360.8	-242.4	-145.4	-1396.1
biotin	-24.0	-98.1	-35.1	-21.1	-101.0
total	-98.7	97.7	44.4	26.6	-27.6

Although streptavidin-biotin is known to be a very stable complex, the free solvation energy ($\Delta E^{GIST} - T\Delta S^{scaled}$) favors the dissociation. This is expected: the binding of biotin to

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

streptavidin is facilitated through numerous H-bonds and over polar interactions. As such, water favorably solvates both biotin and streptavidin and an energetic penalty is paid when displacing the water during the binding process. In this case, we find that the changes in the energy of solvation and of the internal energy $\Delta E^{internal}$ more or less cancel out, indicating that binding is largely entropy-driven. At first glance, this is surprising, since isothermal titration calorimetry (ITC) of biotin-streptavidin shows significant enthalpic binding contributions [95, 96]. However, we do not take the conformational transition of the streptavidin binding pocket into account.

Prior computational studies suggest a ΔG of -26.6 kcal mol^{-1} for the binding of biotin into the closed conformation of streptavidin [85]. This indicates that our result is in good agreement with literature. To improve the agreement with ITC measurements, the conformational changes of the binding pocket should be included.

4.4.7 Further steps

In this tutorial, we obtained a value for the binding of biotin into the closed conformation of the streptavidin binding pocket. Investigating the effect of different conformations on the binding affinity can be achieved by performing molecular dynamics simulations of the complex and/or monomers, followed by GIST analysis on multiple cluster representatives. Including the effect of lid closing in the binding process may require free energy calculation methods such as umbrella sampling. For faster calculations, consider using smaller GIST grids focused on the binding pocket, or rotating the solute molecule along its principal axes to fit the cuboid grid more exactly; note that such adjustments should be made *before* the MD simulation, as rotating the trajectory compromises the periodic box information.

5 Recent developments

In recent years, several updates to the original GIST implementation have been published. The basic functionality of the program, however, has not been changed. In this chapter, we will shortly present each of those updates.

5.1 GPU implementation

In reference [14], a GPU implementation of the energy calculation was presented, which typically increases the speed by 1–2 orders of magnitude. If `cpptraj` is compiled with GPU support, the energy calculation uses the GPU automatically, without any additional input. However, PME-GIST is not supported on the GPU, and no GPU code will be used when specifying `pme`.

5.2 PME implementation

In reference [4], a PME implementation of the GIST energy was presented. For typical systems, this implementation is slightly slower than GPU-GIST, but much faster than the original CPU code. Furthermore, PME-GIST offers the best agreement between GIST energies and those observed in a classical MD simulation. To run PME-GIST, the `pme` flag to the `gist` command can be used in `cpptraj`. The output will contain two additional columns: `PME_dens` and `PME_norm`. They contain the potential of solute molecule and solvent evaluated at the solvent positions and divided by two. Furthermore, the `Eww` and `Esw` columns are also computed using PME, and are reported as usual. For typical use cases, we recommend using `Eww` and `Esw` over the `PME` column.

5.3 MPI parallelization

The most recent addition to GIST is MPI parallelization of both energy and entropy calculations [90]. Since the MPI parallelization is orthogonal to the other improvements, it can be used with both PME or GPU accelerated GIST.

5.4 GIST with non-water solvents

In references [5, 11, 14], an extension of GIST to solvents other than water is described. To run a GIST calculation in a solvent other than water, use the `solute` flag with a `cpptraj` selection mask to select the solute molecule (e.g. `solute :1-5` if the first five residues are the solute). All other molecules will be treated as solvent. If there are multiple solvent species, also use the `solventmols` flag as described in Section 5.5.

The code will automatically choose three atoms per solvent species to define the solvent orientation, and print this selection to the output. Sometimes, however, the automatic selection is not optimal. For instance, in methanol one can either incorporate the C–H bonds or the O–H bond in the selection. The latter is probably more relevant since it can incorporate hydrogen bonding effects. Assuming that the alcohol hydrogen atom is called H1, this could be specified as follows:

Cpptraj

```
gist gridcntr <x> <y> <z> \
griddim <Nx> <Ny> <Nz> gridspacn <val> \
solute ^1 rigidatoms O1 H1 C1
```

Here, `solute ^1` defines residue 1 as the solute. The `rigidatoms O1 H1 C1` keyword specifies atoms (e.g., in methanol solvent) to define the solvent's orientation, with O1 being the central atom.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

5.5 GIST with salt-water mixtures

In reference [3], an extension of GIST was presented that can use salt-water mixtures as a solvent. This allows treating salting-out effects, although it was shown that the salting-out coefficient is over-predicted by GIST because of the first-order entropy approximation. Generally, GIST should be able to treat arbitrary solvent mixtures as long as each solvent molecule is sufficiently rigid.

In the `cpptraj` implementation of GIST, the `solute` keyword specifies which components of the system are treated as a solute molecule. Everything else will be treated as solvent. If the solvent contains more than one molecular species, a list of solvent molecule names (i.e., "`solventmols WAT,NA,CL`") must be specified. For each solvent molecule in this list, densities (e.g., `g_mol_NA`) and energies (e.g., `Eww_mol_NA`) will be computed and written to .dx files. Note that, e.g., `Esw_mol_NA` contains the interactions of every "`NA`" solvent molecule with the solute molecule, and `Eww_mol_NA` contains the interactions among "`NA`" solvent molecules as well as their interactions with all other solvents, divided by 2 to account for double counting. The entropy will be computed using only the density of the first solvent molecule.

For instance, a GIST calculation in salt water, including the first-order water entropy, can be run as follows:

Cpptraj

```
gist gridcntr <x> <y> <z> \
griddim <Nx> <Ny> <Nz> gridspacn <val> \
solute !(:WAT,NA,CL) solventmols WAT,NA,CL
```

The current implementation of the entropy calculation in `cpptraj` requires at least 3 atoms in the main solvent. Note that the calculation of the reference value is complicated by the introduction of additional solvents. For an overview of how to correctly handle such calculations, refer to reference [3]. Additionally, to compute the first-order entropy of the ions, as well as an approximate second order entropy, Python code is available on https://github.com/liedllab/second-disorder.

6 Visualization

6.1 Visualizing DX files

A minimal PyMOL script to visualize a dx file is shown here. This loads an input structure from a file named `streptavidin.pdb` and visualizes the oxygen density at an isolevel of 2 (i.e., twice the reference density). The expected output is shown in Figure 7.

PyMOL

```
load output/streptavidin/gist.pdb, streptavidin
as surface
color gray70, streptavidin
load output/streptavidin/gist-g0.dx, g0
isomesh g0_high, g0, 2
color slate, g0_high
```

A more sophisticated visualization, including the energy, entropy, and free energy of removing a single water solvent molecule, is found in Figure 8.

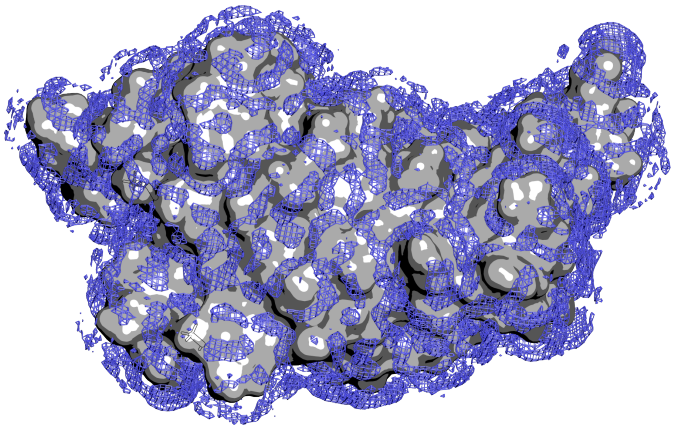
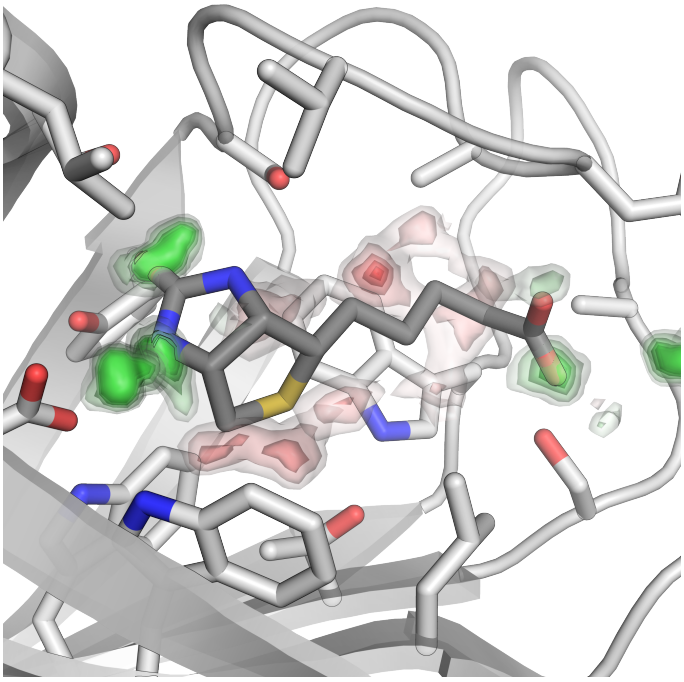
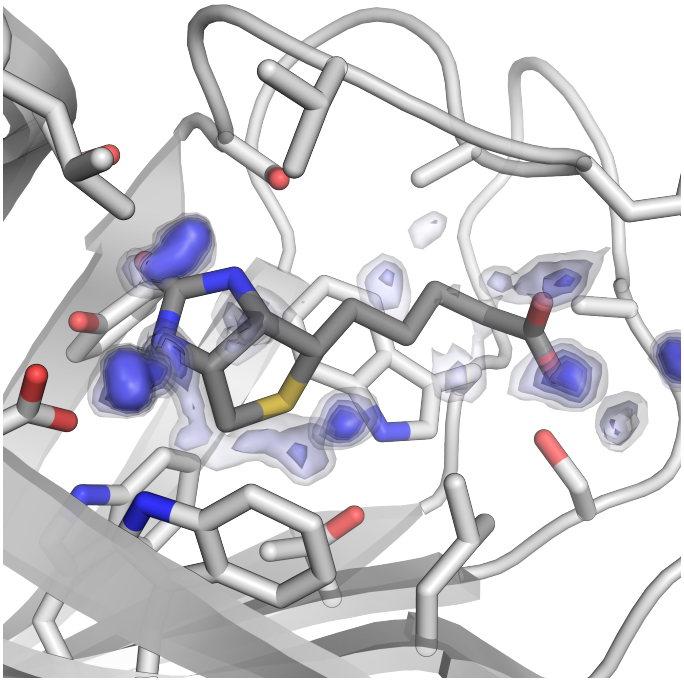


Figure 7. Oxygen density around streptavidin at an isolevel of twice the reference density (i.e., bulk)

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059



(a) GIST solvent energy (based on E_{sw} + 2E_{ww}).



(b) GIST solvent entropy

Figure 8. Biotin in the streptavidin binding pocket, with isosurfaces based on the GIST calculation on the holo structure. Isolevels shown are (±) 0.25, 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 kcal mol^{-1} Å^{-3}, except for the positive energy, where they are divided by two to visually account for its overall lower density. The negative solvent energy is shown in green, positive solvent energy in red and negative entropy in blue. Only voxels within 4 Å of biotin are shown.

6.2 Solvent Accessible Surface (SAS)

The solvent accessible surface (SAS) represents the interface between regions that are occupied by the solute molecule and those which can be accessed by the solvent. GIST provides a very natural way of creating a SAS using an isosurface of the oxygen density `g0`. At a very low isovalue, an isosurface of the water's oxygen or hydrogen shows areas the water can reach, essentially creating a SAS. Here, the SAS was generated from a simulation of water around streptavidin without biotin, leaving the binding site solvent-accessible. The expected output is shown in Figure 9.

PyMOL

```
load ../gist.pdb
load ../gist-g0.dx
isomesh sas, gist-g0, 0.1
```

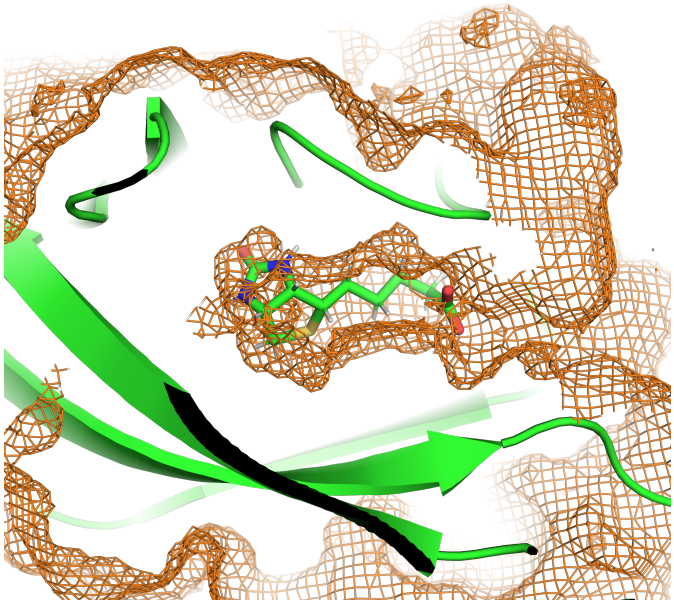


Figure 9. SAS for streptavidin focused on the binding pocket

6.3 Defining the binding pocket

Specific regions in a protein can be defined to calculate the thermodynamic properties for that region. This is especially significant when calculating the properties of water that will be expelled when a ligand binds to the protein. For example, voxels that are located within the binding pocket can be defined using a ligand solute molecule and a distance criterion. In section 4.4.6, we show how to integrate over regions such as the binding pocket. To visualize the considered region, we can create a OpenDX file where we set the considered voxels to 1 and all others to 0. Here, we show what this would look like for a radius of 3.5 Å around the heavy atoms of biotin.

Python

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

```
import gisttools as gt
def select(traj, sel):
        """
        Slice a Trajectory by selection mask.
        """
        return traj.atom_slice(traj.top.select(sel))

### Load GIST data
compl = gt.gist.load_gist_file(
        'complex/gist/gist.dat',
        struct='complex/gist/gist_nowat.pdb')
### Define ligand atoms and find voxels around them
rmax = 3.5
biotin_mask = 'resname BTN and not element H'
btn = select(compl.struct, biotin_mask).xyz[0] * 10.
ind, _, _ = compl.distance_to_spheres(btn, rmax)

### Set voxels within 3.5A of biotin to 1, write out dx file
compl.loc[ind, 'BP'] = 1
compl.save_dx('BP', 'binding_pocket.dx')
```

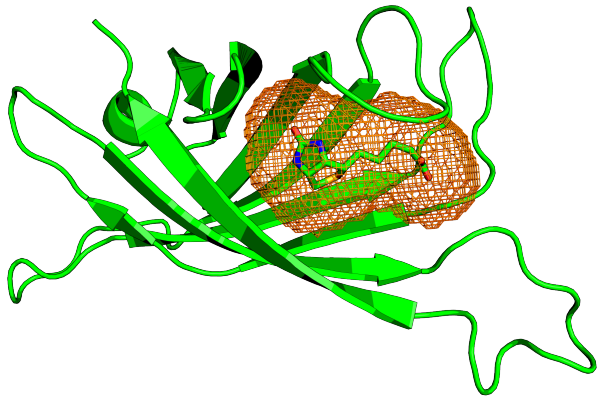


Figure 10. Volume of streptavidin defined around 3.5 Å of biotin.

7 Theory

GIST is an implementation of inhomogeneous fluid solvation theory (IST) [35] that discretizes the free energy of solvation ΔA_{solv} on a three-dimensional grid. It was first devised by Nguyen et al. [1] and its implementation in `cpptraj` was thoroughly described by Ramsey et al. [6]. Here, we only provide a short overview of the theory behind GIST. For more technical information (such as implementation details), we recommend one of the more recent publications on developments in GIST [4, 14, 90].

7.1 On Thermodynamic Quantities

The PV difference between the Helmholtz and Gibbs free energies is negligible for condensed phase systems at biological and STP conditions. Hence, in the literature, the terms ΔG_{solv} and ΔA_{solv} are often used interchangeably. The $T\Delta S$ columns in GIST output are labeled dTS for brevity.

7.2 Solvation Entropy

IST expresses the free energy of solvation in terms of the solvent density in a coordinate system defined by the solute molecule. The solute molecule is kept fixed in space, following the standard solvation process established by Ben-Naim [97].

The entropy can be approximated as an infinite expansion of correlations of increasing order. In a slightly simplified view, the first order is the log solvent density $ln\left(g_{sw}(\mathbf{r},\omega)\right)$ at each position $\mathbf{r}$ and orientation ω. The first-order entropy omits all solvent-solvent (and higher) correlations, while the solute-solvent correlation is taken into account because the coordinates are relative to the solute. Thus, it is called the solute-water entropy S_{sw}. Note that this definition is different and more direct than the one used originally by Ben-Naim [97], as recently discussed in [98]. The solvent density is expressed relative to the bulk density ρ^0. The entropy integral is also normalized by $8\pi^2$, which is the volume of the orientational space. In bulk or at a high distance to the solute molecule, the quantity $g_{sw}(\mathbf{r},\omega)$ approaches one, which leads to zero first-order entropy. Therefore, no reference entropy is needed unless there is a numeric bias, e.g., when the MD frames are statistically dependent because the time between them is insufficient.

$$\Delta S_{\text{solv}} \approx \Delta S_{sw} \equiv -R\frac{\rho^0}{8\pi^2}\int g_{sw}(\mathbf{r},\omega)\ln g_{sw}(\mathbf{r},\omega)d\mathbf{r}d\omega \quad (6)$$

7.2.1 Entropy Calculations in cpptraj

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

In the first method, the solvation entropy is broken down into

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

translational and orientational contributions.

$$\Delta S_{solv} = \Delta S_{trans} + \Delta S_{orient} \tag{7}$$

This is exact assuming that the distribution of the orientation ω is constant within a voxel k, but might converge slower than computing the entropy from the combined space (see below). The second method directly calculates the solvation entropy by evaluating the six-dimensional integral (3 for position and 3 for orientation).

A first nearest neighbor (NN) approach is used to evaluate the entropy based on the orientational and translational distributions. The distributions are computed relative to a homogeneous distribution of solvent molecules, given a bulk density ρ^0. The average entropy contributions per solvent molecule in voxel k are calculated as

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

$$S_k^{orient} = -R\left(\gamma + \frac{1}{N_k}\sum_{i=1}^{N_k} \ln g_{NN,i}(\omega)\right), \tag{9}$$

and

$$S_k^{\text{six}} = -R\left(\gamma + \frac{1}{N_k}\sum_{i=1}^{N_k} \ln g_{NN,i}(\mathbf{r},\omega)\right), \tag{10}$$

where N_k is the number of water solvent molecules found in voxel k, γ is Euler's constant accounting for the bias of the naive entropy estimator, and g_{NN} is the nearest neighbor estimate of the density.

7.3 Solvation Energy

The solvation energy is calculated from the water-water and water-solute interactions from the force field.

$$\Delta E_{solv} = \Delta E_{sw} + \Delta E_{ww} \tag{11}$$

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 molecule. In practice, E_{sw} is computed 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.

$$\begin{aligned}\Delta E_{sw}(\mathbf{r}) &\equiv \frac{1}{8\pi^2}\int g_{sw}\left(\omega\,|\,\mathbf{r}\right) U_{sw}\left(\mathbf{r},\omega\right) d\omega \\ \Delta E_{sw,k} &= \left\langle \sum_i^{solvent,k} \sum_j^{solute} U_{ij} \right\rangle\end{aligned} \tag{12}$$

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 12 is computed per voxel. Similar to the entropy integrals, the solute-water solvation energy decays with increasing distance to the solute molecule. Hence, it can be approximated by local spatial integrals. The solvent-solvent energy E_{ww} is computed similarly. It can also be expressed in terms of density functions, but is practically computed as a sum over solvent-solvent interactions per voxel k. In contrast to E_{sw}, E_{ww} does not tend to zero in bulk. Therefore, a reference corresponding to the average energy of a bulk water solvent molecule needs to be subtracted. The referenced solvent-solvent energy will be denoted as E_{ww}^{corr}.

$$\begin{aligned}\Delta E_{ww}^{\text{corr}}(\mathbf{r}) \equiv &\left(\frac{1}{8\pi^2}\right)^2 \rho^o \int g_{sw}(\omega\,|\,\mathbf{r}) \\ &\times \left[g_{sw}(\mathbf{r}',\omega') - g_{ww}^0(\mathbf{r},\omega,\mathbf{r}',\omega')\right] \\ &\times U_{ww}(\mathbf{r},\omega,\mathbf{r}',\omega') d\omega d\mathbf{r}' d\omega' \\ \Delta E_{ww,k}^{\text{corr}} = &\frac{1}{N_k}\left\langle \sum_i^{\text{solvent},k} \sum_{j\neq i}^{\text{solvent}} U_{ij} \right\rangle - \left\langle E_{ww}^{\text{bulk}} \right\rangle\end{aligned} \tag{13}$$

7.3.1 Interpreting the energy values

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:

$$\Delta E_{ww} = \sum_k^{\text{voxels}} \frac{E_{ww,k}^{\text{corr}}}{2} \tag{14}$$

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

$$\Delta E_{ww}^{\text{R}} = \sum_k^{\text{voxels in } R} E_{ww,k}^{\text{corr}} \tag{15}$$

When the region R comprises more than one solvent molecule, interactions within this region are double-counted while interactions to the outside are not. This could be solved using the solvent-solvent energy between each pair of voxels $E_{ww,k\,l}$:

$$\Delta E_{ww}^{\text{R}} = 2\left(\sum_k^{\text{voxels in R}} E_{ww,k}^{\text{corr}} - \sum_k^{\text{voxels in R}} \sum_l^{\text{l>k}} E_{ww,k\,l}\right) \tag{16}$$

While this is supported by the standard GIST implementation (using the `doeij` flag), it is rarely done due to the large size of the $E_{ww,k\,l}$ matrix. Integrating GIST values over the whole grid corresponds to a process where all the solvent is removed into bulk. However, when integrating over a small region, this process is less well-defined, since it is unclear to what extent the remaining solvent would reorganize. This depends on the local environment and on *what* the solvent would be

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

replaced by. Therefore, the effect of reorganization must be judged for each case individually. One way to treat reorganization is an end-state approach as shown in the tutorial: all states before and after solvent reorganization are considered. In the tutorial, this corresponds to the solvation of biotin and streptavidin separated (initial states) and in complex (end-state). Note that all solvent contributions must be considered to obtain thermodynamically accurate results in such a case. In practice, this therefore requires the integration over the whole grid or at least all voxels with solvent properties different from bulk (i.e. setting a large cutoff).

7.3.2 PME energy

In the original version of GIST, the energies are calculated based on the minimum image convention. In PME-GIST, the electrostatic energy E_{elec} is calculated using the particle mesh Ewald (PME) method, which yields energies that are highly consistent with the Amber MD engine [4]. The Lennard-Jones part E_{lj} is computed separately in direct space.

$$E_{\mathrm{total}} = E_{\mathrm{elec}} + E_{\mathrm{lj}} \tag{17}$$

During the electrostatics calculation, the system is treated as periodic and the 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_{self} (called E_{corr} in the original publication [4]), which corrects for the self-interaction of each solvent molecule in the reciprocal term.

$$E_{elec} = E_{\mathrm{dir}} + E_{\mathrm{rec}} + E_{\mathrm{self}} \tag{18}$$

The short-range Lennard-Jones contribution is computed in the direct space using a distance cutoff. Furthermore, a long-range correction term is computed that accounts for the contributions above this cutoff assuming a homogeneous distribution of particles.

$$E_{\mathrm{lj}} = E_{\mathrm{lj,\,short}} + E_{\mathrm{lj,\,corr}} \tag{19}$$

For a more detailed description of the PME-GIST implementation and the pairwise decomposition of the PME energy, we refer to the original publication [4].

8 Checklists, Cheat Sheets and Flowcharts

SIMULATION SETTINGS

Suggested values

- ☐ Simulation time: 100 ns
- ☐ Number of analyzed frames: 100 000
- ☐ Restraints: 2.5 kcal mol^{-1} Å^{-2} or greater on solute heavy atoms.
- ☐ Time between stored water configurations: 2 ps

CHOOSING AN ENERGY METHOD

- ☐ CPU: slow, most general (can write out E_{ij} matrices)
- ☐ CPU, PME: highly consistent with Amber MD, fast
- ☐ GPU, direct space: fastest for a single CPU core

All methods profit from MPI-parallelization. PME with multiple CPU cores (>4) is usually the fastest method.

OBTAINING ABSOLUTE ΔA_{solv}

- ☐ Check the radial convergence (see Fig. 3)
- ☐ 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

HANDLING CONVERGENCE PROBLEMS

- ☐ Insufficient sampling leads to noise in the results. Run longer simulations.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

THE CPPTRAJ GIST ACTION

Options

Various flags and options can be provided when running a GIST calculation in cpptraj. A list of possible and required options is provided here:

I/O Options	
name <dataset name>	Name for output datasets in cpptraj.
prefix <prefix>	Output file name prefix. Default is 'gist'.
ext <extension>	Output grid file name extension. Default is '.dx'.
out <file>	Name of the main GIST output file. If not specified set to '<prefix>-output.dat'.
info <file>	Name of main GIST info file. If not specified info is written to standard output.
floatfmt <fmt>	Format for floating point values in GIST output file. Options are 'double', 'scientific' or 'general'. Default chooses 'fixed' or 'scientific' automatically.
floatwidth <val>	Width of floating point values in GIST output file. Default is no width restriction.
floatprec <val>	Precision of floating point values in GIST output file. Default is the system default.
intwidth <val>	Width of integer point values in GIST output file. Default is no width restriction.
doeij	Output the triangular matrix representing the water-water interactions between pairs of voxels. Not supported with PME or GPU GIST.
Grid Options	
gridcntr <xval> <yval> <zval>	Coordinates in Å of the center of the grid. Default is 0.0 0.0 0.0
griddim <xval> <yval> <zval>	Grid dimensions in voxels along each coordinate axis. Default is 40 40 40.
gridspacn <val>	Grid spacing (linear dimension of each voxel) in Angstroms. Default is 0.5 Å.
rmsfit <mask>	If specified, grid will be rotated and translated to follow the atoms selected by mask.
GIST Calculation Options	
skipE	Skip all the energy calculations (cannot be specified with 'doeij')
skipS	Skip all the entropy calculations.
refdens <val>	Reference density of bulk solvent, used in computing g_O,g_H, and the translational entropy. Default is 0.0334 molecules/Å^3.
temp <val>	Temperature of the input trajectory.
noimage	Disable distance imaging in energy calculation.
neighborcut <val>	Cutoff in Å for determining solvent neighbors. Default is 3.5 Å, typical for water O-O.
oldnnvolume	Use the old reference volume for the nearest neighbor entropy.
nnsearchlayers <val>	Layers of neighboring voxels considered in nearest neighbor search. Higher values may improve entropy convergence for little sampling or fine grid spacings, but increase the calculation time. Default is 1.
PME Options	
nopme	Do not use particle mesh Ewald for the non-bonded calculation. This is set on default.
pme	Use particle mesh Ewald for the non-bonded electrostatics calculation. The van der Waals energy will be calculated using a long-range correction for periodicity.
cut <val>	Direct space cutoff for pme. Default is 8.0 Å.
dsumtol <val>	Direct sum tolerance used to determine Ewald coefficient. Default is 0.00001.
ewcoeff <val>	Ewald coefficient in 1/Å.
erfcdx <val>	Spacing to use for the ERFC splines. Default is 0.0002 Å.
skinnb	Used to determine pairlist atoms (added to cut, so pairlist cutoff is cut + skinnb); included in order to maintain consistency with results from sander.
ljswidth <val>	If specified, use a force-switching form for the Lennard-Jones calculation from <cutoff>-<val> to <cutoff>
order <val>	Spline order for charges.
nfft <nfft1>,<nfft2>,<nfft3>	Explicitly set the number of FFT grid points in each dimension. Will be determined automatically if not specified.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

THE CPPTRAJ GIST ACTION

Further Options

Solvent Options

solute <mask>	Selection mask for the solute. All other molecules will be solvent. If this is omitted, the standard solute/solvent assignment will be used.
solventmols <mols>	Comma-separated list of solvent molecules residue names. Energies will be computed per solvent molecule. For the entropy, only the first solvent will be used. Use this for calulations with more than one solvent of interest, e.g. for ions.
rigidatoms <c> <a1> <a2>	Specifies the molecular orientation for the entropy calculation from a central atom and two additional atoms, e.g. O H1 H2 for water. By default, a simple heuristic will be used. Use this option if the automatically chosen atoms are collinear or do not represent the orientation well.
nocom	Do not use the center of mass to define the molecular position. Instead, use the first atom in rigidatoms. Use this flag to restore the behavior of old GIST runs.

Order Calculation Options

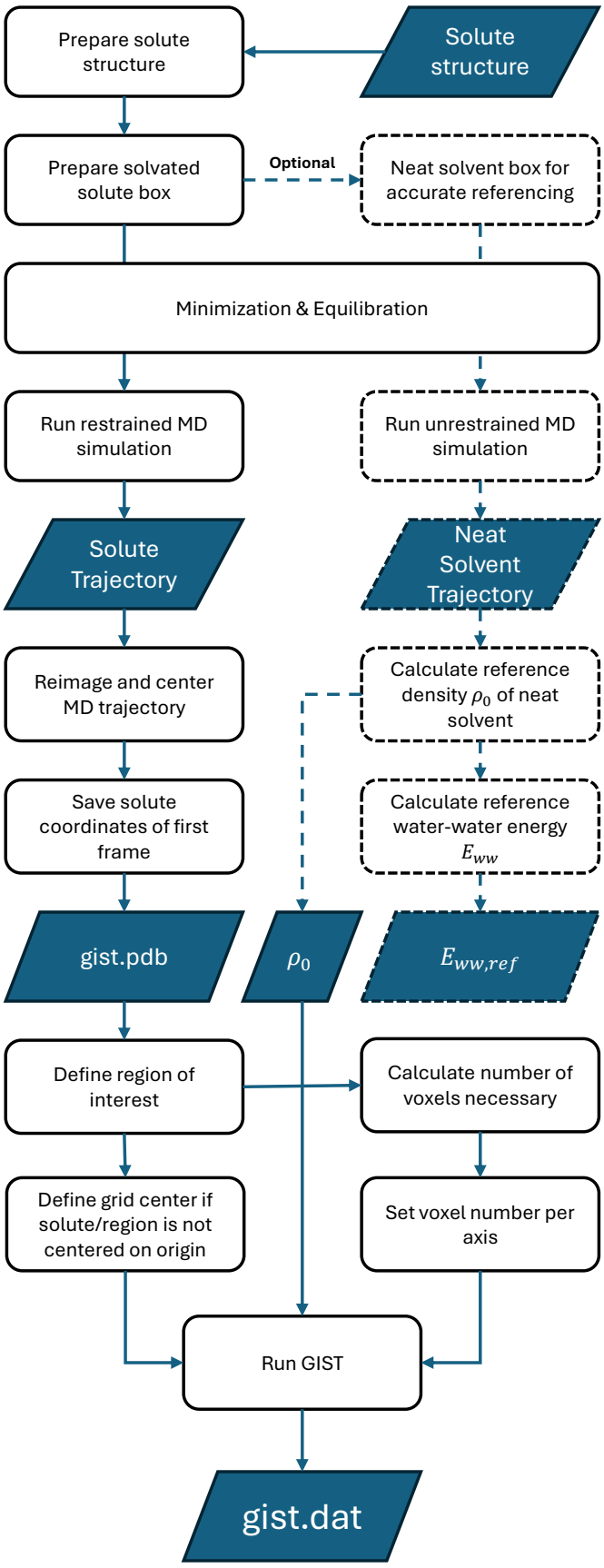
doorder	Calculate the water order parameter [reference] for each voxel
nopl	Do not use pair list for order calculation
plcut <val>	Pair list cutoff for order calculation. Default is 10 Å

Output

GIST calculations write a variety of data sorted by voxel into an output file specified by the 'out' keyword. Some of the output data is also automatically written to 'open data explorer' (.dx) files for convenient visualization in software such as VMD or PyMOL. Note that some columns will be written out both with '_norm' and '_dens' suffixes, referring to normalization by solvent molecule or voxel volume respectively. The following columns can be found in the output file:

Name	Keyword	Description
index		Voxel indices
x, y, z		Coordinates of the voxel centers
pop		Population of solvent in voxel over entire simulation
gX		For each element in the main solvent, the number density of atoms found in the voxel, in units of the bulk density. If the same element occurs multiple times, the bulk density is scaled accordingly.
g_mol_Y	[solventmols]	Density of every solvent species Y. Scaled by rho0.
Esw		Mean solute-solvent interaction energy.
Eww		Mean solvent-solvent interaction energy.
Esw_mol_Y	[solventmols]	Mean solute-solvent interaction energy for solvent species Y.
Eww_mol_Y	[solventmols]	Mean solvent-solvent interaction energy for solvent species Y.
PME	[pme]	Solvent PME energy.
U_PME	[pme]	Solvent PME energy.
dTStrans		First order translational entropy.
dTSorient		First order orientational entropy.
order		(if `doorder` was specified) Average Tetrahedral Order Parameter [99].
dipolex		x-component of the mean solvent dipole moment density
dipoley		y-component of the mean solvent dipole moment density
dipolez		z-component of the mean solvent dipole moment density
dipole		Magnitude of mean dipole moment density (polarization).
neighbor		Mean number of solvent molecules neighboring the solvent molecules found in this voxel.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059



1. MD Simulation

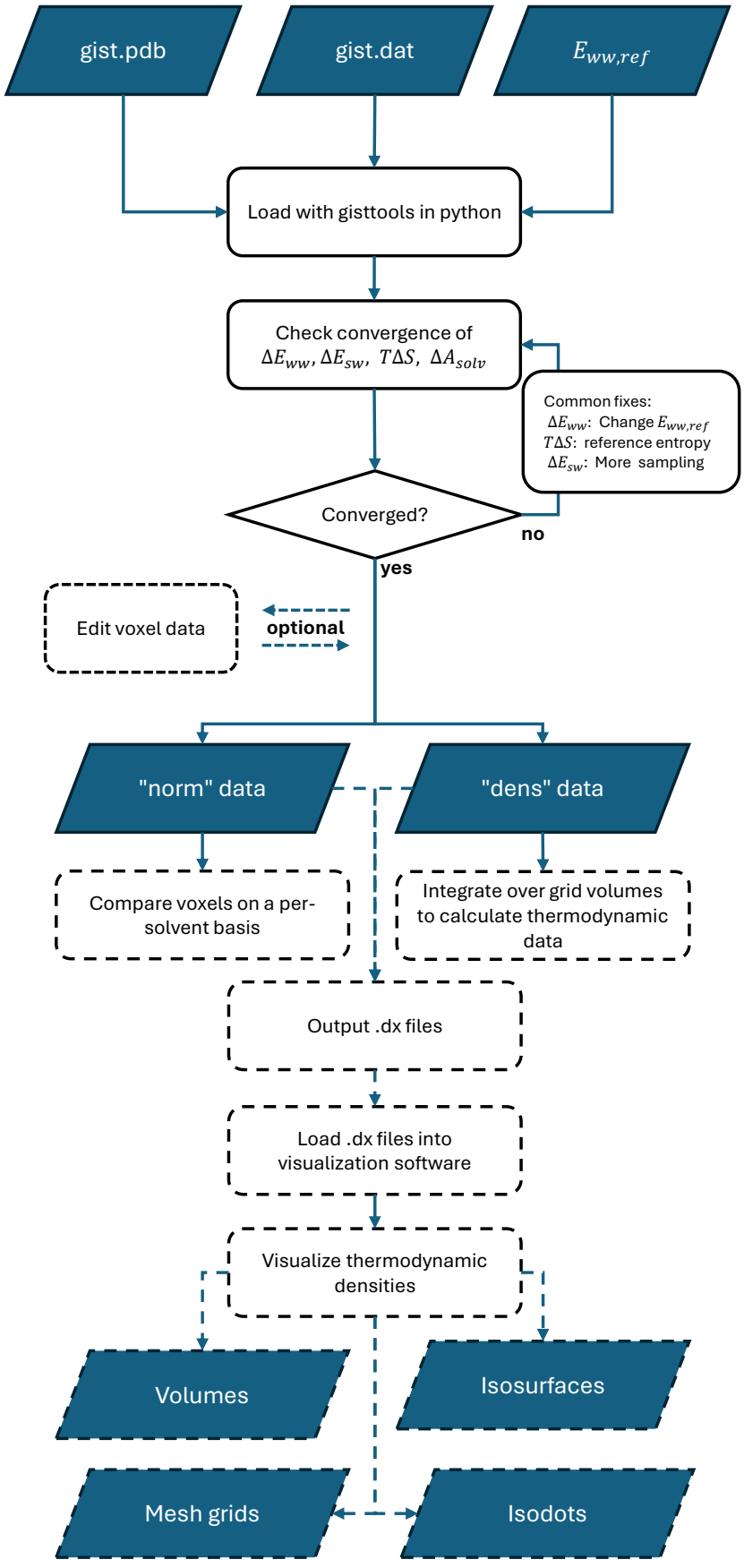
- ❑ Add missing atoms
- ❑ Check protonation states
- ❑ Select force field
- ❑ Add solvent
  At least 10-15Å buffer or 3 solvation layers depending on solvent
- ❑ Optional: create neat solvent box and determine NN entropy bias
- ❑ Restrain heavy atoms for solute simulation
  Harmonic restraint potential $V_r(x) = k_r \cdot \Delta x^2$ with k_r =10 - 100 kcal mol^{-1} Å^{-2}
- ❑ Run simulation
  ≥10 ns (100 ns ideal)
  ≥ 10 000 frames (100 000 ideal)
- ❑ Confirm rigidity of solute heavy atoms: Suggested RMSD ≤ 0.1Å

2. Prepare GIST settings

- ❑ Determine reference water density ρ_0
  Calculate reference from neat simulation or use tabulated value for the solvent model.
- ❑ Determine grid size
  a) binding site
  b) full system
  c) 10 Å around region of interest
- ❑ Determine voxel number
  For each cartesian direction, the number of voxels necessary is the length of the region of interest divided by the voxel side length (0.5 Å).

3. Run GIST in cpptraj

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059



4a. Postprocess & Analysis

- ❑ Determine $E_{ww,ref}$
  Use tabulated value, calculate from reference simulation or auto-reference from outer grid voxels, if the grid is large enough (> 3 solvent layers)
- ❑ Load gist.dat with gisttools
  A pdb structure of the solute without water is necessary for this. Preset $E_{ww,ref}$ or use the auto-referencing.
- ❑ Check convergence of thermodynamic properties – integrated values should approach a fixed value with distance from the region of interest (see Figure 4)
- ❑ Optional: edit voxel data
  For example, apply the empirical entropy correction factor of 0.6.
- ❑ Use “norm“ data to compare voxel to voxel
- ❑ Use “dens“ data to integrate over voxels or for visualization purposes

4b. Visualization (optional)

- ❑ If properties were modified or newly created, output density files for visualization
- ❑ Visualize .dx files
  e.g. with PyMOL or VMD

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

Author Contributions

All authors were involved in reviewing and editing the original manuscript.
VJEH and FW conceptualized the research and co-wrote the initial draft.
VM contributed the original drafts of sections 7 and 6.
HC helped with code curation and validation.
MLFQ helped with conceptualization and gave writing input for the initial draft.
SR and DR contributed technical input.
KRL, MKG and TK were involved with conceptualization, funding acquisition, project management and supervision.

For a more detailed description of author contributions, see the GitHub issue tracking and changelog at https://github.com/liedllab/gist-tutorial.

Potential Conflicts of Interest

MKG has an equity interest in and is a cofounder and scientific advisor of VeraChem LLC.

Funding Information

This research was funded in whole or in part by the Austrian Science Fund (FWF) [10.55776/P34518].

Author Information

ORCID:
Valentin J. Egger-Hoerschinger: 0000-0002-4469-3238
Franz Waibl: 0000-0003-0527-0803
Vjay Molino: 0009-0005-2055-9295
Helmut Carter: 0000-0003-0273-4107
Monica L. Fernández-Quintero: 0000-0002-6811-6283
Steven Ramsey: 0000-0001-7441-3228
Daniel R. Roe: 0000-0002-5834-2447
Klaus R. Liedl: 0000-0002-0985-2299
Michael K. Gilson: 0000-0002-3375-1738
Tom Kurtzman: 0000-0003-0900-772X

References

[1] **Nguyen CN**, Kurtzman Young T, Gilson MK. Grid Inhomogeneous Solvation Theory: Hydration Structure and Thermodynamics of the Miniature Receptor Cucurbit[7]uril. J Chem Phys. 2012; 137(4):044101. https://doi.org/10.1063/1.4733951.

[2] **Nguyen CN**, Kurtzman T, Gilson MK. Spatial Decomposition of Translational Water–Water Correlation Entropy in Binding Pockets. J Chem Theory Comput. 2015; 12(1):414–429. https://doi.org/10.1021/acs.jctc.5b00939.

[3] **Waibl F**, Kraml J, Fernández-Quintero ML, Loeffler JR, Liedl KR. Explicit Solvation Thermodynamics in Ionic Solution: Extending Grid Inhomogeneous Solvation Theory to Solvation Free Energy of Salt–Water Mixtures. J Comput-Aided Mol Des. 2022; 36(2):101–116. https://doi.org/10.1007/s10822-021-00429-y.

[4] **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. J Chem Theory Comput. 2021; 17(5):2714–2724. https://doi.org/10.1021/acs.jctc.0c01185.

[5] **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.

[6] **Ramsey S**, Nguyen C, Salomon-Ferrer R, Walker RC, Gilson MK, Kurtzman T. Solvation Thermodynamic Mapping of Molecular Surfaces in AmberTools: GIST. J Comput Chem. 2016; 37(21):2029–2037. https://doi.org/10.1002/jcc.24417.

[7] **Balius TE**, Fischer M, Stein RM, Adler TB, Nguyen CN, Cruz A, Gilson MK, Kurtzman T, Shoichet BK. Testing Inhomogeneous Solvation Theory in Structure-Based Ligand Discovery. PNAS. 2017; 114(33):E6839–e6846. https://doi.org/10.1073/pnas.1703287114.

[8] **Hüfner-Wulsdorf T**, Klebe G. Protein–Ligand Complex Solvation Thermodynamics: Development, Parameterization, and Testing of GIST-Based Solvent Functionals. J Chem Inf Model. 2020; 60(3):1409–1423. https://doi.org/10.1021/acs.jcim.9b01109.

[9] **Olson B**, Cruz A, Chen L, Ghattas M, Ji Y, Huang K, Ayoub S, Luchko T, McKay DJ, Kurtzman T. An Online Repository of Solvation Thermodynamic and Structural Maps of SARS-CoV-2 Targets. J Comput-Aided Mol Des. 2020; 34(12):1219–1228. https://doi.org/10.1007/s10822-020-00341-x.

[10] **Kraml J**, Kamenik AS, Waibl F, Schauperl M, Liedl KR. Solvation Free Energy as a Measure of Hydrophobicity: Application to Serine Protease Binding Interfaces. J Chem Theory Comput. 2019; 15(11):5872–5882. https://doi.org/10.1021/acs.jctc.9b00742.

[11] **Kamenik AS**, Kraml J, Hofer F, Waibl F, Quoika PK, Kahler U, Schauperl M, Liedl KR. Macrocycle Cell Permeability Measured by Solvation Free Energies in Polar and Apolar Environments. J Chem Inf Model. 2020; 60(7):3508–3517. https://doi.org/10.1021/acs.jcim.0c00280.

[12] **Waibl F**, Fernández-Quintero ML, Kamenik AS, Kraml J, Hofer F, Kettenberger H, Georges G, Liedl KR. Conformational Ensembles of Antibodies Determine Their Hydrophobicity. Biophys J. 2021; 120(1):143–157. https://doi.org/10.1016/j.bpj.2020.11.010.

[13] **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.

[14] **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. J Chem Inf Model. 2020; 60(8):3843–3853. https://doi.org/10.1021/acs.jcim.0c00289.

[15] **Salomon R**, Nguyen C, Ramsey S, Gough JD, Walker R, Kurtzman T, AMBER Tutorials: 25. Analysis of Water Thermodynamics Using Grid Inhomogeneous Solvation Theory of Factor Xa Active Site;. https://ambermd.org/tutorials/advanced/tutorial25/.

[16] **Borhani TN**, García-Muñoz S, Vanesa Luciani C, Galindo A, Adjiman CS. Hybrid QSPR Models for the Prediction of the Free Energy of Solvation of Organic Solute/Solvent Pairs. Phys Chem Chem Phys. 2019; 21(25):13706–13720. https://doi.org/10.1039/c8cp07562j.

[17] **Fredenslund A**, Jones RL, Prausnitz JM. Group-contribution estimation of activity coefficients in non-ideal liquid mixtures. AIChE J. 1975; 21(6):1086–1099. https://doi.org/10.1002/aic.690210607.

[18] **Miertuš S**, Scrocco E, Tomasi J. Electrostatic Interaction of a Solute with a Continuum. A Direct Utilizaion of Ab-initio Molecular Potentials for the Prevision of Solvent Effects. Chem Phys. 1981; 55(1):117–129. https://doi.org/10.1016/0301-0104(81)85090-2.

[19] **Klamt A**, Schüürmann G. COSMO: a New Approach to Dielectric Screening in Solvents with Explicit Expressions for the Screening Energy and Its Gradient. J Chem Soc, Perkin Trans 2. 1993; (5):799–805. https://doi.org/10.1039/p29930000799.

[20] **Mennucci B**. Continuum Solvation Models: What Else Can We Learn from Them? J Phys Chem Lett. 2010; 1(10):1666–1674. https://doi.org/10.1021/jz100506s.

[21] **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.

[22] **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. 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.

[23] **Genheden S**, Ryde U. The MM/PBSA and MM/GBSA Methods to Estimate Ligand-Binding Affinities. Expert Opin Drug Discovery. 2015; 10(5):449–461. https://doi.org/10.1517/17460441.2015.1032936.

[24] **Liu S**, Cao S, Hoang K, Young KL, Paluch AS, Mobley DL. Using MD Simulations To Calculate How Solvents Modulate Solubility. J Chem Theory Comput. 2016; 12(4):1930–1941. https://doi.org/10.1021/acs.jctc.5b00934.

[25] **Swails JM**, York DM, Roitberg AE. Constant pH Replica Exchange Molecular Dynamics in Explicit Solvent Using Discrete Protonation States: Implementation, Testing, and Validation. J Chem Theory Comput. 2014; 10(3):1341–1352. https://doi.org/10.1021/ct401042b.

[26] **Haider K**, Wickstrom L, Ramsey S, Gilson MK, Kurtzman T. Enthalpic Breakdown of Water Structure on Protein Active-Site Surfaces. J Phys Chem B. 2016; 120(34):8743–8756. https://doi.org/10.1021/acs.jpcb.6b01094.

[27] **Pratt LR**, Chaudhari MI, Rempe SB. Statistical Analyses of Hydrophobic Interactions: A Mini-Review. J Phys Chem B. 2016; 120(27):6455–6460. https://doi.org/10.1021/acs.jpcb.6b04082.

[28] **Mobley DL**, Bayly CI, Cooper MD, Shirts MR, Dill KA. Small Molecule Hydration Free Energies in Explicit Solvent: An Extensive Test of Fixed-Charge Atomistic Simulations. J Chem Theory Comput. 2009; 5(2):350–358. https://doi.org/10.1021/ct800409d.

[29] **Mobley DL**, Guthrie JP. FreeSolv: a Database of Experimental and Calculated Hydration Free Energies, with Input Files. J Comput-Aided Mol Des. 2014; 28(7):711–720. https://doi.org/10.1007/s10822-014-9747-x.

[30] **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.

[31] **Kirkwood JG**. Statistical Mechanics of Fluid Mixtures. J Chem Phys. 1935; 3(5):300–313. https://doi.org/10.1063/1.1749657.

[32] **Peter C**, Oostenbrink C, van Dorp A, van Gunsteren WF. Estimating Entropies from Molecular Dynamics Simulations. J Chem Phys. 2004; 120(6):2652–2661. https://doi.org/10.1063/1.1636153.

[33] **Persson RAX**. Note on the Physical Basis of Spatially Resolved Thermodynamic Functions. Mol Simul. 2022; 48(13):1186–1191. https://doi.org/10.1080/08927022.2022.2074994.

[34] **Hansen JP**. Theory of Simple Liquids. 4. ed. **McDonald IR**, editor, Amsterdam: Elsevier; 2013. https://doi.org/10.1016/c2010-0-66723-x.

[35] **Lazaridis T**. Inhomogeneous Fluid Approach to Solvation Thermodynamics. 1. Theory. J Phys Chem B. 1998; 102(18):3531–3541. https://doi.org/10.1021/jp9723574.

[36] **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.

[37] **Kovalenko A**, Hirata F. Three-Dimensional Density Profiles of Water in Contact with a Solute of Arbitrary Shape: A RISM Approach. Chem Phys Lett. 1998; 290(1–3):237–244. https://doi.org/10.1016/s0009-2614(98)00471-0.

[38] **Young T**, Abel R, Kim B, Berne BJ, Friesner RA. Motifs for Molecular Recognition Exploiting Hydrophobic Enclosure in Protein–Ligand Binding. PNAS. 2007; 104(3):808–813. https://doi.org/10.1073/pnas.0610202104.

[39] **Abel R**, Young T, Farid R, Berne BJ, Friesner RA. Role of the Active-Site Solvent in the Thermodynamics of Factor Xa Ligand Binding. JACS. 2008; 130(9):2817–2831. https://doi.org/10.1021/ja0771033.

[40] **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. 2017; 14(1):418–425. https://doi.org/10.1021/acs.jctc.7b00592.

[41] **Li Z**, Lazaridis T. In: Computing the Thermodynamic Contributions of Interfacial Water, vol. 819 Springer New York; 2011. p. 393–404. https://doi.org/10.1007/978-1-61779-465-0_24.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

[42] **Huggins DJ**. Studying the Role of Cooperative Hydration in Stabilizing Folded Protein States. J Struct Biol. 2016; 196(3):394–406. https://doi.org/10.1016/j.jsb.2016.09.003.

[43] **Reinhard F**, Grubmüller H. Estimation of Absolute Solvent and Solvation Shell Entropies Via Permutation Reduction. J Chem Phys. 2007; 126(1). https://doi.org/10.1063/1.2400220.

[44] **Heinz LP**, Grubmüller H. Computing Spatially Resolved Rotational Hydration Entropies from Atomistic Simulations. J Chem Theory Comput. 2019; 16(1):108–118. https://doi.org/10.1021/acs.jctc.9b00926.

[45] **Heinz LP**, Grubmüller H. Per|Mut: Spatially Resolved Hydration Entropies from Atomistic Simulations. J Chem Theory Comput. 2021; 17(4):2090–2098. https://doi.org/10.1021/acs.jctc.0c00961.

[46] **Fogolari F**, Esposito G. Optimal Relabeling of Water Molecules and Single-Molecule Entropy Estimation. Biophysica. 2021; 1(3):279–296. https://doi.org/10.3390/biophysica1030021.

[47] **Fogolari F**, Borelli R, Dovier A, Esposito G. The k-th Nearest Neighbor Method for Estimation of Entropy Changes from Molecular Ensembles. Wiley Interdiscip Rev: Comput Mol Sci. 2023; 14(1):e1691. https://doi.org/10.1002/wcms.1691.

[48] **Gilson MK**, Kurtzman T. Free Energy Density of a Fluid and Its Role in Solvation and Binding. J Chem Theory Comput. 2024; 20(7):2871–2887. https://doi.org/10.1021/acs.jctc.3c01173.

[49] **Heyden M**. Disassembling Solvation Free Energies into Local Contributions–Toward a Microscopic Understanding of Solvation Processes. Wiley Interdiscip Rev:Comput Mol Sci. 2018; 9(2):e1390. https://doi.org/10.1002/wcms.1390.

[50] **Mukherjee S**, Schäfer LV. Spatially Resolved Hydration Thermodynamics in Biomolecular Systems. J Phys Chem B. 2022; 126(20):3619–3631. https://doi.org/10.1021/acs.jpcb.2c01088.

[51] **Bayden AS**, Moustakas DT, Joseph-McCarthy D, Lamb ML. Evaluating Free Energies of Binding and Conservation of Crystallographic Waters Using SZMAP. J Chem Inf Model. 2015; 55(8):1552–1565. https://doi.org/10.1021/ci500746d.

[52] **Cui G**, Swails JM, Manas ES. SPAM: A Simple Approach for Profiling Bound Water Molecules. J Chem Theory Comput. 2013; 9(12):5539–5549. https://doi.org/10.1021/ct400711g.

[53] **Henchman RH**. Partition Function for a Simple Liquid Using Cell Theory Parametrized by Computer Simulation. J Chem Phys. 2003; 119(1):400–406. https://doi.org/10.1063/1.1578622.

[54] **Gerogiokas G**, Calabro G, Henchman RH, Southey MWY, Law RJ, Michel J. Prediction of Small Molecule Hydration Thermodynamics with Grid Cell Theory. J Chem Theory Comput. 2013; 10(1):35–48. https://doi.org/10.1021/ct400783h.

[55] **Henchman RH**. Free Energy of Liquid Water from a Computer Simulation via Cell Theory. J Chem Phys. 2007; 126(6). https://doi.org/10.1063/1.2434964.

[56] **Hensen U**, Gräter F, Henchman RH. Macromolecular Entropy Can Be Accurately Computed from Force. J Chem Theory Comput. 2014; 10(11):4777–4781. https://doi.org/10.1021/ct500684w.

[57] **Ali HS**, Chakravorty A, Kalayan J, de Visser SP, Henchman RH. Energy-Entropy Method Using Multiscale Cell Correlation to Calculate Binding Free Energies in the SAMPL8 Host-Guest Challenge. J Comput-Aided Mol Des. 2021; 35(8):911–921. https://doi.org/10.1007/s10822-021-00406-5.

[58] **Kalayan J**, Chakravorty A, Warwicker J, Henchman RH. Total Free Energy Analysis of Fully Hydrated Proteins. Proteins:Struct, Funct, Bioinf. 2022; 91(1):74–90. https://doi.org/10.1002/prot.26411.

[59] **Case DA**, Aktulga HM, Belfon K, Ben-Shalom IY, Berryman JT, Brozell SR, Cerutti DS, Cheatham TEI, Cisneros GA, Cruzeiro VWD, Darden TA, Forouzesh N, Giambasu G, Ghazimirsaeed M, Giambaşu K, Giese T, Gilson MK, Gohlke H, Goetz AW, Harris R, et al., AMBER 2024. University of California, San Francisco; 2024.

[60] **Case DA**, Aktulga HM, Belfon K, Cerutti DS, Cisneros GA, Cruzeiro VWD, Forouzesh N, Giese TJ, Götz AW, Gohlke H, Izadi S, Kasavajhala K, Kaymak MC, King E, Kurtzman T, Lee TS, Li P, Liu J, Luchko T, Luo R, et al. AmberTools. J Chem Inf Model. 2023; 63(20):6183–6191. https://doi.org/10.1021/acs.jcim.3c01153.

[61] **Roe DR**, CPPTRAJ Documentation;. https://raw.githubusercontent.com/Amber-MD/cpptraj/master/doc/CpptrajManual.pdf.

[62] **Roe DR**, Various Authors, AMBER Tutorials: 4. Trajectory Analysis;. https://ambermd.org/tutorials/TrajectoryAnalysis.php.

[63] **Bergonzo C**, Alviz-Amador AA, Roe DR, Love O, Winkler L, Galindo-Murillo R, Cheatham III TE, AMBER-Hub;. https://amberhub.chpc.utah.edu.

[64] **Abraham MJ**, Murtola T, Schulz R, Páll S, Smith JC, Hess B, Lindahl E. GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers. SoftwareX. 2015; 1–2:19–25. https://doi.org/10.1016/j.softx.2015.06.001.

[65] **Pronk S**, Páll S, Schulz R, Larsson P, Bjelkmar P, Apostolov R, Shirts MR, Smith JC, Kasson PM, van der Spoel D, Hess B, Lindahl E. GROMACS 4.5: a High-Throughput and Highly Parallel Open Source Molecular Simulation Toolkit. Bioinformatics. 2013; 29(7):845–854. https://doi.org/10.1093/bioinformatics/btt055.

[66] **Sousa da Silva AW**, Vranken WF. ACPYPE - AnteChamber PYthon Parser interfacE. BMC Res Notes. 2012; 5(1):367. https://doi.org/10.1186/1756-0500-5-367.

[67] **Roe DR**, Brooks BR. Quantifying the Effects of Lossy Compression on Energies Calculated from Molecular Dynamics Trajectories. Protein Sci. 2022; 31(12):e4511. https://doi.org/10.1002/pro.4511.

[68] **McGibbon RT**, Beauchamp KA, Harrigan MP, Klein C, Swails JM, Hernández CX, Schwantes CR, Wang LP, Lane TJ, Pande VS. MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. Biophys J. 2015; 109(8):1528–1532. https://doi.org/10.1016/j.bpj.2015.08.015.

[69] **Hunter JD**. Matplotlib: A 2D Graphics Environment. Comput Sci Eng. 2007; 9(3):90–95. https://doi.org/10.1109/mcse.2007.55.

[70] **Harris CR**, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, et al. Array Programming with NumPy. Nature. 2020; 585(7825):357–362. https://doi.org/10.1038/s41586-020-2649-2.

[71] **McKinney W**. Data Structures for Statistical Computing in Python. In: van der Walt S, Millman J, editors. *Proceedings of the 9th Python in Science Conference* SciPy, SciPy; 2010. p. 56–61. https://doi.org/10.25080/majora-92bf1922-00a.

[72] **Kluyver T**, Ragan-Kelley B, Pérez F, 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. **Loizides F**, Schmidt B, editors, Netherlands: IOS Press; 2016. https://eprints.soton.ac.uk/403913/.

[73] **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.

[74] **Jorgensen WL**, Chandrasekhar J, Madura JD, Impey RW, Klein ML. Comparison of Simple Potential Functions for Simulating Liquid Water. J Chem Phys. 1983; 79(2):926–935. https://doi.org/10.1063/1.445869.

[75] **Wang LP**, Martinez TJ, Pande VS. Building Force Fields: An Automatic, Systematic, and Reproducible Approach. J Phys Chem Lett. 2014; 5(11):1885–1891. https://doi.org/10.1021/jz500737m.

[76] **Jorgensen WL**, Madura JD. Temperature and Size Dependence for Monte Carlo Simulations of TIP4P Water. Mol Phys. 1985; 56(6):1381–1392. https://doi.org/10.1080/00268978500103111.

[77] **Horn HW**, Swope WC, Pitera JW, Madura JD, Dick TJ, Hura GL, Head-Gordon T. Development of an Improved Four-Site Water Model for Biomolecular Simulations: TIP4P-Ew. J Chem Phys. 2004; 120(20):9665–9678. https://doi.org/10.1063/1.1683075.

[78] **Mahoney MW**, Jorgensen WL. A Five-Site Model for Liquid Water and the Reproduction of the Density Anomaly by Rigid, Nonpolarizable Potential Functions. J Chem Phys. 2000; 112(20):8910–8922. https://doi.org/10.1063/1.481505.

[79] **Berendsen HJC**, Grigera JR, Straatsma TP. The Missing Term in Effective Pair Potentials. J Phys Chem. 1987; 91(24):6269–6271. https://doi.org/10.1021/j100308a038.

[80] **Takemura K**, Kitao A. Water Model Tuning for Improved Reproduction of Rotational Diffusion and NMR Spectral Density. J Phys Chem B. 2012; 116(22):6279–6287. https://doi.org/10.1021/jp301100g.

[81] **Izadi S**, Anandakrishnan R, Onufriev AV. Building Water Models: A Different Approach. J Phys Chem Lett. 2014; 5(21):3863–3871. https://doi.org/10.1021/jz501780a.

[82] **Izadi S**, Onufriev AV. Accuracy Limit of Rigid 3-Point Water Models. J Chem Phys. 2016; 145(7):074501. https://doi.org/10.1063/1.4960175.

[83] **Dundas CM**, Demonte D, Park S. Streptavidin–Biotin Technology: Improvements and Innovations in Chemical and Biological Applications. Appl Microbiol Biotechnol. 2013; 97(21):9343–9353. https://doi.org/10.1007/s00253-013-5232-z.

[84] **McConnell DB**. Biotin's Lessons in Drug Design. J Med Chem. 2021; 64(22):16319–16327. https://doi.org/10.1021/acs.jmedchem.1c00975.

[85] **Bansal N**, Zheng Z, Song LF, Pei J, Merz KM. The Role of the Active Site Flap in Streptavidin/Biotin Complex Formation. JACS. 2018; 140(16):5434–5446. https://doi.org/10.1021/jacs.8b00743.

[86] **Weber PC**, Ohlendorf DH, Wendoloski JJ, Salemme FR. Structural Origins of High-Affinity Biotin Binding to Streptavidin. Science. 1989; 243(4887):85–88. https://doi.org/10.1126/science.2911722.

[87] **Roe DR**, Brooks BR. A Protocol for Preparing Explicitly Solvated Systems for Stable Molecular Dynamics Simulations. J Chem Phys. 2020; 153(5):054123. https://doi.org/10.1063/5.0013849.

[88] **Frisch HL**, Lebowitz JL, Rice SA. The Equilibrium Theory of Classical Fluids. Phys Today. 1965; 18(5):78–80. https://doi.org/10.1063/1.3047436.

[89] **Huggins DJ**. Estimating Translational and Orientational Entropies Using the k-Nearest Neighbors Algorithm. J Chem Theory Comput. 2014; 10(9):3617–3625. https://doi.org/10.1021/ct500415g.

[90] **Roe DR**, Brooks BR. MPI-Parallelization of the Grid Inhomogeneous Solvation Theory Calculation. J Comput Chem. 2024; 45(10):633–637. https://doi.org/10.1002/jcc.27278.

[91] **Ge Y**, Wych DC, Samways ML, Wall ME, Essex JW, Mobley DL. Enhancing Sampling of Water Rehydration on Ligand Binding: A Comparison of Techniques. J Chem Theory Comput. 2022; 18(3):1359–1381. https://doi.org/10.1021/acs.jctc.1c00590.

[92] **Melling OJ**, Samways ML, Ge Y, Mobley DL, Essex JW. Enhanced Grand Canonical Sampling of Occluded Water Sites Using Nonequilibrium Candidate Monte Carlo. J Chem Theory Comput. 2023; 19(3):1050–1062. https://doi.org/10.1021/acs.jctc.2c00823.

[93] **Schrödinger, LLC**, The Pymol Molecular Graphics System, Version 3.0; 2025.

[94] **Humphrey W**, Dalke A, Schulten K. VMD: Visual Molecular Dynamics. J Mol Graphics. 1996; 14(1):33–38. https://doi.org/10.1016/0263-7855(96)00018-5.

[95] **L Mpye K**, Gildenhuys S, Mosebi S. The Effects of Temperature on Streptavidin-Biotin Binding Using Affinity Isothermal Titration Calorimetry. AIMS Biophys. 2020; 7(4):236–247. https://doi.org/10.3934/biophy.2020018.

[96] **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 Sci. 2006; 15(3):459–467. https://doi.org/10.1110/ps.051970306.

[97] **Ben-Naim A**. Solvation Thermodynamics. New York, NY: Springer US; 1987. https://doi.org/10.1007/978-1-4757-6550-2.

https://doi.org/10.33011/livecoms.6.1.3059
Living J. Comp. Mol. Sci. 2025, 6(1), 3059

[98] **Heinz LP**, Grubmüller H. Why Solvent Response Contributions to Solvation Free Energies Are Compatible with Ben-Naim's Theorem. J Chem Theory Comput. 2023; 19(22):8013–8019. https://doi.org/10.1021/acs.jctc.3c00655.

[99] **Chatterjee S**, Debenedetti PG, Stillinger FH, Lynden-Bell RM. A Computational Investigation of Thermodynamics, Structure, Dynamics and Solvation Behavior in Modified Water Models. J Chem Phys. 2008; 128(12). https://doi.org/10.1063/1.2841127.