Generalized Weighted Histogram Analysis Method

Dejun Lin

Department of Biochemistry and Biophysics, University of Rochester Medical Center, Rochester, New York
dejun_lin@urmc.rochester.edu

April 13, 2015

Contents

1 About this document

This document describes the theory and implementation of a generalized version of Weighted Histogram Analysis Method (WHAM). Section 2 describes what types of problem WHAM can solve and section 3 describes how WHAM solve these problems. Section 4 describes my C++ implementation of WHAM and section 5 describes how to use this implementation. All users are strongly encouraged to read carefully and understand this documentation before using the software. The software is under the GNU General Public License (GPL) except for a set of C++ routines from Numerical Recipes (6).

2 Introduction

Weighted Histogram Analysis Method has been widely used in the field of molecular dynamics simulations in calculating the potential of mean force (PMF) (4, 7). It has also been used in free energy calculation in a alchemical scheme (4) as well as estimating the ensemble average from generalized ensemble simulations (1). However, commonly used implementation of WHAM such as Alan Grossfield's implementation (2) and GROMACS “g_wham" only support PMF calculation. Moreover, most implementations uses the direct iteration to solve for PMFs while it's been shown (8) that there are more efficient way to reach the same answers. Here I combine the reasonings behind the aforementioned literature and put together an implementation of WHAM that's capable of solving the following general problem:

Suppose the we have K systems of the same thermodynamic ensemble with each characterized by different thermodynamics parameters. If we let

$\mathbf{x} \in \mathbb{R}^N$ being the generalized coordinates, which has N degrees of freedom, and

$$H_i(\mathbf{x}) = \beta_i[U_i(\mathbf{x}) + P_i V(\mathbf{x}) + \boldsymbol{\mu}_\mathbf{i} \cdot \mathbf{n}(\mathbf{x})] \tag{2.1}$$

be the Hamiltonian of the i'th system, where $\beta_i, U_i, P_i, V, \boldsymbol{\mu}_i$ and $\mathbf{n}$ are the inverse temperature, potential energy, the pressure, the volume, the chemical potentials for each system components and the number of molecules for each components, respectively. The potential energy U_i can be further decomposed into the internal potential U and L external potentials $\mathbf{U}_b$, both of which are of the same form across the K systems, so that

$$U_i(\mathbf{x}) = U(\mathbf{x}) + \boldsymbol{\lambda}_\mathbf{i} \cdot \mathbf{U}_\mathbf{b} \tag{2.2}$$

Each of the K systems can has its own specific $\boldsymbol{\lambda}_i$, which is often used in the alchemical calculations. Here both $\boldsymbol{\lambda}_i$ and $\mathbf{U}_b$ are of the dimension of L. To simplify the expression, we rewrite equation 2.1 into

$$H_i(\mathbf{x}) = \beta_i \mathbf{R}_\mathbf{i} \cdot \mathbf{q}(\mathbf{x}) \tag{2.3}$$

where

$$\mathbf{R}_i \equiv \{1, \boldsymbol{\lambda}_i, P_i, \boldsymbol{\mu}_i\} \tag{2.4}$$

and

$$\mathbf{q}(\mathbf{x}) \equiv \{U(\mathbf{x}), \mathbf{U}_b(\mathbf{x}), V(\mathbf{x}), \mathbf{n}(\mathbf{x})\} \tag{2.5}$$

are the arrays of intensive and extensive parameters, respectively. We further define unnormalized probability density

$$p_i(\mathbf{x}) \equiv \exp(-H_i(\mathbf{x})) \tag{2.6}$$

and the partition function

$$Q_i \equiv \int p_i(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{2.7}$$

Now we'd like to estimate the ensemble average of some observable $O \equiv O(\mathbf{x})$ in any of the system i

$$\langle O \rangle_i \equiv \frac{\int p_i(\mathbf{x}) O(\mathbf{x}) \mathrm{d}\mathbf{x}}{Q_i} \tag{2.8}$$

For example, the PMF of a reaction coordinate $\boldsymbol{\xi} \equiv \boldsymbol{\xi}(\mathbf{x})$ in system i is

$$w_i(\boldsymbol{\xi}) \equiv \langle \delta(\boldsymbol{\xi} - \boldsymbol{\xi}(\mathbf{x})) \rangle_i \tag{2.9}$$

where δ is the Dirac delta function. Or in other cases we want to know the free energy difference between any two systems i and j:

$$\Delta G_{ij} \equiv -k_B T \ln \frac{Q_j}{Q_i} \tag{2.10}$$

To do that, we carry out S simulations ($S \geq K$) with each simulation visiting at least 1 of the K systems. As soon as the transition from one systems to the other follows the same statistics as those used in transition between configuration in any of the systems, we can simply assert $S = K$ for simplicity and the cases where $S > K$ can be handled accordingly without troubles at all. The purpose of WHAM is to provide an optimal estimate of O and ΔG_{ij} by combining the estimates from all the S simulations.

3 Theory

To solve equation 2.8, we can in principle just sample the configuration space $\mathbf{x}$ by brute forces. However, in the presence of rugged high-dimensional energy surface, it's in general extremely difficult to get a converged answer by solving equation 2.8 directly, i.e., by sampling the configuration space especially at near room temperature. Usually the system would exhibit metastability and get stuck at some local energy minima and it could take a very long to escape from them. The general idea behind a lot of enhanced sampling methods, such as the generalized ensemble approach or the umbrella sampling method, is to let the system be able to walk up-and-down in the energy surface faster by either perturbing the probability of transitioning between energy levels or to modify the energy surface *per se* systematically so that the equilibrium thermodynamics is maintained, which allows us to estimate the average in the right equilibrium ensemble. Generally, these enhanced sampling approach works by either having the system perform random walk at different energy levels or by having different system explore configurations at different energy levels. Even though the details of how to carry out the simulations might be different, they're all under a common premise, which is they all need to have a set of quantities which are conserved across all the systems at different energy levels or with different perturbation to the energy surface so that the information from different systems can be combined and still give the right answer.

One obvious quantity is $\mathbf{q}$ as defined in equation 2.5. If we know $p_i(\mathbf{q})$, the probability density of $\mathbf{q}$, we can reformulate equations 2.7 and 2.8 in terms of $p_i(\mathbf{q})$:

$$Q_i = \int p_i(\mathbf{q}) \mathrm{d}\mathbf{q} \tag{3.1}$$

$$\langle O \rangle_i = \frac{\int p_i(\mathbf{q}) O(\mathbf{q}) \mathrm{d}\mathbf{q}}{Q_i} \tag{3.2}$$

and $p_i(\mathbf{q})$ is defined as

$$p_i(\mathbf{q}) \equiv \Omega(\mathbf{q}) \exp(-H_i(\mathbf{q})) \tag{3.3}$$

where

$$\Omega(\mathbf{q}) \equiv \int \delta(\mathbf{q} - \mathbf{q}(\mathbf{x})) \mathrm{d}\mathbf{x} \tag{3.4}$$

is the density of state at $\mathbf{q}$. $O(\mathbf{q})$ can be obtained by

$$O(\mathbf{q}) \equiv \frac{\int \delta(\mathbf{q} - \mathbf{q}(\mathbf{x})) O(\mathbf{x}) \mathrm{d}\mathbf{x}}{\Omega(\mathbf{q})} \tag{3.5}$$

Equation 3.5 can be understood as the weighted average of O as as a function of $\mathbf{q}$ with the weights for each value of O being the number of configurations that gives a certain pair of values of $\mathbf{q}$ and O at the same time. Note that in this case, the Hamiltonian of the system is completely determined by $\mathbf{q}$. However, we'll show later that $\mathbf{q}$ only needs to be a subset of equation 2.5 such that the the corresponding subset of equation 2.4 characterizes all the differences between any two of the systems. For example, if any two systems in the NPT ensemble are different in temperature only, then the minimal subset of $\mathbf{q}$ we need to consider is the interal potential energy U.

WHAM works in the following way. First, we discretize the space of $\mathbf{q}$ into a $d \equiv \dim(\mathbf{q})$ -dimensional grid, where each grid point can be identified by a unique vector $\mathbf{m} \in \mathbb{Z}^d$. Now we denote the value of a function of $\mathbf{q}$: $f(\mathbf{q})$ takes in bin $\mathbf{m}$ as $f_{\mathbf{m}}$. Say, we have a total of N_i samples from the i'th simulation where $n_{i,\mathbf{m}}$ are histogrammed into the bin $\mathbf{m}$. Then we combine the histograms from all S simulations to optimally estimate $\Omega_{\mathbf{m}}$ and thus $p_{i,m}$. There're several approaches to arrive at the same set of WHAM equations that gives the optimal estimate of $\Omega_{\mathbf{m}}$: the original WHAM paper(4) as well as the PT/STWHAM one(1) treats $\Omega_{\mathbf{m}}$ as a weighted sum of $\Omega_{i,\mathbf{m}}$, the density of state in bin $\mathbf{m}$ estimated from the i'th simulation alone, where

the weights are $\delta^2\Omega_{i,\mathbf{m}}$, the respective statistical uncertainty of $\Omega_{i,\mathbf{m}}$ from each simulation. This approach introduces the statistical uncertainty in the samples and help the development of basis error analysis. But the number of WHAM equations resulting from this approach is the same as number of bins, which takes a tremendous amount of computational resources to solve. Here we take the maximum likelihood approach instead, which was reported by Zhu and Hummer (8), because it shows a way to solve the same problem in terms of Q_i in lieu of $\Omega_{\mathbf{m}}$, which dramatically reduces the number of parameters to determine and boost the computation significantly (8). Although in that paper, their results only applied to calculating PMFs, which is not an limitation on the discussion here.

The maximum likelihood approach maximize the following likelihood function given $\{p_{i,\mathbf{m}}\}_i$ and $\{n_{i,\mathbf{m}}\}_i$, where $\{f_i\}_i$ denotes the set of f_i for all i:

$$A(\{n_{i,\mathbf{m}}\}_{i,\mathbf{m}}|\{p_{i,\mathbf{m}}\}_{i,\mathbf{m}}) \equiv \prod_{i=1}^{S} P_i(\{n_{i,\mathbf{m}}\}_{\mathbf{m}}|\{p_{i,\mathbf{m}}\}_{\mathbf{m}}) \tag{3.6}$$

where $P_i(\{n_{i,\mathbf{m}}\}_{\mathbf{m}}|\{p_{i,\mathbf{m}}\}_{\mathbf{m}})$ is the probability of observing the i'th histograms $\{n_{i,\mathbf{m}}\}_{\mathbf{m}}$ given $\{p_{i,\mathbf{m}}\}_{\mathbf{m}}$:

$$P_i(\{n_{i,\mathbf{m}}\}_{\mathbf{m}}|\{p_{i,\mathbf{m}}\}_{\mathbf{m}}) \equiv \frac{(N_i)!}{\prod_{\mathbf{m}}(n_{i,\mathbf{m}})!} \prod_{\mathbf{m}} (p_{i,\mathbf{m}}/Q_i)^{n_{i,\mathbf{m}}} \tag{3.7}$$

We define our log-likelihood function F from the relation by plugging equation 3.3 into 3.6:

$$\ln(A(\{\Omega_{i,\mathbf{m}}\}_{i,\mathbf{m}})) = -F(\{\Omega_{\mathbf{m}}\}_{\mathbf{m}}) + const. \tag{3.8}$$

where we treat all the quantities known during the analysis, such as $\{n_{i,\mathbf{m}}\}_{i,\mathbf{m}}$ and $\exp(-H_{i,\mathbf{m}})$, as constants. Our optimal estimate of $\{\Omega_{i,\mathbf{m}}\}_{i,\mathbf{m}}$ would be the one that minimizes F. Substituting equations 3.6 and 3.7 into 3.8 gives

$$F(\{\Omega_{\mathbf{m}}\}_{\mathbf{m}}) = -\sum_{\mathbf{m}} C_{\mathbf{m}} \ln \Omega_{\mathbf{m}} - \sum_i N_i \ln f_i \tag{3.9}$$

where $C_{\mathbf{m}} \equiv \sum_i n_{i,\mathbf{m}}$ is the number of counts in bin $\mathbf{m}$ and f_i is inverse of Q_i which takes the discretized form

$$f_i \equiv \frac{1}{\sum_{\mathbf{m}} \Omega_{\mathbf{m}} \exp(-H_{i,\mathbf{m}}) M} \tag{3.10}$$

and M is the d-dimensional volume of each bin, which we assume to be constant across different $\mathbf{m}$. If we take the derivative of F in equation 3.9 with respect to $\Omega_{\mathbf{m}}$, we have

$$\frac{\partial F}{\partial \Omega_{\mathbf{m}}} = -C_m/\Omega_m + \sum_i N_i f_i \exp(-H_{i,\mathbf{m}})M \tag{3.11}$$

By demanding equation 3.11 to zero we obtain the WHAM equations in the same form as those obtained before (1, 4, 8):

$$\Omega_{\mathbf{m}} = \frac{C_{\mathbf{m}}}{\sum_i N_i f_i exp(-H_{i,\mathbf{m}})M} \tag{3.12}$$

Since $\{f_i\}_i$ are functions of $\{\Omega_{\mathbf{m}}\}_{\mathbf{m}}$, equation 3.12 is a set of nonlinear equations which are usually solved by iteratively updating $\{f_i\}_i$ and $\{\Omega_{\mathbf{m}}\}_{\mathbf{m}}$ until self-consistency is achieved (2, 4). We can also show that F is a convex function that has one and only one global minimum.

To proceed, we re-interpret F as a new function of $\{f_i\}_i$:

$$F(\{f_i\}_i) = -\sum_i N_i \ln f_i - \sum_{\mathbf{m}} C_{\mathbf{m}} \ln \frac{C_{\mathbf{m}}}{z_{\mathbf{m}}} \tag{3.13}$$

or a function of $g_i \equiv \ln f_i$

$$F(\{g_i\}_i) = -\sum_i N_i g_i - \sum_{\mathbf{m}} C_{\mathbf{m}} \ln \frac{C_{\mathbf{m}}}{z_{\mathbf{m}}} \tag{3.14}$$

with $c_{i,\mathbf{m}} \equiv M \exp(-H_{i,\mathbf{m}})$ and $z_{\mathbf{m}} \equiv z_{\mathbf{m}}(\{g_i\}_i) \equiv \sum_i N_i c_{i,\mathbf{m}} \exp(g_i)$ for convenience. It's straightforward to show that

$$\frac{\partial F}{\partial g_i} = 0 \,\forall i \Leftrightarrow \frac{\partial F}{\partial \Omega_{\mathbf{m}}} \,\forall \mathbf{m} \tag{3.15}$$

To show that $F(\{g_i\}_i)$ is a convex function of $\{g_i\}_i$ and thus has one global minimum, we first give the first and second partial derivatives of F:

$$\frac{\partial F}{\partial g_i} = N_i(\exp(g_i) \sum_{\mathbf{m}} \frac{C_{\mathbf{m}} c_{i,\mathbf{m}}}{z_{\mathbf{m}}} - 1) \tag{3.16}$$

$$\frac{\partial^2 F}{\partial g_i \partial g_j} = \delta_{ij} N_i \exp(g_i) \sum_{\mathbf{m}} \frac{C_{\mathbf{m}} c_{i,\mathbf{m}}}{z_{\mathbf{m}}} - N_i N_j \exp(g_i + g_j) \sum_{\mathbf{m}} \frac{C_m c_{i,\mathbf{m}} c_{j,\mathbf{m}}}{z_{\mathbf{m}}^2} \quad (3.17)$$

where δ_{ij} is the kronecker delta. Let $\mathbf{B}$ be the Hessian matrix of F. We have $\mathbf{B} = \mathbf{C} - \sum_{\mathbf{m}} C_{\mathbf{m}} \mathbf{D}_{\mathbf{m}} \otimes \mathbf{D}_{\mathbf{m}}$ where $\mathbf{C}$ is a diagonal matrix with the main diagonal being $\{N_i \exp(g_i) \sum_{\mathbf{m}} \frac{C_{\mathbf{m}} c_{i,\mathbf{m}}}{z_{\mathbf{m}}}\}_i$, the first term in equation 3.17, and $\mathbf{D}_{\mathbf{m}} \equiv \{N_i \exp(g_i) c_{i,\mathbf{m}} / z_{\mathbf{m}}\}_i$. Next we'll show $\mathbf{a}^{\mathrm{T}} \mathbf{B} \mathbf{a} \geq 0 \; \forall \mathbf{a} \in \mathbb{R}^K$.

$$\begin{aligned}
\mathbf{a}^{\mathrm{T}} \mathbf{B} \mathbf{a} \geq 0 \Leftrightarrow \\
\mathbf{a}^{\mathrm{T}} \mathbf{C} \mathbf{a} \geq \sum_{\mathbf{m}} C_{\mathbf{m}} \mathbf{a}^{\mathrm{T}} \mathbf{D}_{\mathbf{m}} \otimes \mathbf{D}_{\mathbf{m}} \mathbf{a} \Leftrightarrow \\
\mathbf{a}^{\mathrm{T}} \mathbf{C} \mathbf{a} \geq \sum_{\mathbf{m}} C_{\mathbf{m}} (\mathbf{a} \cdot \mathbf{D}_{\mathbf{m}})^2 \Leftrightarrow \\
\sum_{\mathbf{m}} \frac{\sum_i a_i^2 N_i \exp(g_i) C_{\mathbf{m}} c_{i,\mathbf{m}}}{z_{\mathbf{m}}} \geq \sum_{\mathbf{m}} \frac{C_{\mathbf{m}} (\sum_i N_i c_{i,\mathbf{m}} \exp(g_i) a_i)^2}{z_{\mathbf{m}}^2} \Leftarrow \\
z_{\mathbf{m}} \sum_i a_i^2 N_i \exp(g_i) c_{i,\mathbf{m}} \geq (\sum_i N_i c_{i,\mathbf{m}} \exp(g_i) a_i)^2, \; \forall \mathbf{m} \Leftrightarrow \\
|\mathbf{s}_{\mathbf{m}}|^2 |\mathbf{t}_{\mathbf{m}}|^2 \geq |\mathbf{s}_{\mathbf{m}} \cdot \mathbf{t}_{\mathbf{m}}|^2 \qquad (3.18)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{s}_{\mathbf{m}} &\equiv \{\sqrt{N_i c_{i,\mathbf{m}} \exp(g_i)}\}_i, \\
\mathbf{t}_{\mathbf{m}} &\equiv \{a_i \sqrt{N_i c_{i,\mathbf{m}} \exp(g_i)}\}_i \qquad (3.19)
\end{aligned}$$

and the last step in equation 3.18 follows the Cauchy-Schwarz inequality. The twice differentiable function F has positive semidefinite Hessian for all $\{g_i\}_i$ and thus is convex.

4 Implementation

The purpose of this section is not to elaborate the details of my implementation, as I don't and will not have the intention/time to do so, but to provide a general understanding of how my implementation works or how one could

potentially implement their own WHAM program. The current WHAM program I wrote contains a series of application program interfaces (APIs) and a kernel. The APIs prepare and present the data to the kernel while the kernel is responsible of solving equation 3.12.

4.1 Kernel

There are three major steps in the kernel: 1) Caching $\{N_i \exp(-H_{i,\mathbf{m}})\}_i$ in memory; 2) Minimize function F in equation 3.14. This step is an optional; 3) Solve for equation 3.12 iteratively. (Note that the iteration is virtually skipped if step one results in a converged answer.) I will give more details about these steps as well as the user interface in my implementation in the following sections.

4.1.1 Caching $\{N_i \exp(-H_{i,\mathbf{m}})\}_i$

Because evaluating the exponential function is in general an expensive operation, we can precompute $\{\exp(-H_{i,\mathbf{m}})\}_i$ or more effectively, $\{N_i \exp(-H_{i,\mathbf{m}})\}_i$ before we do any minimization of iteration to save computation time. One caveat is that doing this could be memory demanding especially when we're dealing with high-dimensional reaction coordinate space with a large number of bins. My recommendation is to perform initial WHAM runs with a small number of bins to obtain a set of $\{g_i\}_i$, which can then be used to seed another round of WHAM with larger number of bins. My implementation will cache $\{N_i \exp(-H_{i,\mathbf{m}})\}_i$ in any case and it's thus worth watching for the memory requirement in the initial run.

4.1.2 Minimizing F in equation 3.14

As shown by Zhu and Hummer (8), it's more efficient to minimize F as a function of $\Delta g_{ij} \equiv g_i - g_j$. The form of such function was documented in their paper (8) except that we need to replace the harmonic potential with the generalized Hamiltonian in section 3. However, since we are dealing with a generalized form of Hamiltonian and typically multidimensional reaction coordinate, simply taking i and j as the nearest neighbors of sampling windows won't lead to high enough computational efficiency to justify the extra cost of calculating the gradient of F as compare to simply solving equation 3.12

iteratively. Although I've not come up with an analytic justification for pairing of i and j in general, I often found in practice that pairing i and j when the corresponding histograms maximally overlap with each other works very well. This can be seen from equation 3.16 where $\frac{\partial F}{\partial g_i}$ is predominantly determined by those j which have significant number of samples in the bins $\mathbf{m}$ where $c_{i,\mathbf{m}}$ is non trivial, i.e., i and j histograms overlap significantly. Thus, the first step to minimization of F is to construct a nearest neighbor list of g_i and g_j where the corresponding histograms overlap maximally. This results in a tree-like structure with each g_i being the node and each Δg_{ij} being the edge linking the two nodes g_i and g_j. Note that inside my implementation, such a tree is not implemented in the usual sense of tree structure in most computer program; it's rather a "aggregation" of nodes and edges that recognize one another by a map. The reason I did this is I couldn't find any reliable C++ tree structure implementation that meets my need. However, the construction of such a tree needs to be done only once and is rather cheap operation in most of the application to date (considering that most of the sampling simulations have less than 1000 sampling windows to be analyzed by WHAM). After we got all the Δg_{ij}, the calculation of F and its gradient is straightforward. It's thus easy to use any of of the minimization algorithm to minimize F. In my implementation, I used Polak-Ribiere conjugate gradient method with BrentâĂŹs line search (6) but the adaptation to other methods should be straightforward.

4.1.3 Solve equation 3.12

The procedure of solving equation 3.12 is very straightforward and will not be elaborated here. I just want to mention that in my implementation the evaluation of $\Omega_{\mathbf{m}}$ is in its own module module called the "DensityOfState" so that the "WHAM" class is only responsible for checking the self-consistency in equation 3.12. The "DensityOfState" class also provides some basic application program interfaces (APIs) such as the square bracket operator for computing other thermodynamic quantities as mentioned in section 2.

4.2 A note on floating point precision required by the WHAM kernel

In order to use minimization of F, a high-precision floating point type is often necessary to maintain numerical stability, especially in case where the free

energies of different states span several orders of magnitude. In the current implementation, I used the MPFR C++ library by Holoborodko (3) (version last updated in 2014-07-04 with my slight tweaks for C++11 compatibility). The floating point type in this library is called "mpreal". I provide a set of build scripts to build the "gwham" program with this library for the user's convenience. The precision of "mpreal" is embedded in these build scripts and is set at compilation time with the default of 20 digits for release build and 50 for debug build. For details about how to compile the programs, see section 5.5.

4.3 Application Program Interfaces (APIs)

The section describes the APIs for preparing necessary data for WHAM. The three major components of APIs are parsing the sampling data, creating the histogram and setting the Hamiltonian.

The sampling data is usually a time series generated from the simulation program. In the current implementation, the time series are not needed by WHAM kernel *per se* but are directly binned into histogram. The data parser is instantiated by a generic interfacial class (see below).

Histograms are represented by a multidimensional array implemented in the "gnarray" class. The dimensionality is determined at run time (by the output from the time-series data parser). The underlying data structure of the "gnarray" class is a map from the reaction coordinate values (or the bin) to a integer, i.e., the number of counts in the bin. Basic accessor APIs to the underlying array element are provided.

The Hamiltonian is basically a functor that maps the reaction coordinates to the potential energy. The class "Hamiltonian" provides APIs to compute the potential energy for all the reaction coordinate bins. It's generalized to handle a wide range of potential energy functions. The Hamiltonian is wrapped inside another class called "Ensemble", which provides generic interface to access the Hamiltonian of different thermodynamic ensembles. The "Ensemble" class along with its member "Hamiltonian" is instantiated by a generic interfacial class (see below).

To better organize the codes and provide more user-friendly interface, I also provide a generic interfacial class to link the 3 aforementioned components together. This interfacial class is called "MDP" (meaning Molecular Dynamics Parameter). It reads the software specific parameter file, create and instruct a time-series file reader and construct the necessary Hamilto-

nian. This way, a generic main function can be used to handle different output files from different simulation software. The main function's job is just to check for the consistency of command line arguments, to instantiate the "MDP" class and pass the parameter files to it, to use the time-series file reader from the "MDP" class to parse the time series into histogram and to call the WHAM solver. Therefore, there should be different derived classes from the "MDP" class corresponding to different simulation software. Due to the limited amount of time I have, I could only implement the derived class to handle the GROMACS version 4.6.3 (or potentially any version with compatible output format) format but I believe the extension is straightforward. The users are encouraged to write their own data parser; as for users who are not proficient C++ programmer, it's also straightforward to transform your data into the supported GROMACS format and user the GROMACS APIs directly (I'll give more details in section 5).

5 Usage

The user can use the WHAM kernel as a library in his/her own program or to use the "gwham" program to compute the thermodynamic quantity of interest.

5.1 Use the WHAM kernel

The user is strongly encouraged to read section 4 before using the WHAM kernel. The WHAM kernel is implemented in the C++ template class "WHAM" in "gwham.hpp". All the WHAM calculation is done upon instantiation of this class, i.e., all the function calls are directly inside or nested in the constructor of the "WHAM" class. The "WHAM" class takes three template arguments: 1) the pointer to the generic ensemble class; 2) histogram class; 3) multidimensional array class. Its constructor expects a list of arguments described in the comments right over its declaration in "gwham.hpp" and will not be elaborated here. The description is straightforward and the user should read it carefully. Since the "WHAM" class is a template, there's no need to link against it upon the compilation of whatever program the user is writing. However, the user needs to link his/her program against the "Hamiltonian" and "Ensemble" class objects (compiled from the respective "hamiltonian.cpp" and "ensemble.cpp" file) if he/she uses it.

5.1.1 Supported features

While the WHAM kernel is ignorant about the thermodynamic ensemble or the form of the corresponding Hamiltonian, the Ensemble class has only implemented NVE, NVT and NPT ensemble and the Hamiltonian class assumes the form of the Hamiltonian to be a linear combination of an arbitrary number of scalar functions. However, it's very straightforward to extend these two C++ classes to include other ensembles and other forms of Hamiltonians. To that end, a template functor class in "functor.hpp" is provided for implementing other forms of Hamiltonians.

In general, any thermodynamic observables described in equation 3.2 can be calculated once we have solved equation 3.12 but I only implemented in the WHAM class the calculation of PMF along the specified RCs. The extension to the computation of other observables can be implemented as member function of the WHAM class or as an external function of the density of state.

5.2 Use the "gwham" program

The "gwham" executable reads a set of simulation parameter files, parse the time series along the reaction coordinates (RCs) and compute the potential of mean force (PMF) along the RCs. For now, it can only read parameter and time series files in GROMACS format but the users can always transform their data into the GROMACS format in order to use it (see below).

5.2.1 File naming convention

Each simulation parameter file corresponds to one simulation window and tells the "gwham" program how to read the time series, what the RCs and the Hamiltonians are. The time series are the data output from the simulation software and usually are the RC values as a function of time. The simulation parameter files and the time series files must be in ASCII and must be named by the following convention: "prefix_window_run[suffix]", where "prefix" is any arbitrary string that identifies this set of simulations and is provided as a command line argument to the "gwham" program (see below) and "window" is an index that is consistent between a simulation parameter file and a consecutive set of time series file which are further indexed by "run". The index "run" is there for the users' convenience because usually

a simulation is run in chunks and produces a consecutive set of output files; "run" thus tells the program to identify each chunk by this unique index. E.g., due to the constraints on wall time allocation in a Linux cluster, a 30 μs simulation might be performed in 10 consecutive chunks with each being 3 μs and produce 10 consecutive set of output files and these files should be identified by the "run" index of $0, 1, \ldots, 9$. Both "window" and "run" are non-negative integers that start from 0. "[suffix]" is a string that's dependent on the specific MDP class (see section 4.3) and tells the program what file it's about to read. For example, if we have a set of **10** umbrella sampling simulations ("called foo") in GROMACS with each consists of **5** runs, the simulation parameter files should be named as: "foo_0.mdp", "foo_1.mdp", $\cdots$, "foo_9.mdp" and for the i'th "mdp" file, there should be M time series files: "foo_i_0x.xvg", "foo_i_0x.xvg", $\cdots$, "foo_i_4x.xvg" (where the "x.xvg" suffix is expected by the "GMXMDP" class).

5.2.2 Supported GROMACS free energy or PMF calculation schemes

The following GROMACS free energy calculation schemes are currently supported, i.e., the options in "mdp" files that can be understood by the "gwham" program:

1. PMF calculation via umbrella sampling
2. PMF calculation via umbrella sampling with Hamiltonian replica exchange
3. PMF calculation via umbrella sampling in expanded ensemble

Note that features 2 and 3 are only available for a in-house version of GROMACS 4.6.3 developed by me. Also note although "gwham" can handle cases where the temperatures and/or pressures in different simulation windows vary, e.g., in a temperature replica exchange simulation or in a umbrella sampling simulation coupled to temperature replica exchange, the program further expects a set of "ener.xvg" files in addtion to the "x.xvg" files and I have not tested such cases yet and the users might need to contact me for further instruction on this.

5.2.3 How to prepare input files for "gwham"

For users who want to use "gwham" to analyse the data from GROMACS, it's sufficient to rename their files following the convention in section 5.2.1. Alternatively, you can generate symbolic links to the data according to the naming convention so you can keep your original data without renaming them. The syntax of the simulation parameter files ("mdp" files) and the time series "x.xvg" files should following strictly those documented in the GROMACS 4.6 manual.

Alternative to the time series file, the user can also supply the histograms directly so that the program won't need to transform the time series into histograms. There should be one histogram for each state/ensemble. Note that the number of state doesn't necessarily equal the number of simulations, e.g., in a expanded ensemble simulation, and is automatically determined by the program when analysing the "mdp" files. The histogram file should contain $2N + 1$ columns, where N is the dimensionality of the RCs, with the first N columns denotes the indices of the multidimensional grid, the following N columns denotes the RC values of the corresponding grid point and the last column denotes the number of samples in that grid point. Only grid points with at least 1 sample needs to be specified.

For those who are not GROMACS users, you need transform your original data into GROMACS 4.6 format in order to use "gwham". If you're interested, I strongly recommend you to read this documentation as well as the GROMACS 4.6 manual carefully and you're welcomed to contact me if you need any help.

5.2.4 Command line arguments

It expects the list of arguments in the following order:

1. File name prefix for simulation parameter files and time series file (see section 5.2.1 for what this means)

2. The simulation parameter file suffix that tells "gwham" what types of simulation package you're using – for now this has to be the string "mdp", i.e., only GROMACS "mdp" files can be parsed.

3. Number of simulation windows (see section 5.2.1 for what this means)

4. Number of runs for each window (see section 5.2.1 for what this means). Note that you can simply put a large enough integer here so that the program will read as many runs as possible.

5. The indices (starting from 0) of the RCs that you want to calculate PMFs on. E.g., if you performed a 5-dimensional umbrella sampling and you want to calculate the PMF along the 1st, the 3rd and the 4th dimension, you should use "0 2 3" here.

6. Number of bins (deliminated by spaces) to use for histogramming for each of the dimension of the RCs. E.g., if your umbrella sampling simulation is along a 3-dimensional RC, you should use "$n_1 n_2 n_3$" to specify how many bins you want.

7. Upper bounds (deliminated by spaces) to use for histogramming for each of the dimension of the RCs.

8. lower bounds (deliminated by spaces) to use for histogramming for each of the dimension of the RCs.

9. Tolerance for the error in converging the solution to WHAM equation.

10. Which data point onwards to be regarded as the first sample in all the time series file. If you want to exclude the first N data points in all the time series, you should put N here. Note that the comments are not counted as valid data since they are automatically ignored by the parser.

11. How frequently we skip reading data points in the time series, i.e., only read the time series only every this number of data points.

12. Which data point to be the last one included in the analyse. All the samples after this are excluded.

13. The name of a "mdp" file (without the ".mdp" suffix) that specifies the thermodynamic parameters of a ensemble where we want the PMF to be calculated. E.g., if you want the PMF to be in 310 K instead of 300 K, you should prepare a "mdp" file with the temperature specified as 310 K.

14. A file contains the seeding values for the free energy in each state/ensemble. The free energy values must be specified in one line deliminated by spaces. The number of values should match the number of states in the simulations or otherwise the content of this file will be ignored. Note that the number of state doesn't necessarily equal the number of simulations, e.g., in a expanded ensemble simulation, and is automatically determined by the program when analysing the "mdp" files.

15. A boolean value (1 or 0) that tells the program if to perform minimization before WHAM iteration.

16. A boolean value (1 or 0) that tells the program if we can directly reads in histogram instead of time series data. The format histogram file is documented in section 5.2.3.

5.3 Output from the "gwham" executable

The followings are printed to standard output when running "gwham":

1. What type of command line arguments it's expecting

2. The arguments that are given

3. The names of the "mdp" file

4. The simulation parameters that it successfully reads in

5. The names of the "x.xvg" files (or potentially any time series files) or histogram files

6. The histograms that it creates or reads

7. The index of the histograms that contribute to each bin

8. The overlap matrix (in percentage) between any pair of histograms

9. Seeding free energies in each state

10. Seeding free energy difference between any pair of states

11. The free energy difference during optimization (printed out every 100 steps of minimization)

12. Indication of whether convergence has been met or not for minimization

13. Free energies of each state during WHAM iteration (printed out every 100 steps of iteration)

14. Indication of whether convergence has been met for WHAM iteration

15. PMF along the specified dimensions of RCs as well as the corresponding relative probability. This contains $2N + 2$ columns with the first $2N$ being the specification of the indices as well as the RC values for each grid point and the last two columns are the respective PMF and probability.

5.3.1 Examples

An example for using "gwham" executable is provided in the "example" directory along with this release. The data set is the PMF of 1 C16-KGGK lipopeptide binding to a POPE:POPG membrane as published in this paper (5). The PMF was calculated using umbrella sampling along the Z component of the distance between the center of mass of the lipopeptide and the membrane bilayer. The files are named with the prefix "md0" and the output are "gwham_mpreal_min.out", "gwham_mpreal_nomin.out" and "gwham_mpreal_min_hist.out" with "min", "nomin" and "hist" meaning minimization was performed, minimization was not performed and histograms were directly read, respectively. See section 5.5 for the meaning of "mpreal". The PMFs are at the end of the corresponding output files and I've excerpted the PMFs into the respective "pmf_mpreal_min", "pmf_mpreal_nomin" and "pmf_mpreal_min_hist".

5.4 Using the "mc" program

The "mc" program run a Monte Carlo (with the Metropolis-Hastings algorithm) umbrella sampling on the following potential:

$$H(x_0, x_1, \ldots, x_i, \ldots, x_{N-1}) = \sum_{i=0}^{N-1} a x_i^2 (b x_i^2 + c) \tag{5.1}$$

and recover H using the WHAM kernel. This is for testing the WHAM kernel. The "mc" program takes the list of arguments in the following order:

1. N in equation 5.1

2. Number of bins in histogramming each dimension of H or each x_i
3. Upper bounds in histogramming each dimension of H or each x_i
4. Lower bounds in histogramming each dimension of H or each x_i
5. Number of windows in the umbrella sampling along each dimension of H, i.e., x_i
6. Number of Monte Carlo sampling steps for all the windows
7. Step size when moving along all x_i in the Monte Carlo sampling
8. Tolerance for the error in converging the solution to WHAM equation
9. The indices (starting from 0) of the RCs, i.e., the list of i in x_i, where you want the PMF to be printed out. E.g., if you performed a 5-dimensional umbrella sampling and you want to calculate the PMF along the 1st, the 3rd and the 4th dimension, you should use "0 2 3" here.

The output of the "mc" program is similar to that of the "gwham" program except that it output which umbrella sampling it's currently performing on the fly.

5.5 Compilation

The build system I use in the current release is CMake. The CMake configuration file ("CMakeLists.txt") for building the "gwham" and "mc" executables is provided along with the release. I also provide 4 bash scripts for building the executables in Linux environment: a debug and a release builds as well as these two builds with "mpreal" (see section 4.2 for what this means) turned on. To use these bash scripts, download and unpack the source codes in a directory, say, /tmp, make a sub directory /tmp/build, copy one of the bash scripts in /tmp/build and execute it. The precision for "mpreal" is 20 for release build and 50 for debug build. The user can change the precision in the corresponding build script by a assigning different value to the "MPREAL-CXX" macro when executing the "cmake" command. Note that these bash scripts use gcc as the compiler and assume a default path ("/usr/local") to the STL library and the user might need to change them. To use the CMake

build system with the current release, you need to have CMake up to at least version 3.1.0; you also need to have "GNU" gcc up to at least version 4.9 if you are using it as the compiler. Other compilers with the complete set of C++11 features implemented might also be used although I have not tested those cases.

References

[1] John D. Chodera, William C. Swope, Jed W. Pitera, Chaok Seok, and Ken A. Dill. Use of the weighted histogram analysis method for the analysis of simulated and parallel tempering simulations. *J. Chem. Theory Comput.*, 3(1):26–41, 2007. doi: 10.1021/ct0502864. URL `http://pubs.acs.org/doi/abs/10.1021/ct0502864`.

[2] Alan Grossfield. WHAM: an implementation of the weighted histogram analysis method. URL `http://membrane.urmc.rochester.edu/content/wham/,version2.0.5`.

[3] Pavel Holoborodko. Mpfr c++. http://www.holoborodko.com/pavel/mpfr/, 2008-2014.

[4] Shankar Kumar, John M. Rosenberg, Djamal Bouzida, Robert H. Swendsen, and Peter A. Kollman. The weighted histogram analysis method for free-energy calculations on biomolecules. i. the method. *J. Comput. Chem.*, 13(8):1011–1021, 1992. ISSN 1096-987X. doi: 10.1002/jcc.540130812. URL `http://dx.doi.org/10.1002/jcc.540130812`.

[5] Dejun Lin and Alan Grossfield. Thermodynamics of antimicrobial lipopeptide binding to membranes: Origins of affinity and selectivity. *Biophys. J.*, 107(8):1862 – 1872, 2014. ISSN 0006-3495. doi: http://dx.doi.org/10.1016/j.bpj.2014.08.026. URL `http://www.sciencedirect.com/science/article/pii/S000634951400928X`.

[6] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing, 3rd edition*. Cambridge University Press, 2007.

[7] Benoît Roux. The calculation of the potential of mean force using computer simulations. *Computer Physics Communications*, 91 (1-3):275 – 282, 1995. ISSN 0010-4655. doi: http://dx.doi.org/10.1016/0010-4655(95)00053-I. URL `http://www.sciencedirect.com/science/article/pii/001046559500053I`.

[8] Fangqiang Zhu and Gerhard Hummer. Convergence and error estimation in free energy calculations using the weighted histogram analysis method. *J. Comput. Chem.*, 33(4):453–465, Feb 2012. doi: 10.1002/jcc.21989. URL `http://dx.doi.org/10.1002/jcc.21989`.