An implementation of WHAM: the Weighted Histogram Analysis Method Version 2.0.9

Alan Grossfield

Contents

1 Introduction

These programs (wham and wham-2d) implement the Weighted Histogram Analysis Method of Kumar, et al ("Multidimensional free-energy calculations using the weighted histogram analysis method", J. Comput. Chem., 16:1339-1350, 1995). The code generally follows the notation used by Benoit Roux ("The calculation of the potential of mean force using computer simulations", Comput. Phys. Comm., 91:275-282, 1995). Consult these papers for the theoretical background and justification for the method.

This code is available for download from my web page (http://membrane.urmc.rochester.edu/content/wham/). The code doesn't change all that often, but it's probably worth checking periodically. If you run into trouble using these programs, feel free to contact me (alan_grossfield@urmc.rochester.edu), and I'll try to help you. This code is available under the GPL and BSD licenses, as you prefer. The exception to this licensing is a set of routines from Numerical Recipes, which are not mine to give away.

If you use this code as part of an original piece of research, I'd appreciate a reference or acknowledgment. There's no publication to reference, so please use something like:
Grossfield, A, "WHAM: an implementation of the weighted histogram analysis method", http://membrane.urmc.rochester.edu/content/wham/, version XXXX
For that matter, just letting me know what you're using my code for would be nice, although again I don't insist upon it.

Suggestions and patches are welcome.

2 New in release 2.0.9

The is a bug-fix release; there was a memory allocation error in wham-2d that caused a segfault if the number of bins in the 2nd dimension was greater than the number of windows. Caught by Bastien Loubet from the Center for Biomembrane Physics in Denmark.

3 New in release 2.0.8

Major performance improvements to wham-2d (roughly 100x faster). The final answers should be indistinguishable from previous versions. Code con-

tributed by Nikolay Plotnikov from Stanford's Chemistry Department.

4 New in release 2.0.7

I messed up the ability to switch units from kcal to kJ: in addition to defining k_B in the header files, I also had a define in wham_2d.c. Caught by Yi Yao of UNC.

5 New in release 2.0.6

The primary new feature in version 2.0.6 is the ability to exclude regions of a 2D reaction coordinate from a calculation, which we're calling "masking". This is useful in cases where the PMF is being used to sample a pathway that is two dimensional, but where large areas of the 2D surface are either uninteresting or physically unattainable; for example, if the two reaction coordinates are RMSDs from 2 distinct structures, it is almost certainly impossible to sample the region where the RMS distance to each structure is very small. In practice, you could generally do the calculation anyway and just accept lots of NaNs and Infs floating around in the output, but sometimes that would fail and this way is far cleaner and robust. In this case, I chose to implement a very simple automasking procedure, such that any bin that has no data from any window is excluded. I suppose I could further generalize it, so that any bin with fewer than N points is excluded, but I haven't done that yet.

There's one other trivial but often requested feature in version 2.0.6: if you prefer to work in SI units, I've made it easier to switch energy units from kcal to kJ (the units of your reaction coordinate are, as always, up to you). It's a compile-time choice, but I figure that's fine because very few people switch units constantly. The switch is easy – before you build the code, edit `wham/wham.h` and `wham-2d/wham-2d.h` and change which version of the constant `k_B` is defined.

6 New in release 2.0.4 and 2.0.5

Version 2.0.5 is a trivial patch on 2.0.4, upping the length of lines allowed when reading files.

Release 2.0.4 has seen major revision to the bootstrap error analysis. I've known for a while that the way I computed the uncertainty in the free energy was suboptimal, since it just extrapolated from the uncertainty in the probability using the assumption that the fluctuations were gaussianly distributed. I knew this wasn't a great assumption, but Michael Shirts showed me data showing just how bad it was. So, in this version, we're doing something different. For 1D wham, we're directly computing the fluctuations in the PMF at each bin, aligning the pmfs such that their partition functions are all 1; this amounts to shifting them so that their Boltzmann-weighted free energies are the same. I think the errors make more sense now.

For 2D wham, I think doing this analysis uncovered some other flaws in how the error analysis is done, ones that to be honest I'm not totally sure how to fix. For now, I'm simply removing the option to do bootstrapping in 2D. I hope to put it back once I solve the problem, in which case I'll make another release.

Many thanks to Michael Shirts, for helpful discussions and contributing much of the code changes that went into this release.

7 Installation

Untarring wham.tgz will create a directory wham/, which in turn contains several directories (wham/ wham-2d/ doc/ nr/).

Before you build the code, you need to choose your units for energy. If you want kcal/mol, do nothing. If you prefer to work in kJ, you need to edit 2 files, `wham/wham.h` and `wham-2d/wham-2d.h`. In each case, near the top of the file you'll see a pair of lines that look like

```
#define k_B 0.001982923700 // Boltzmann's constant in kcal/mol K
//#define k_B  0.0083144621 // Boltzmann's constant kJ/mol-K
```

To get kJ, you simply remove the "//" from the second line and put it before the first line, and save the file. In principle, you can do just wham or just wham-2d, but I recommend against it for your own sanity.

To build the standard 1-D wham code on a machine which has gcc, you should

```
cd wham
make clean
make
```

(which will delete all object files and executables, and build the wham executable). By default, the Makefile uses the gcc compiler, but I also have flags present for the Intel compiler, plus the native compilers for Irix and Tru64. The latter two have not been tested recently, but ought to work, since at various times this code has been used successfully on various flavors of linux, MacOS X, AIX, Irix, and Tru64. If you find that you need to do anything special to make it work on your particular system, I'd appreciate it if you could let me know so I can add to the Makefile.

To build the 2-D version, say

```
cd wham-2d
make clean
make
```

Several other directories are also created (doc/, which you presumably found because you're reading this, and nr/, which contains a couple of files from Numerical Recipes). You don't need to do anything with these directories.

8 Command line arguments and file formats

To get a listing of the command line arguments for either wham or wham-2d, just run the command without any arguments. Optional arguments are included in brackets. Both programs will echo their command line into the output file, to help you figure out what you did.

8.1 wham

8.1.1 Command line arguments for WHAM

```
wham [P|Ppi|Pval] hist_min hist_max num_bins tol temperature numpad  \
        metadatafile freefile [num_MC_trials randSeed]
```

The first (optional) argument specifies the periodicity of the reaction coordinate. For a nonperiodic reaction coordinate (a distance, for example), it should be left out. "P" means that the reaction coordinate has a periodicity of 360, appropriate for angles. "Ppi" specifies a periodicity of 2*pi, appropriate for angles measured in radians. "Pval" specifies periodicty of some

arbitrary amount, val, which should be an integer or floating point number. For example, "P180.0" would be appropriate for an angle with twofold symmetry.

hist_min and hist_max specify the boundaries of the histogram. As a rule, all data points outside the range (hist_min, hist_max) are silently ignored. The only exception is that if an entire trajectory is outside the range, the program halts with an error message. The solution is to remove that file from the metadata file. hist_min and hist_max should be floating point numbers.

num_bins specifies the number of bins in the histogram, and as a result the number of points in the final PMF. It should be an integer.

tol is the convergence tolerance for the WHAM calculations. Specifically, the WHAM iteration is considered to be converged when no F_i value for any simulation window changes by more than tol on consecutive iterations. As the program runs, it prints the average change in the F values for the most recent iteration. Obviously, this number will be smaller than tol before the computation converges, because convergence is triggered by the largest change as opposed to the average.

temperature is a floating point number representing the temperature in Kelvin at which the weighted histogram calculation is performed. This does not have to be the temperature at which the simulations were performed (see below for discussion).

numpad specifies the number of "padding" values that should be printed for periodic PMFs. This number should be set to 0 for aperiodic reaction coordinates. It doesn't actually affect the calculation in any way. Rather, it just alters the final printout of the free energy, to make plotting of periodic reaction coordinates simpler. This is more important for wham-2d than wham.

metadatafile specifies the name of the metadata file. The format of this file is described below.

freefile is the name used for the file containing the final PMF and probability distribution.

num_MC_trials and randSeed are both related to the performance of Monte Carlo bootstrap error analysis. If these values are not supplied, error analysis is not performed. num_MC_trials should be an integer specifying the number of fake data sets which should be generated. randSeed is an integer which controls the random number seed – the value you pick should be irrelevant, but I let the user set it primarily for debugging purposes.

8.1.2 File formats

Each line of the metadata file should be blank, begin with a "#" (marking a comment), or have the following format:

```
/path/to/timeseries/file    loc_win_min    spring    [correl time] [temperature]
```

This first field is the name of one of the time series files (more on this in a moment). The second field, loc_win_min, is the location of the minimum of the biasing potential for this simulation, a floating point number. The third field, spring, is the spring constant for the biasing potential used in this simulation, assuming the biasing potential is of the format

$$V = \frac{1}{2}k(x - x_0)^2. \tag{1}$$

Many simulation packages, including TINKER, AMBER, and CHARMM, do not include the $\frac{1}{2}$ when they specify spring constants for their restraint terms. This is a common source of error (I'd love to change my code to match the other packages' behavior, but then experienced users who don't read the manual would get messed up). Also, the units for the spring constant must match those for the time series. So, if your time series is a distance recorded in Ångstroms, the spring constant must be in kcal/mol-Å^2. AMBER users should take care when using angular restraints: the specification and output of angles is in degrees, but AMBER's spring constants use kcal/mol-rad^2.

The fourth argument ("correl time") specifies the decorrelation time for your time series, in units of time steps. It is only used when generating fake data sets for Monte Carlo bootstrap error analysis, where it modulates the number of points per fake data set. This argument is optional, and is ignored if you don't do error analysis. If you're doing multiple temperatures but not bootstrapping, set it to any integer value as a placeholder, and it'll be ignored. See section 9.2 for more discussion about how to do bootstrapping.

Finally, the last (optional) field is the temperature for this simulation. If not supplied, the temperature specified on the command line is used. In the present version of the code, you must either leave the temperature unspecified for all simulations or specify it for all simulations.

The time series files must follow one of two formats, depending on whether the temperature was specified in the metadata file. If no temperature was specified, the file should contain two columns, where the first is the time (which isn't actually used), and the second is the position of the system

along the reaction coordinate. Both numbers should be in floating point format. Lines beginning with "#" are ignored as comments. Additional columns of data are ignored.

If the simulation temperature is specified, there must be a third column of data, containing the system's potential energy at that time point. It should be a floating point value.

8.1.3 Output

The first line of the output file contains echoes command line. The next line or two contain comments describing the periodicity used and the number of simulation windows present. While the calculation is running, it will print out lines that look like the following:

```
#Iteration 10:  0.106019
#Iteration 20:  0.062269
#Iteration 30:  0.039890
#Iteration 40:  0.027003
```

This specifies the current iteration number, and the average change in the F values for the current iteration. This number is not used for deciding when the calculation has converged; rather, the maximum change, as opposed to the average, is used.

Every 100 iterations, the current version of the PMF is dumped into the output file. These lines look like

```
-178.000000     0.014212        4909.138943
-174.000000     0.062631        4525.390035
-170.000000     0.227076        3432.434076
-166.000000     0.494262        2190.487110
-162.000000     0.817734        1271.708620
```

The first column is the value of the reaction coordinate, the second is the value of the PMF, and the third is the unnormalized probability distribution.

Once the calculation has converged, wham will produce output resembling

```
# Dumping simulation biases, in the metadata file order
# Window  F (free energy units)
# 0     0.000004
```

```
# 1     -4.166136
# 2     -3.241052
# 3     -4.475215
# 4     -6.324340
# 5     -7.128731
```

These are the final F values from the wham calculation, and can be used for computing weighted averages for properties other than the free energy.

You may have noticed that all of the lines except the free energies are preceded by "#". This allows you to check convergence of your wham calculation by simply plotting the output file in gnuplot. If the free energy curves have stopped changing, your tolerance is small enough.

If you specified a nonzero number of Monte Carlo bootstrap error analysis trials, you will see lines that resemble

```
#MC trial 0: 990 iterations
#MC trial 1: 973 iterations
#MC trial 2: 970 iterations
#MC trial 3: 981 iterations
#MC trial 4: 984 iterations
```

at the end of the file.

The free energy data file is written when the calculation converges, and resembles:

#Coor	Free	+/-	Prob	+/-
-178.000000	0.014386	0.000098	0.106389	0.000017
-174.000000	0.068560	0.000151	0.097128	0.000025
-170.000000	0.250825	0.000350	0.071496	0.000042
-166.000000	0.523786	0.000294	0.045186	0.000022

The first column is the value of the reaction coordinate, the second is the free energy. The third is the statistical uncertainty of the free energy (which is only meaningful if you performed Monte Carlo bootstrapping). The fourth and fifth columns are the probability and it's associated statistical uncertainty. Again, the latter is only meaningful if bootstrapping is performed. See section 9.2 for further discussion of error estimation.

8.2 wham-2d

8.2.1 Command line arguments

```
wham-2d Px[=0|pi|val] hist_min_x hist_max_x num_bins_x  \
        Py[=0|pi|val] hist_min_y hist_max_y num_bins_y  \
        tol temperature numpad metadatafile freefile \
        use_mask
```

The command line arguments largely have the same meaning as they do for the one dimensional wham program.

The periodicity arguments are not optional.

"Px" by itself indicates that the first dimension of the reaction coordinate has a period of 360. "Px=0" turns off periodicity. "Px=pi" specifies a period of 2*pi, and "Px=val" allows you to choose an arbitrary value for the period.

hist_min_x, hist_max_x, and num_bins_x behave exactly like hist_min, hist_max, and num_bins do in the 1 dimensional program.

Py, hist_min_y, etc., behave the same as Px, hist_min_x, etc., except they control the second coordinate of the PMF.

use_mask expects an integer value, and if its values is non-zero turns on the automasking feature, which causes bins for which there is no sample data to be excluded from the wham calculation.

8.2.2 File formats

As with regular 1 dimensional wham, each line of the metadata file should either be blank, begin with a "#", or have the following format

```
/path/to/timeseries/file loc_win_x loc_win_y spring_x spring_y [correl time] [temp]
```

This first field is the name of one of the time series files. loc_win_x and loc_win_y are the locations of the minimum of the biasing terms in the first and second dimensions of the reaction coordinate. spring_x and spring_y are the spring constants used for the biasing potential in this simulation, assuming the biasing potential is of the format

$$V = \frac{1}{2}(k_x(x - x_0)^2 + k_y(y - y_0)^2) \tag{2}$$

The sixth argument ("correl time") specifies the decorrelation time for your time series, in units of time steps. It is only used when generating

fake data sets for Monte Carlo bootstrap error analysis, where it modulates the number of points per fake data set. This argument is optional, and is ignored if you don't do error analysis. If you're doing multiple temperatures but not bootstrapping, set it to any integer value as a placeholder, and it'll be ignored. See section 9.2 for more discussion about how to do bootstrapping.

Finally, the last field is the temperature this simulation was run at. If not supplied, the temperature specified on the command line is used. In the present version of the code, you must either leave the temperature unspecified for all simulations or specify it for all simulations.

The time series files must follow one of two formats, depending on whether the temperature was specified in the metadata file. If no temperature was specified, the file should contain three columns, where the first is the time (which isn't actually used), and the second and third are the position of the system along the x and y reaction coordinates, respectively. Both numbers should be in floating point format. Lines beginning with "#" are ignored as comments. Additional columns of data are ignored.

If the simulation temperature is specified, there must be a fourth column of data, containing the system's potential energy at that time point. It should be a floating point value. See the section on replica exchange for more details.

8.2.3 Output

The output largely resembles that for wham, except with more columns. The first line echoes the command line, followed by a specification of the periodicity, and the number of windows. The iteration lines have the same meaning. When the current value for the PMF is dumped, the format looks like

```
-172.500000     -172.500000     1.968750        15.394489
-172.500000     -157.500000     2.574512        5.522757
-172.500000     -142.500000     3.147538        2.094142
-172.500000     -127.500000     3.505869        1.141952
```

where the first two columns are the values of the first and second dimensions of the reaction coordinate, the third column is the PMF, and the last column is the unnormalized probability.

Once the calculation has converged, wham will produce output resembling

```
# Dumping simulation biases, in the metadata file order
```

```
# Window  F (free energy units)
#0       -0.000004
#1       -0.156869
#2       -0.534845
#3       -2.445469
```

These are the final F values from the wham calculation, and can be used for computing weighted averages for properties other than the free energy.

If you specified a nonzero number of Monte Carlo bootstrap error analysis trials, you will see lines that resemble

```
#MC trial 0: 990 iterations
#MC trial 1: 973 iterations
#MC trial 2: 970 iterations
#MC trial 3: 981 iterations
#MC trial 4: 984 iterations
```

at the end of the file.

The free energy data file is written when the calculation converges, and resembles:

```
-232.500000     -232.500000     4.812986        0.003185        0.000001        0.000000
-232.500000     -217.500000     4.830312        0.003741        0.000001        0.000000
-232.500000     -202.500000     4.898622        0.001009        0.000000        0.000000
```

The first two columns are the locations along the first and second dimensions of the reaction coordinate. The third is the free energy, while the fourth is the statistical uncertainty in the free energy. The fifth and sixth columns are the normalized probability and its statistical uncertainty. The two uncertainty columns will be zero if you did not use Monte Carlo bootstrapping.

9 Discussion

9.1 Periodicity

Use of periodic boundary conditions only changes one thing in the code: when calculating the biasing potential from a simulation window for a specific bin in the histogram (the $w_j(X_i)$ values in Equation 8 of Roux's paper, cited above), the minimum image convention is applied. Thus, for a window with

the biasing potential centered at 175 degrees, the "distance" to the bin at -175 is 10 degrees, not 350 degrees.

The numpad argument on the command line is useful primarily for periodic reaction coordinates. It specifies a number of additional windows to be prepended and appended to the final output, such that the periodicity is explicitly visible in the free energy. So, if a calculation was done using 360 degree periodicity, 36 windows, with the reaction coordinate ranging -180 to 180, and numpad=5, a total of 46 values would be output, from -225 to +225. The numpad value has no effect at all on the values computed for the PMF and probability.

9.2 Monte Carlo Bootstrap Error Analysis

The premise of bootstrapping error analysis is fairly straightforward. For a time series containing N points, choose a set of N points at random, allowing duplication. Compute the average from this "fake" data set. Repeat this procedure a number of times and compute the standard deviation of the average of the "fake" data sets. This standard deviation is an estimate for the statistical uncertainty of the average computed using the real data. What this technique really measures is the heterogeneity of the data set, relative to the number of points present. For a large enough number of points, the average value computed using the faked data will be very close to the value with the real data, with the result that the standard deviation will be low. If you have relatively few points, the deviation will be high. The technique is quite robust, easy to implement, and correctly accounts for time correlations in the data. Numerical Recipes has a good discussion of the basic logic of this technique. For a more detailed discussion, see "An introduction to the bootstrap", by Efron and Tibshirani (Chapman and Hall/CRC, 1994). Please note: bootstrapping can only characterize the data you have. If your data is missing contributions from important regions of phase space, bootstrapping will not help you figure this out.

In principle, the standard bootstrap technique could be applied directly to WHAM calculations. One could generate a fake data set for each time series, perform WHAM iterations, and repeat the calculation many times. However, this would be inefficient, since it would either involve a) generating many time series in the file system, or b) storing the time series in memory. Neither of these strategies is particularly satisfying, the former because it involves generating a large number of files and the latter because it would

consume very large amounts of memory. My implementation of WHAM is very memory efficient because not only does it not store the time series, it doesn't even store the whole histogram of that time series, but rather just the nonzero portion.

However, there is a more efficient alternative. The principle behind bootstrapping is that you're trying to establish the various of averages calculated with N points sampling the true distribution function, using your current N points of data as an estimate of the true distribution. The histogram of each time series is precisely that, an estimate of the probability distribution. So, all we have to do is pick random numbers from the distribution defined by that histogram. Once again, Numerical Recipes shows us how to do it: we compute the normalized cumulant function, $c(x)$, generate a random number between 0 and 1 R, and solve $c(x) = R$ for x. Thus, a single Monte Carlo trial is computed in the following manner:

1. For each simulation window, use the computed cumulant of the histogram to generate a new histogram, with the same number of points.

2. Perform WHAM iterations on the set of generated histograms

3. Store the average normalized probability and free energy, and their squares for each bin in the histogram

There's a subtlety to how you compute fluctuations in the free energy estimates, since the potential of mean force is only defined up to a constant. I have chosen to align the PMFs by computing them from the normalized probabilities, which is effectively the same as setting the Boltzmann-averaged free energies equal. This is a somewhat arbitrary choice (for example, one could also set the unweighted averages equal), but it seems reasonable. If you want something bulletproof, use the probabilities and their associated fluctuations, which don't have this problem.

The situation is slightly more complicated when one attempts to apply the bootstrap procedure in two dimensions, because the cumulant is not uniquely defined. My approach is to flatten the two dimensional histogram into a 1 dimensional distribution, and take the cumulant of that. The rest of the procedure is the same as in the 1-D case. **In release 2.0.4, the option to do 2D bootstrapping has been commented out. I'm not sure if there's a programming problem, or implementing the better way**

of doing the 1D case simply revealed a deeper problem, but 2D bootstrapping is currently broken.

There is one major caveat throughout all of this analysis: thus far, we have assumed that the correlation time in time series is shorter than the snapshot interval. To put it another way, we've assumed that all of the data points are statistically independent. However, this is unlikely to be the case in a typical molecular dynamics setting, which means that the sample size used in the Monte Carlo bootstrapping procedure is too large, which in turn causes the bootstrapping procedure to underestimate the statistical uncertainty.

My code deals with this by allowing you to set the correlation time for each time series used in the analysis, in effect reducing the number of points used in generating the fake data sets (see section refss:format). For instance, if a time series had 1000 points, and you determined by other means that the correlation time was 10x the time interval for the time series, then you would set "correl time" to 10, and each fake data set would have 100 points instead of 1000. If the value is unset or is greater than the number of data points, then the full number of data points is used. Please note that the actual time values in the time series are not used in any way in this analysis; for purposes of specifying the correlation time, the interval between consecutive points is always considered to be 1.

The question of how to determine the correlation time is in some sense beyond the scope of this document. In principle, one could simply compute the autocorrelation function for each time series; if the autocorrelation is well approximated by a single exponential, then 2x the decay time (the time it takes the autocorrelation to drop to $1/e$) would be a good choice. If it's multiexponential, then you'd use the longest time constant. However, be careful: you really want to use the longest correlation time sampled in the trajectory, and the fluctuations of the reaction coordinate may fluctuate rapidly but still be coupled to slower modes.

It is important to note that the present version of the code uses the correlation times only for the error analysis and not for the actual PMF calculation. This isn't like to be an issue, as the raw PMFs aren't that sensitive to the correlation times unless they vary by factors of 10 or more.

9.3 Using the code for replica exchange simulations

One major application for the ability to combine simulations run at different temperatures is the analysis of replica exchange simulations, and if the email I've gotten over the last couple of years is any indication, it's a pretty common one. My code can be used for replica exchange, but I should start by admitting that it wasn't designed with it in mind, and may seem a bit clumsy.

First, the metadata file format has changed as of the November, 2007 release of the code. If you want to specify temperatures in the metadata file, you also have to specify the number of Monte Carlo points to use (if you're not using bootstrapping, you can safely set this to any integer). See section 8.1.2 for details.

In order to use wham with time series collected at different temperatures, the first thing to do is to follow the instructions given in section 8.1.2 regarding the format of the metadata and time series files, while setting the spring constants to 0. Indeed, for simple circumstances involving small systems this may be enough for you to make a successful calculation.

However, for large systems this simple approach will almost certainly get you nothing but a bunch of NaNs in your output. If this happens, the most likely candidate is either a overflow or underflow in the probability histograms. The reason is that the temperature-sensitive version of the code increments the histogram by $\exp(-E/k_BT)$ for each point (as opposed to counting each point as 1). Since the potential energies for condensed-phase molecular dynamics systems using standard force fields are typically of order -50,000 kcal/mol, the means we'd be taking the exponential of a very large number, which is a Bad Thing numerically.

However, in many circumstances one can work around this easily, by shifting the location of zero energy. The simplest procedure is to locate this lowest energy in any of the trajectories, and shift *all* of the energies in *all* of the trajectories such that the lowest (most negative) value is now zero. This will eliminate the overflows, since the largest contribution from an individual data point will now be 1.

However, shifting the energies upward can lead to a different set of problems, where a given simulation appears to have no probability associated with it, e.g. the sum of $\exp(-E/k_BT)$ for the trajectory underflows and is effectively zero. This can occur if the energies in the simulation are significantly higher than those in the lowest energy trajectories, which is expected

for condensed phase systems at high temperatures. Underflow in itself isn't a problem, but if that simulation is the only one which contributes to a bin in the histogram (or more generally if all of the simulations which sample a given bin have zero overall weight), the result will be a division by zero causing the probability to be NaN or Inf.

Solving this problem is sometimes quite simple: reshift the energies by a few kcal/mol, such that the lowest energy is moderately small instead of zero (say -5 kcal/mol). If the problem is just numerical underflow, a small shift may be sufficient to make the problem numerically well-behaved. However, if the relevant portion of the histogram really is unaccessible except at high temperature, then there may be no way to fix the problem, short of running an additional umbrella-sampled trajectory.