3-1-2019

Implementation of the Bin Hierarchy Method for Restoring a Smooth Function from a Sampled Histogram

Olga Goulko
Boise State University

Alexander Gaenko
University of Michigan

Emanuel Gull
University of Michigan

Nikolay Prokof'ev
University of Massachusetts

Boris Svistunov
University of Massachusetts

Publication Information

Goulko, Olga; Gaenko, Alexander; Gull, Emanuel; Prokof'ev, Nikolay; and Svistunov, Boris. (2019). "Implementation of the Bin Hierarchy Method for Restoring a Smooth Function from a Sampled Histogram". *Computer Physics Communications, 236*, 205-213.
http://dx.doi.org/10.1016/j.cpc.2018.09.019

Implementation of the Bin Hierarchy Method for restoring a smooth function from a sampled histogram

Olga Goulko[a,b,*], Alexander Gaenko[c], Emanuel Gull[c], Nikolay Prokof'ev[a,d], Boris Svistunov[a,d,e]

[a] *Department of Physics, University of Massachusetts, Amherst, MA 01003, USA*
[b] *Present address: Raymond and Beverly Sackler School of Chemistry and School Physics and Astronomy, Tel Aviv University, Tel Aviv 6997801, Israel*
[c] *Department of Physics, University of Michigan, Ann Arbor, MI 48109, USA*
[d] *National Research Center "Kurchatov Institute," 123182 Moscow, Russia*
[e] *Wilczek Quantum Center, School of Physics and Astronomy and T. D. Lee Institute, Shanghai Jiao Tong University, Shanghai 200240, China*

Abstract

We present `BHM`, a tool for restoring a smooth function from a sampled histogram using the bin hierarchy method. The theoretical background of the method is presented in [1]. The code automatically generates a smooth polynomial spline with the minimal acceptable number of knots from the input data. It works universally for any sufficiently regular shaped distribution and any level of data quality, requiring almost no external parameter specification. It is particularly useful for large-scale numerical data analysis. This paper explains the details of the implementation and the use of the program.

PROGRAM SUMMARY

Manuscript Title: Implementation of the Bin Hierarchy Method for restoring a smooth function from a sampled histogram
Authors: Olga Goulko, Alexander Gaenko, Emanuel Gull, Nikolay Prokof'ev, Boris Svistunov
Program Title: BHM
Journal Reference:
Catalogue identifier:
Licensing provisions: GPLv3
Programming language: C++
Operating system: Tested on Linux
RAM: 1–5 MB
Keywords: Data analysis, Function restoration, Spline fitting, Histogram, Smoothing
Classification: 4.9
External routines/libraries: CMake, GSL
Nature of problem: Restoring a smooth function from a sampled histogram in an efficient, reliable and automatized way is crucial for numerical and experimental data analysis.
Solution method: To make use of all information contained in the sampled data, the BHM algorithm generates a hierarchy of overlapping bins of different sizes from the initially supplied fine histogram. The bin hierarchy is fitted to a polynomial spline with the minimal acceptable number of knots, the positions of which are determined automatically. The output is a smooth function with error band.
Running time: Typically less than a second

1. Introduction

Numerical approaches to problems in condensed matter and quantum many-body physics often involve generating data points according to an unknown probability density $f(x)$, which needs to be restored from the sampled data. The amount of data generated in large-scale quantum Monte Carlo simulations is usually so large that it is impossible (or at least impractical) to store the complete list of sampled data points x_i, in order to

*Corresponding author.
E-mail address: goulko@umass.edu

use density estimation protocols [2–4] to recover $f(x)$. Instead, data points are typically collected into a histogram, the histogram bins representing integrals over the sampled distribution. This does not involve any significant loss of information, as long as the bins are sufficiently small to resolve the features of the distribution (which is always possible provided that $f(x)$ is sufficiently smooth). More sophisticated sampling methods exist, which retain more information about the individual points, but these are in general less efficient and require a case-dependent implementation. We provide a universal and efficient program to restore a smooth distribution, which uses the standard histogram as input.

`BHM` is an implementation of the bin hierarchy method, introduced in [1]. It is

1. unbiased:
   - utilizes all relevant information contained in the data;
   - non-parametric fit automatically adjusts to data quality;
   - provides maximally featureless solution (least acceptable number of spline knots);
2. efficient:
   - based on regular histogram, which is efficient to sample;
   - fast analysis;
3. automatic:
   - very little user input;
   - no adjustment for different types of sampled functions;
   - no adjustment with simulation time as more data is collected.

The paper is organized as follows. The general problem setup is presented in Sec. 2. In Sec. 3, we give an overview of the algorithm. We explain how to use the program in Sec. 4, giving a detailed explanation of the input and output formats, as well as possible parameter specifications. Several examples are presented in Sec. 5.

2. Problem setup

The central object in `BHM` is a smooth function $f(x)$ defined on a bounded domain $\mathcal{D}$. Statistical sampling with a probability density $p(x)$ is performed to generate samples for $f(x)$ according to $f_j = f(x_j)/p(x_j)$ with p-distributed x_j. In the simplest case, when $f(x)$ itself is a normalized probability distribution, $p(x) = f(x)$ can be chosen, implying $f_j = 1$. The samples are binned into a histogram with 2^K bins. We are interested in restoring a smooth function $\tilde{f}(x)$ from this histogram.

Each histogram bin i with bin boundaries $x_{i,\min}$ and $x_{i,\max}$ represents the stochastic integral

$$I_i = \int_{x_{i,\min}}^{x_{i,\max}} f(x)dx \tag{1}$$

through the following relations:

$$I_i = \bar{f}_i \frac{N_i}{N}, \tag{2}$$

$$M_2(I_i) = M_2(f_i) + \bar{f}_i^2 \frac{N_i(N-N_i)}{N}, \tag{3}$$

$$\mathrm{Var}(I_i) = \frac{M_2(I_i)}{N-1}, \tag{4}$$

$$\delta I_i = \sqrt{\frac{\mathrm{Var}(I_i)}{N}}, \tag{5}$$

where the "scaled variance" $M_2(f_i) = (N_i - 1)\mathrm{Var}(f_i)$ is the sum of squares of differences from the mean, N_i is the number of samples in bin i and N the total number of samples. Note that in the simplest case $p(x) = f(x)$ the above quantities are determined through N_i and N alone.

The goal is to find a function $\tilde{f}(x)$ whose *integrals* over different parts of the domain $\mathcal{D}$ agree with the sampled integrals. Working with integrals rather than interpolated function values allows us to include combinations of histogram bins into the fitting. Rebinning data to larger bin sizes leads to a reduction of statistical noise, while retaining small bins results in a higher resolution due to smaller discretization errors.

The resulting fit $\tilde{f}(x)$ is a polynomial spline of order m, where m is the highest power with non-zero coefficient. The spline function and its derivatives up to order $m-1$ are continuous, to account for the smoothness of the original $f(x)$.

3. Overview of the algorithm

In this section we give a brief overview of the algorithm. More details on the theoretical background of the method can be found in [1]. A flowchart of the algorithm is shown in Fig. 1.

- The algorithm starts from a list of 2^K histogram bins supplied in an input file (for a detailed format description, see Sec. 4). Typical values of K are 7 – 15. It should be noted that the bins are not required to have the same size; however, in practice there is no need to have variable size bins. The bins must not overlap or leave gaps.

- From this input the code generates a hierarchy of histogram bins of increasing size. Combining two neighboring bins of the 2^K initial bins leads to 2^{K-1} larger bins with, on average, twice as many entries. Successive repetitions of this rebinning result in a hierarchy of levels with $2^{K-2}, \ldots, 2, 1$ bins on each level, respectively, where the final level consists of one bin over the entire domain containing an average over all sampled data. Bins that do not contain enough data for meaningful statistic, i.e. when the bin counter N_i is smaller than a user defined minimal value, are excluded from the fitting process. Likewise, levels that do not contain enough usable bins (the minimal fraction can be defined by the user) are also excluded. This implies that in general fitting starts with a level $K' > K$ so that the original binning can be chosen to be very fine without introducing noise into the final fit. For bins that will be used for fitting, the bin integrals and their errors are computed via Eqs. (2),(3), (4) and (5).

- The code checks if the data is compatible with zero on the whole domain. There is an option not to proceed with the fit if this is the case. This feature is particularly useful for data suffering from a severe sign problem.

- The next step is fitting a spline of order m on the given spline interval division. The starting point is one spline interval, which means that one polynomial is fitted on the whole domain. The fit minimizes

$$\sum_{n=0}^{K} \frac{\chi_n^2}{2^n}, \tag{6}$$

  where χ_n^2 is defined for bins on hierarchy level n in the usual way.

- Afterwards the goodness of fit is evaluated on each hierarchy level individually. The criterion is

$$\frac{\chi_n^2}{\tilde{n}} \leq 1 + T \quad \frac{\overline{2}}{\tilde{n}}, \tag{7}$$

  where T is the fit acceptance threshold (input parameter) and $\tilde{n}$ the number of bins on level n that were used for fitting. The expression $\sqrt{2/\tilde{n}}$ corresponds to one standard deviation of the χ^2-distribution.

- If at least one level fails the global goodness-of-fit check, the goodness-of-fit is then evaluated on each spline interval separately (again level by level). Spline intervals on which the fit was acceptable remain unchanged, while the others are split into two parts, by introducing a spline knot in the middle ("number of bins"-wise).

- If any of the resulting intervals is too small, meaning that there is not enough data to fit on that interval, the code exits without having produced an acceptable spline. Otherwise the BHM fit is repeated on the new interval division.

- Once an acceptable spline has been found, there is an option to refit the data on the same interval division with an additional constraint that aims to minimize the jump in the highest derivative.

- The resulting BHM spline is output (spline coefficients and error coefficients). In addition, the spline values can be output evaluated on a grid.

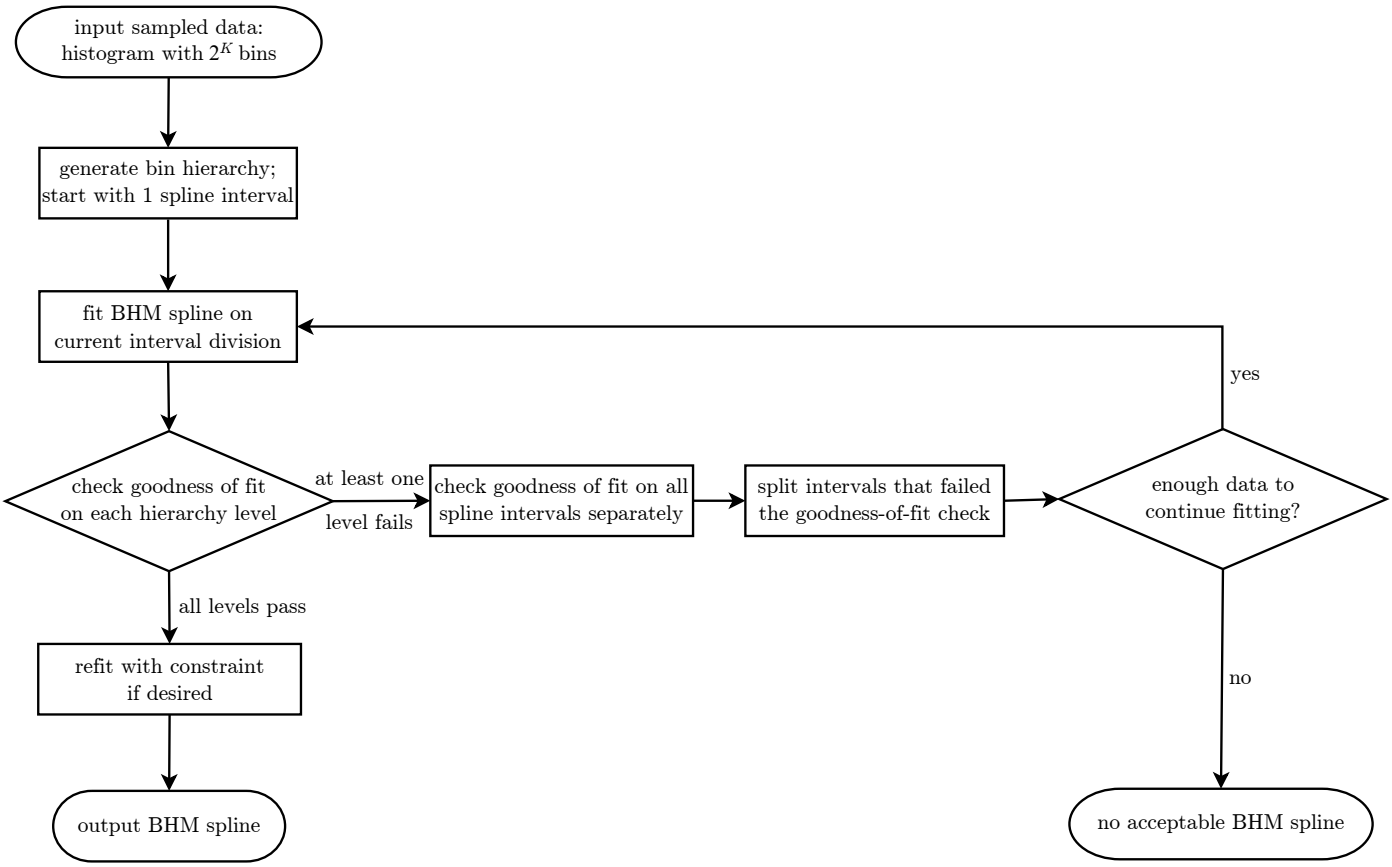

Figure 1: Flowchart of the algorithm

4. Input and output

4.1. Running the program

Instructions for compiling the program and executing unit tests can be found in the README file.

The program executable requires 1 argument, the name of the parameter file, e.g.:

```
$ ./bhm in.param
```

In particular, the parameter file determines the name of the input file with the histogram data and the name of the output file for the BHM spline (see below).

As a special case, if the parameter file name is an empty string, the default parameters will be used which are suitable for most applications:

```
$ ./bhm "" <histogram.dat >spline.dat
```

In this case, the histogram data input is expected to be provided at the standard input, and the results will be printed to the standard output. In the example above, the standard input is redirected from file `histogram.dat`, and the standard output is redirected to file `spline.dat`.

Without an argument, the program prints a short help message and exits.

4.2. Histogram input format

The input histogram data is text-based, line-oriented, and has the following format:

$$
\begin{array}{lll}
A \;\; N_{\text{exc}} & & 1 \\
x_{1,\min} \quad N_1 \;\; \bar{f}_1 \;\; M_2(f_1) & & 2 \\
x_{2,\min} \quad N_2 \;\; \bar{f}_2 \;\; M_2(f_2) & & 3 \\
\cdots\cdots & & 4 \\
x_{i,\min} \quad N_i \;\; \bar{f}_i \;\; M_2(f_i) & & 5 \\
\cdots\cdots & & 6 \\
x_{\max} & & 7
\end{array}
$$

where the first line specifies an overall normalization factor A and the number N_{exc} of samples outside of the histogram bounds. The normalization step is omitted if either $A = 1$ or $A = 0$. Otherwise, all values $\bar{f}_i$ and $M_2(f_i)$ are divided

by A and A^2, respectively, before constructing the BHM fit. The value $N_{\rm exc}$ is used to calculate the total number of samples $N = N_{\rm exc} + \sum_i N_i$, which is needed for Eqns. (2–5). $N_{\rm exc}$ can be zero.

Starting from the second line, each line, except the last one, contains 2 or 4 blank-separated values, specifying the left bin boundary, the number of samples in the bin, and, optionally, mean value and scaled variance. For example, line 5 of the listing corresponds to a bin i with the left boundary $x_{i,\min}$, number of samples N_i, mean value $\bar{f}_i$ and scaled variance $M_2(f_i)$ (see Eqns. 2–5). If the mean value and the scaled variance are both omitted, they are assumed to be $\bar{f}_i = 1$ and $M_2(f_i) = 0$, which corresponds to only ever adding 1 to bin counters, or in other words $p(x) = f(x)$. The last line of the file (line 7 of the listing) must contain a single entry $x_{\max}$, the right boundary of the last bin.

The numbers $x_{1,\min} \ldots < x_{i,\min} \ldots < x_{\max}$ must form a strictly monotonically increasing sequence, corresponding to non-overlapping, finite-size bins with no gaps. In the current implementation, the number of bins must be a power of 2 (in the later versions we may remove this limitation).

It is important to note that all sampled data and variances are assumed to be *uncorrelated*. If correlations are present, they have to be removed prior to the BHM fit, for example through appropriate blocking analysis or by scaling the variances with the estimated correlation factor.

4.3. Input parameter format

The input parameter file is a text-based, line-oriented file that has a `key = value` format. An example input is shown in Fig. 2. The keys are case-insensitive; the string values may be enclosed in quotes; the `#` symbol starts a comment which is ignored until the end of the line. The meaning of each parameter is indicated in the figure in the corresponding comment. Below we provide more detailed explanations for some of the parameters.

`DataPointsMin` in line 1 specifies the minimal number of data points that a bin must contain in order to be used for fitting. Bins that contain fewer sampled points are ignored (but still contribute in combination with other bins at higher hierarchy levels). `DataPointsMin` must be at least 10, in order to ensure that meaningful statistics can be made from the data. The default value is 100. If a hierarchy level does not contain enough usable bins (the minimal number is given by the parameter `UsableBinFraction` in line 7, times the total number of bins on that level) then this level and all subsequent levels are completely omitted from the fitting.

The maximal possible number of interval divisions is determined by the parameter `MinLevel` in line 3. For example, if there are 2^K elementary bins, `MinLevel=2` means that the smallest possible spline intervals coincide with the bins on hierarchy level $K-2$. `MinLevel` must be at least 2 (corresponding to a total of at least 1+2+4=7 bins per interval); moreover, `MinLevel` must be large enough to ensure that the fit is underdetermined for each interval.

The fit acceptance threshold T (lines 4-6) can be either set to a fixed value, or to a range of values between `Threshold` and `ThresholdMax`. In the latter case, a BHM fit is first attempted with the smallest value `Threshold`. If no acceptable fit is found, the threshold value is successively increased in `ThresholdSteps` equidistant steps, until either an acceptable fit is produced or `ThresholdMax` is reached. Setting `ThresholdMax` to be smaller or equal to `Threshold` and/or setting `ThresholdSteps=0` corresponds to only using one fixed value of T. Note that threshold values that are too low can result in overfitting (too many spline pieces) or the failure to produce an acceptable fit. Values that are too high can result in underfitting (too few spline pieces and a poor fit with underestimated error bars). These issues are illustrated in Example 5.2. The value $T = 2.0$ is good generic choice. Specifying a range of threshold values reduces the statistical chance that there is no acceptable BHM fit with a given threshold, even though the data quality is adequate. The default range between $T = 2.0$ and $T = 4.0$ is suitable for most data sets.

Generally, the default parameter values in the example file are suitable for all types of sampled

Parameter File in.param

```
DataPointsMin=100              #minimal number of data points per bin
SplineOrder=3                  #spline order
MinLevel=2                     #minimal number of levels per interval
Threshold=2.0                  #minimal goodness of fit threshold
ThresholdMax=4.0               #maximal goodness of fit threshold
ThresholdSteps=4               #number of steps to take from Threshold to ThresholdMax
UsableBinFraction=0.25         #minimal proportion of good bins for a level to be used
JumpSuppression=false          #suppress highest order derivative (the error is unreliable)
Verbose=true                   #verbose output
PrintFitInfo=true              #print the fit info
FailOnBadFit=true              #do not proceed if the fit is bad
FailOnZeroFit=true             #do not proceed if the data is consistent with zero
Data="histogram.dat"           #location of histogram input data
OutputName="spline.dat"        #location of the output file
GridOutput="spline_plot.dat"   #location of the spline-on-a-grid output
GridPoints=1024                #number of points for spline-on-a-grid output
```

Figure 2: Sample parameter file

functions, and hence there is no need to change any of the parameters unless specifically desired.

4.4. Output format

The default verbose output is printed to standard error and contains auxiliary information such as values of the input parameters, a brief description of the input histogram, and the log of the fitting process. The fitting log is described in detail in Example 5.1. If requested by the `PrintFitInfo` input parameter, information about the final fit is also printed to the standard output.

The output of the program is both human and machine-readable, and has the following text-based, line-oriented, blank-separated format:

```
# Arbitrary comments                                    1
# ...                                                   2
m  s                                                    3
x_1  x_2  ...  x_s                                      4
# spline piece 1                                        5
a_0  a_1  a_2  ...  a_m                                 6
ε_0  ε_1  ε_2...  ε_2m                                  7
...                                                     8
# spline piece i                                        9
a_0  a_1  a_2  ...  a_m                                 10
ε_0  ε_1  ε_2  ...  ε_2m                                11
# spline piece (i+1)                                    12
...                                                     13
```

Any lines at the beginning of the file that start with `#` are considered comments and are ignored. The first significant line of the file (line 3 of the listing) specifies the spline polynomial order m and the number of splines pieces s; the next line (line 4 of the listing) lists all $(s+1)$ spline piece boundaries $x_1, \ldots, x_{s+1}$. The following lines form s sections describing each spline piece $\tilde{f}_i$, for $i = 1 \ldots s$. Each section (lines 5–7, 9–11 of the listing) consists of 3 lines:

1. Header (starts with `#`) specifying the spline piece number (i),
2. $(m+1)$ numbers specifying the spline piece coefficients $a_0 \ldots a_m$ $(\tilde{f}_i(x) = \sum_{k=0}^{m} a_k x^k)$,
3. $(2m+1)$ numbers $\varepsilon_0 \ldots \varepsilon_{2m}$ specifying the error bar $E_i(x) = \sqrt{\sum_{k=0}^{2m} \varepsilon_k x^k}$.

4.5. Plotting the resulting spline

The simplest way to plot the resulting spline is to use the provided Python3 script `bhm_spline.py`, as follows:

```
$ python3 bhm_spline.py spline.dat
```

On the other hand, it may be convenient to customize the plot and/or compare it with a known function, or plot it interactively (e.g., from a Jupyter notebook). For this purpose the script can be imported as a module that provides a BHM Spline class. The following listing demonstrates a possible way of using the module.

```
import numpy as np                                    1
import matplotlib.pyplot as plt                       2
from bhm_spline import BHMSpline                      3
                                                      4
spline=BHMSpline("spline.dat")                        5
x=np.linspace(*spline.domain())                       6
# reference function:                                 7
def fn(x): return (x**4-0.8*x*x)                      8
   /0.171964
# plot the spline and the reference:                  9
plt.plot(x,spline(x), x,fn(x))                        10
# plot the errorbar:                                  11
plt.plot(x,spline.errorbar(x))                        12
# plot the spline with errorbars:                     13
spline.plot()                                         14
# plot the spline and a reference:                    15
spline.plot(fn)                                       16
# plot difference between spline and                  17
   reference with error bar:
spline.plot_difference(fn)                            18
```

In line 3 the class `BHMSpline` is imported; line 5 creates the object representing the spline. In line 6 an interval of x-values is created corresponding to the domain of the spline. Line 8 defines a reference function to compare with the spline. In line 10 the spline and the reference function are plotted using the `Matplotlib` plotting library; in line 12 the error bar $E(x)$ is plotted. The class also provides a convenience plotting method: when called without arguments (as on line 14), the spline is plotted along with the error bars; when a function is passed as an argument (line 16), its graph is plotted also. It is also possible to plot the difference between the spline and the reference function with error bar (line 18).

4.6. Grid output

If the `GridOutput` parameter in the parameter file is set to a non-empty filename, the program also outputs to the specified file the values and the error bars of the spline computed on a one-dimensional grid of points. A plotting program, such as `gnuplot`, can then be used to plot the generated function and the error bars and to compare them with a reference function; for example:

```
$ gnuplot                                             1
gnuplot> quartic(x)=(x**4-0.8*x*x)                    2
   /0.171964
gnuplot> plot "spline_plot.dat" with                  3
   errors
gnuplot> replot quartic(x)                            4
```

In this example, line 1 of the listing starts the `gnuplot` program; line 2 defines a reference function (quartic polynomial); line 3 plots the grid output file generated by `BHM`; and line 4 plots the reference function on the same graph.

5. Examples

In this section we present three detailed examples of the features of `BHM` illustrated on different distributions $f(x)$. We provide a program to generate the input data for these examples (as well as for several additional test functions). Calling the program without arguments:

```
$ ./generator
```

prints a brief help message, which includes a list of the functions supported by the program.

Calling the program with a single file argument:

```
$ ./generator generator.param
```

generates the histogram data for a given analytical function according to the parameters listed in the `generator.param` file. For all examples discussed below, the parameters are the same as shown in the example generator parameter file shown in Fig. 3 (including the random number generator seed), except when stated otherwise.

Calling the program as:

```
$ ./generator -python name
```

(where *name* is the name of the function, possibly abbreviated) prints the Python code that corresponds to the function, which is convenient for

Parameter File generator.param

```
SampleSize=10000          #total number of points sampled                    1
Function=exponential      #name of test function                             2
PowerBins=10              #generates uniform histogram with 2^PowerBins bins 3
RandomSeed=956475         #specify random seed (0 means random initialization) 4
Output="histogram.dat"    #histogram output file (missing means "standard output") 5
GridOutput="function.dat" #output test function on a grid into this file (if provided) 6
GridPoints=1024           #number of grid points for test function output    7
```

Figure 3: Sample parameter file to generate example input

plotting the analytical function against the approximating spline in an interactive Python environment (as has been discussed in subsection 4.5).

If the `GridOutput` parameter in the parameter file is set to a non-empty filename, the program also outputs the values of the function computed on a one-dimensional grid to the specified file; a plotting program, such as `gnuplot`, can then be used to plot the generated function; for example:

```
$ gnuplot                                                    1
gnuplot> plot "function.dat" with lines                      2
gnuplot> replot "spline_plot.dat" with                       3
    errors
```

In this example, line 1 of the listing starts the `gnuplot` program; line 2 plots the generated function; and line 3 plots the content of the `spline_plot.dat` generated by `BHM` as discussed in subsection 4.6.

5.1. Example 1

This example demonstrates BHM fits for different choices of spline order m.

The original function is a quartic polynomial (`Function=quartic_polynomial`):

$$f(x) = \alpha(x^4 - 0.8x^2). \tag{8}$$

Because $f(x)$ changes sign, sampling on the interval $[-1, 1]$ is performed with the probability density $p(x) = |f(x)|$ and $\alpha = 0.171964$ is chosen to ensure normalization of $p(x)$ on this interval.

The histogram data is fitted with `BHM` using the default parameters, with the exception of `SplineOrder` which is set to 3, 4, and 5 respectively. The fit results are shown in Fig. 4. From the output files `"spline.dat"` it can be seen that the cubic spline has four spline pieces; the quartic spline has one spline piece, as expected; the quintic spline also has one spline piece, its coefficients up to quartic order are similar to the ones obtained via quartic fit, and its highest spline coefficient is small.

We explain in detail the verbose output for the cubic fit $m = 3$. At the beginning of the output, the fit parameters are listed, as well as general information about the input histogram. Then follows information about the goodness-of-fit at the different fitting stages:

```
...                                                          1
BHM fit:                                                     2
Begin BHM fitting with threshold T = 2                       3
Checking separate chi_n^2/n in spline fit                    4
level  n      chi_n^2/n      max chi_n^2/n                   5
0      1        9.7585       3.8284                         6
1      2        2.7736       3.0000                         7
2      4        1.7636       2.4142                         8
3      8      832.1519       2.0000                         9
4      14     539.3412       1.7559                         10
5      24     210.2518       1.5774                         11
6      41     118.5452       1.4417                         12
7      54      67.7739       1.3849                         13
Checking interval 0 (order: 0, number: 0)                    14
0      1        9.7585       3.8284                         15
This interval fit is not good                                16
Checking separate chi_n^2/n in spline fit                    17
level  n      chi_n^2/n      max chi_n^2/n                   18
0      1        0.0020       3.8284                         19
1      2        0.0006       3.0000                         20
2      4        1.6409       2.4142                         21
3      8        7.0923       2.0000                         22
4      14       7.8734       1.7559                         23
```

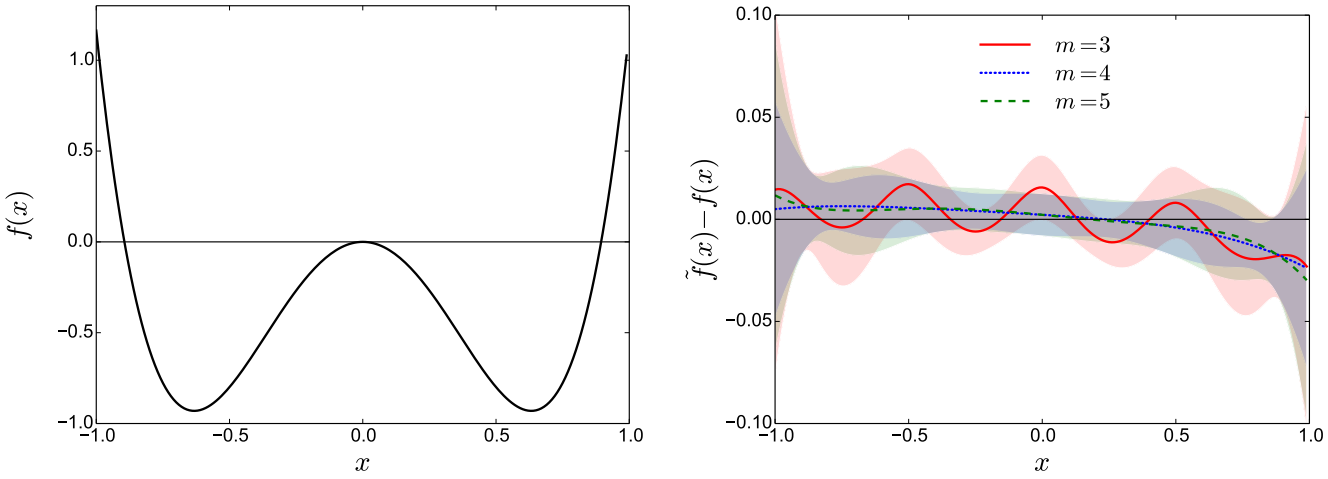

Figure 4: Quartic polynomial test function (left panel). Difference between BHM fit $\tilde{f}(x)$ with different spline orders m and the test function $f(x)$ (right panel).

```
5       24          5.2920      1.5774          24
6       41          3.3034      1.4417          25
7       54          2.0987      1.3849          26
Checking interval 0 (order: 1, number: 0)       27
1       1           0.0005      3.8284          28
2       2           1.7172      3.0000          29
3       4           6.3727      2.4142          30
This interval fit is not good                   31
Checking interval 1 (order: 1, number: 1)       32
1       1           0.0006      3.8284          33
2       2           1.5645      3.0000          34
3       4           7.8119      2.4142          35
This interval fit is not good                   36
Checking separate chi_n^2/n in spline fit       37
level   n       chi_n^2/n       max chi_n^2/n   38
0       1           0.0001      3.8284          39
1       2           0.0002      3.0000          40
2       4           0.0055      2.4142          41
3       8           0.0519      2.0000          42
4       14          0.3837      1.7559          43
5       24          0.8437      1.5774          44
6       41          0.8378      1.4417          45
7       54          0.8693      1.3849          46
Good spline found with threshold T = 2          47
...                                             48
```

First a fit is attempted with one spline piece on the whole domain (lines 4-13). This fit is not acceptable because $\chi^2_n/\tilde{n}$ (third column in the output) exceeds the maximally allowed value $1+T\sqrt{2/\tilde{n}}$ (fourth column in the output) for most of the levels. The second column lists $\tilde{n}$, the number of available bins at each level. This number is in general smaller than 2^n, because some bins do not contain enough data to be used for fitting. Also, hierarchy levels below $n = 7$ were omitted because the fraction of usable bins on these levels was below the set `UsableBinFraction` value.

Since the first fit was unsuccessful, χ^2 is evaluated on each spline interval separately (lines 14-16). In this case, this yields no new information, since only one interval is present. As soon as a level is found where the fit is unacceptable (level 0 in this case), this check stops without proceeding to lower levels, since this is enough to identify a bad interval.

After the interval is divided, another BHM fit is attempted on two intervals (lines 17-26). This fit already has smaller $\chi^2_n/\tilde{n}$ values than the previous one, but still fails the threshold on several levels. Both spline intervals are then again checked separately (lines 27-31 and 32-36, respectively) and both fail the goodness-of-fit check on level 3. Note that level 0 is not present in the individual interval checks, because the bin on this level is larger than each of the spline intervals.

The intervals are numbered consecutively, but additional information is provided so that their location can be recovered (see e.g. lines 27 and 32). The boundaries of an interval always coincide with the boundaries of a bin on a certain hierarchy level (denoted by "order") and "number" denotes

the number of this bin.

After the intervals are again divided, the resulting BHM fit (lines 38-47) is acceptable. No separate interval checks need to be performed and the code exits with the fit result. If `PrintFitInfo` is requested, the goodness-of-fit information of the final result is output again at the end. This includes the $\chi^2_n/\tilde{n}$ values on each level n, the unit standard deviation $\sqrt{2/\tilde{n}}$ of the corresponding χ^2-distribution, as well as the number of standard deviations by which $\chi^2_n/\tilde{n}$ exceeds 1 on each level (last column). If $\chi^2_n/\tilde{n} \leq 1$ the latter value is 0.

5.2. Example 2

This example demonstrates BHM fits for different choices of the threshold T. The sampled distribution is a decaying exponential (`Function=exponential`),

$$f(x) = \alpha \exp(-3x), \tag{9}$$

normalized on the interval $[1, 3]$, which implies $\alpha = 3e^9/(e^6 - 1)$. The function is sampled on the interval $[1, 2.8]$, so that there is a finite number of values N_{exc} sampled outside of the histogram bounds. The total number of sampled points in this example is `SampleSize=100000`.

The histogram data is fitted with `BHM` using the default parameters, with the exception of the parameters defining the fit acceptance threshold, which is set to be fixed at $T = 0$, 2, and 8, respectively. This can be achieved by either setting the value of `ThresholdMax` to be equal or less than the value of `Threshold`, or by setting `ThresholdSteps=0`. The fit results are shown in Fig. 5.

For all threshold values an acceptable fit exists, but with different interval divisions. The extremely low threshold value $T = 0$ (which means that only fits with $\chi^2_n/\tilde{n} \leq 1$ are accepted) yields an overfitted spline with 12 spline pieces. The value $T = 2$ produces a suitable fit with 3 spline pieces that captures the shape of the test function well. The very high value $T = 8$ yields an underfitted spline with only 2 pieces. This spline deviates strongly from the true function and the error on the spline is severely underestimated.

5.3. Example 3

This example demonstrates that `BHM` works for both uniform and non-uniform input histograms. The sampled distribution,

$$f(x) = 0.2G(0, 0.2) + 0.4[G(2, 1) + G(-2, 1)], \tag{10}$$

is a linear combination of three Gaussians $G(\mu, \sigma)$ with mean μ and standard deviation σ (`Function=triple_gaussian`). It has several distinct features and resembles a physically relevant case.

We sample `SampleSize=1000000` data points on the interval $[-5, 5]$ into a uniform and a non-uniform histogram, both with 2^8 bins. Note that the non-uniform histogram binning is predefined and cannot be adjusted by changing the `PowerBins` entry. The non-uniform histogram bins are smaller in the center of the domain (where the sampled function has a sharp feature) and increase exponentially in size towards the domain boundaries. The smallest bin size is equal to the domain length divided by 2^{12}. The non-uniform histogram is always collected in addition to the customizable uniform histogram if `Function=triple_gaussian` is chosen and is output into the file `nonuniform_histogram.dat`.

The fit results are shown in Fig. 6. Both histogram divisions produce fits of similar quality that reproduce the tested distribution well. Since `BHM` automatically considers combinations of elementary bins, there is no need for a case-specific implementation of a non-uniform histogram grid. Note that sampling the same data in a uniform histogram with 2^{12} bins produces nearly the same fit as when using 2^8 uniform bins in this example.

6. Acknowledgments

This work was supported by the Simons Collaboration on the Many Electron Problem and by the National Science Foundation under the grants PHY-1314735 (O.G., N.P., and B.S.) and DMR-1720465 (N.P. and B.S.). O.G. also acknowledges support by the US-Israel Binational Science Foundation (Grants 2014262 and 2016087).

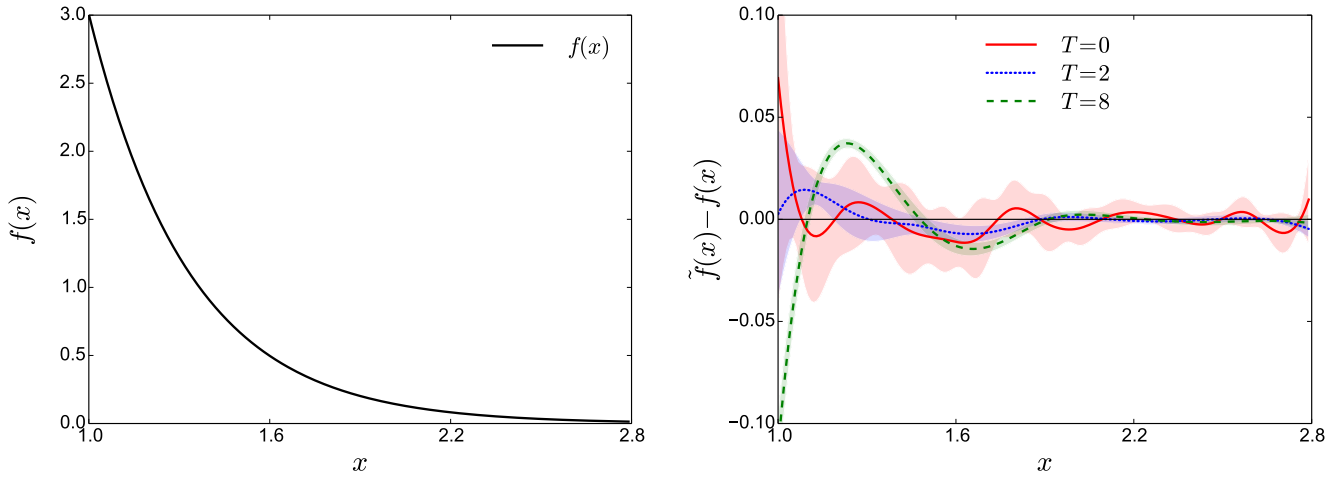

Figure 5: Decaying exponential test function (left panel). BHM fits of the test function with different goodness-of-fit thresholds (right panel).

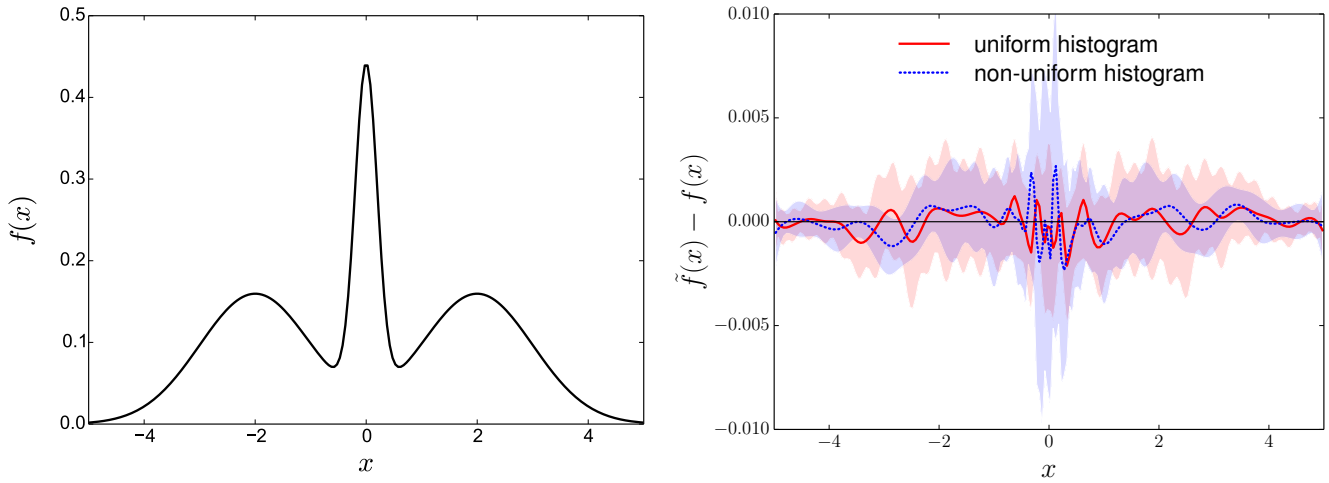

Figure 6: Triple Gaussian test function (left panel). BHM fits of the test function based on a uniform histogram and a histogram with bins of different size (right panel).

References

[1] O. Goulko, N. Prokof'ev, B. Svistunov, Restoring a smooth function from its noisy integrals`arXiv:1707.07625`.

[2] I. Narsky, F. C. Porter, Statistical Analysis Techniques in Particle Physics, John Wiley & Sons, 2013.

[3] D. W. Scott, Multivariate Density Estimation, John Wiley & Sons, 2015.

[4] B. W. Silverman, Density estimation for statistics and data analysis, London: Chapman and Hall, 1986.