MEMLET v1.3 User Guide

Contents

Quick Start

Opening the Program

1. Navigate to the MEMLET folder from Matlab or add the program folder to the Matlab Path (see Initial Setup for more details)
2. Run the program by typing MEMLET into the command window

Load Data

From a File

1. Click **[Select Data File]** in top left corner
2. Navigate to and select file with a list of data (for more information on data loading see Loading Data from a File)

From a Matlab Variable in the workspace

1. Type the variable name of a column vector containing the data in the Variable Name box. The variable will be loaded when another area of the GUI is clicked.

Plot Data

1. Select the plot type (Histogram, Cumulative, or XY Plot) in the "Plot Options" area of the program. See Plotting Data for more details.
2. For Histogram plots, enter the # of Bins for Display you would like to use (for data display only, this choice has no effect on the fitting routine).
3. Click **[Plot Data]**.

Fit Data

1. Choose a PDF from the Select PDF dropdown box (see Selecting a PDF for more details)
2. Optionally check and update the Lower Bounds and Upper Bounds and the Initial Guesses as necessary for your data by clicking the **[Show Fit Options]** checkbox.
   - The order of the bounds and guesses corresponds to the order of the fitting variables listed in the fitting variables box.
   - The simulated annealing fitting algorithm is relatively insensitive to the initial guesses; however it is sometimes useful to provide ballpark guesses if possible.
3. If applicable, enter the known minimum detectable event size in the tmin box in the same units as your data. (See Fitting with tmin and tmax for more details)
4. Click **[Fit Data]**
5. The fitted curve will be plotted over the data and the fitted values will be listed in the Fitted Parameters Output box
6. If desired, choose a new PDF and repeat steps 1-3

More Features

After the data has been fit, you can take advantage of features which allow you to:

- Export figures and fitted values
- Use bootstrapping to determine confidence Intervals
- Determine the significance of fitting to a particular PDF *(i.e. compare a double and single exponential fit)*
- Enter a custom PDF
- Perform global fits on multiple data sets

Initial Setup

Setting up the Program when Matlab is installed

1. Extract the supplied MEMLET Folder to a location of your choice
2. For the program to run, the current Matlab working directory must be the MEMLET folder, or the MEMLET folder must be added to the Matlab Path. It is recommended you add the folder to the Matlab path as described in steps 3 and 4 below.
3. Add the program files to the path by either:
   a. Changing the current Matlab working directory to the MEMLET folder then typing "`addpath(pwd)`" in the Matlab Command Window.

      OR

   b. In the command window, typing "`addpath DIRNAME`" where DIRNAME is replaced by the full path to the program directory (for example "`addpath C:/Matlab Code/MEMLET/`")

4. Type "`savepath`" into the Matlab Command window to ensure the MEMLET folder will be in the Matlab path each time Matlab opens.
5. Type MEMLET in the command window, or run MEMLET.m to run the program.

   Note: The fitting program depends on multiple m files in the program directory, so it is highly recommended to leave all of the files in the program directory. However, the directory can be renamed.

Setting up MEMLET when Matlab is *not* installed (Standalone)

1. Install the program by double clicking the Install MEMLET icon.
2. Follow the installation instructions. If Matlab Runtime is not already installed, the program will prompt you for permission to automatically download and install it. The Runtime from MathWorks is large and may take a while to download.
3. Open the Program by navigating the installation path chosen during setup, and double clicking on MEMLET.exe inside the "application" folder.

Note: Loading data using the Enter Variable Name function does not operate in a standalone installation. Also, the fitted values must be manually copied out of the program and are not saved in the workspace.

Loading Data

Data can be loaded either from a file or from a current Matlab Variable (if Matlab is installed).

Loading Data from a File

You can load data from a text file that is stored with a .csv or .txt extension. To do so simply:

1. Click **[Select Data File]** in the top left of the interface
2. Navigate to the file and select it
3. The data is now loaded and ready to be plotted

File Formatting

The chosen file is read using the built-in Matlab function dlmread.m which reads in delimited text files. These files can be delimited with either commas (.csv files), tabs, or spaces. For more information about how files are read, see the Matlab documentation of dlmread. Data should be arranged in columns within the file.

Multiple Columns in a File

Multiple datasets or a dataset with multiple variables can be read in from a single file by placing each dataset in its own column. However, if the columns are of different lengths, dlmread will fill the short columns with zeros, which may be undesirable during the fitting process. In such cases we recommend creating a cell variable in Matlab containing all the data and loading it directly from the Matlab workspace. Another alternative is to fill the short columns with large negative numbers and use the tmin feature of the program to remove these values from the plot and fitting (i.e. fill short columns with -10000, and set tmin=-9999.)

Loading Data from a Workspace Variable (Matlab Installation Only)

In the case where data is already stored in a variable in the Matlab workspace, type the name of the variable into the Enter Variable Name box, and the data will be automatically loaded when another part of the GUI is clicked. The data should optimally be given as either a single column vector or in multiple columns. If multiple columns are present in the workspace variable, they will be interpreted as separate data sets. See next section.

Tip:

The program loads the data variable by evaluating whatever Matlab expression is inputted into the Enter Variable Name box. This allows some simple operations to be performed on the data during the loading procedure, for example:

1. If your data variable, named *durations*, contains durations measured in milliseconds, but you'd like to visualize it in the units of seconds, you can type "`durations*1000`" into the Enter Variable Name box.
2. If your data variable, *FRETVals*, consists of two columns, the first of which is a time index not relevant to your fitting, and the second of which are your values of interest, you can type "`FRETVals(:,2)`" into the Enter Variable Name box to only load the second column of data.

Multiple Columns in a Data Variable

Multiple columns can be read into the program at one time from a variable with multiple columns. This is necessary when either performing fits on multidimensional data (more than 1 data variable) and/or when performing global fits on multiple data sets. In cases where the datasets are of different sizes, the data can be stored in a cell array and loaded into the program. For detailed information about creating cell arrays, see the Matlab documentation, but consider the following example:

A global fit is desired for two data sets, stored in *dataA* and *dataB*. Data set A has 100 values, while data set B has 150. Create a cell array called *allData* to load into the program by typing in the Matlab command window:

```
allData=cell(2,1)  % creates a cell array with space for two vectors
allData(1)=dataA;  % puts dataA in the first cell
allData(2)=dataB;  % puts dataB in the second cell
```

Now type `allData` into the Enter Variable Name box and click somewhere on the GUI background. A global fit can then be performed on the two sets of data. See the sections on Plotting Data, Global Fits, and Multidimensional Data sets for working with multiple columns of data.

Plotting Data

There are several ways to display the data and the fits which are produced by the program. Regardless of how the data is displayed, the fitting algorithm always operates on the unbinned, raw data.

Plotting Data

After data is loaded into the program, choose the type of plot you would like to use to visualize the data. Then click the **[Plot Data]** button. A new style can be selected at any time and the **[Clear Plot]** button followed by the **[Plot Data]** button can be used to re-plot the data. Any fits that have already been performed on the data can then be plotted again (without needing to fit the data again) by selecting the desired PDF from the list on the left and clicking the **[Replot Fit]** button.

Plotting One Column of a Multidimensional Dataset

If a data variable or file is loaded into the program which contains multiple columns of data (for example from several experiments), it is possible to choose which column of data you would like to plot. For histograms or cumulative frequency distributions, this is accomplished using the X Col selector which appears if the data has more than one column. Select the column number of the data to plot, then press the **[Plot Data]** Button.

Effect of Specified Minimum and Maximum Detectable Event (t_{min} and t_{max}) on Plots

When a non-zero t_{min} or t_{max} is specified before data is plotted, only data points which are greater than or equal to the specified t_{min} and less than or equal to t_{max} will contribute to the plot. If the value of the t_{min} or t_{max} is changed by the user, it will be necessary to press the **[Clear Plot]** button, then **[Plot Data]** button in order to update the plot to only include points between t_{min} and t_{max}.

Plot Types

Histogram

This option creates a normalized histogram giving the probability density on the y-axis, with the number of bins specified in the # of bins for display box. The histogram will be normalized based on the data so that the sum of the histogram bar areas is equal to one. To determine the non-normalized bin heights, multiply the normalized bin height times the bin width times the total number of data points.

Fits are visualized by plotting the normalized PDF over the histogram.

Cumulative Distribution

The cumulative distribution option uses Matlab's empirical cumulative distribution function (ecdf.m) to create a distribution showing the proportion of events (y-axis) which are less than or equal to the value plotted along the x-axis. The final amplitude of this distribution will always be equal to one, since all events (100%) will have occurred at or before the final value of the x-axis.

Fits are visualized by producing a cumulative distribution through numeric cumulative integration of the fitted probability density function.

XY Plots

XY plots are used for datasets which have multiple columns, for instance a dependent and independent variable. The X Data and Y Data column selectors are enabled when an XY Plot is chosen and the loaded data has multiple columns. A scatterplot of the points in the specified columns will be generated.

Because of the variety and complexity of multidimensional PDFs, plotting the results of fits to multidimensional datasets is complex. See the corresponding Multidimensional Data section for more information.

Fitting Data

Selecting a PDF

Select a PDF from the Select PDF drop down list on the left side of the screen. To access additional PDFs, including the Camera PDFs, use the "Show PDFs" menu at the top of the program next to "File". To enter a custom PDF, see the Custom PDF section. Custom PDFs can also be added to the drop down list by editing the PDFList.m file.

Setting Bounds and Guesses

Upper and lower bounds and an initial guess must be specified for each fitted parameter. Default values are loaded from the PDFList.m file for the built in PDFs. To edit these bounds and guesses, make sure the **Show Fit Options** box is checked. The order of the bounds and guesses corresponds to the same order of the fitted parameters listed in the Fitted Variables box. The bounds and guesses for each variable should be separated from each other by a comma. The default values loaded for each PDF can be changed by editing the PDFList.m file.

The program is relatively insensitive to the initial guesses provided, but they still must be given. Some PDFs are more sensitive to initial guesses, so if the fitted values or curve returned by the program does not seem to fit the data well, try different initial guesses. In practice, a guess that is of the same order of magnitude of the true value will typically yield a proper fit. It is important that the PDF function returns a valid numerical result using the initial guesses (not infinity or NaN) or the fitting program will fail.

Fitting Data

Once all the parameters have been set, click **[Fit Data]**. The words "Fitting Data" should appear over the plot area. Once the fitting is complete, "Fitting Data" will disappear and the fit should be plotted over the data.

Editing Fitting Routine Parameters

The parameters which control the fitting algorithm can be accessed in the MLEAnneal.m file. Here the options for both the annealing process and the linear minimization algorithm, which finds the final minimum, can be edited. (A related file MLEAnnealBoot.m exists and is used for fitting during the bootstrapping, by default the annealing temperature is reduced to improve speed, possibly because the initial guesses for the bootstrapping are the fitted values.) More details about these parameters can be found in the Matlab documentation on simulannealbnd.m).

Fitted Values

The variable names and values returned by the fitting algorithm are displayed in the "Fitted Parameters Output" box. The log-likelihood value is also given, which can be used for reference or comparing two models. These values are also stored in the Matlab workspace, as described in Exporting Fits and Figures.

Note about log-likelihood Values: Because the likelihood is obtained from the PDF, which can have values greater than one when properly normalized, the log-likelihood may be either positive or negative. In all cases, a larger (or less negative) log-likelihood indicates a better fit. To compare two models with different numbers of free parameters, however, the log-likelihood ratio test should be used, see Model Testing.

Multiple Fits

After fitting the data, another PDF can be selected from the drop down list and can be fit to the same data by pressing the **[Fit Data]** button. The fitted curve will be added to the plot, and the output box on the far right will be updated with the new fitted parameters. The values from the previous fit may be recalled by selecting the corresponding PDF from the drop down list.

It is possible to plot a subset of the fits performed by clearing all the fits by pressing the **[Clear Plot]** button, selecting each desired PDF, and then pressing the **[Replot Fit]** button for each.

Accounting for Minimum or Maximum Detectable Event Size (t_{min} and t_{max})

The program allows for easily taking into account that in many experiments there is a limitation on the value of the minimum detectable event. For example, when acquiring data at rate of 10 per second, "events" that are shorter than 100 ms will likely be missed. Similarly, there may be a constraint that prevents events longer than a particular value from being observed. In general we refer to these as t_{min} and t_{max}, but the implementation is applicable to non-temporal data.

Implementation in the Program

Currently, a subset of the program's built-in PDF's will automatically renormalize when an instrument dead-time, t_{min}, or t_{max} is provided. These include the ones based on exponential or Gaussian distributions:

- Single Exponential
- Double Exponential
- Triple Exponential
- Bell's Equation
- Gaussian
- Double Gaussian

If there is no t_{min} or t_{max} that applies to your data, leave the tmin or tmax boxes blank. It possible to apply either the correction for t_{min} or t_{max} individually or both corrections at the same time.

In order to use this renormalization feature, simply enter the t_{min} or t_{max} value (in the same units as the data) in the tmin or tmax box near the **[Replot Fit]** button. This will do two things:

1. The PDF will be updated to one which is renormalized based on the entered values
2. All values in the data which are less than the t_{min} (if provided) will not be plotted or used for fitting. Similarly any values larger than the t_{max} (if provided) will not be plotted or used for fitting (values equal to the t_{min} or t_{max} value will be included)

If the tmin or tmax value is changed, prior fitted values will be cleared, and the fit will need to be performed again. The program does not store multiple fits from one PDF with different dead times.

For information about creating custom PDF's which are renormalized based on a given MDE, please see the Custom PDF section.

Theory

Probability density functions (PDFs) are typically normalized over their entire range (e.g. 0 to infinity for exponential functions, negative infinity to positive infinity for Gaussian distributions, etc.). However, when an instrument dead-time (t_{min}) is present, it is not possible (probability = 0) for an event shorter than this time to be present in the data. Thus, to obtain a total probability equal to one, the PDF, g(t), should be renormalized, so that the integral of the PDF from the dead-time through infinity is equal to one:

$$f(\mathrm{t}, t_{min}, \alpha_1, \ldots, \alpha_m) = \frac{g(\mathrm{t}, \alpha_1, \ldots, \alpha_m)}{R(t_{min}, \alpha_1, \ldots, \alpha_m)}$$

where R is a renormalization factor given by

$$R(t_{min}, \alpha_1, \ldots, \alpha_m) = \int_{t_{min}}^{\infty} g(\mathrm{t}, \alpha_1, \ldots, \alpha_m)\, \mathrm{dt}$$

While this renormalization is important for nearly all PDFs, when multiple components are present in the data, such as in a process described by two independent, exponential processes, this correction will enable the relative amplitudes of each phase to be reported correctly. For examples of this, see the publication describing this program, Woody et al. Biophysical J., 2016.

A t_{max} value can be taken into account in a similar way, by replacing negative infinity in the upper limit of integral above with t_{max}.

In addition, this principle allows more complex treatments of instrument dead-times, as in the case of using a camera with a finite frame rate to detect events indicated by fluorescence. In such a case, events which are shorter than a specified cutoff time can actually be detected depending on their exact timing. This is described in detail by, Lewis et al., Biophysical. J. 2016, where a specialized PDF that can be used with MLEFIT to fit lifetimes, including short-lived states approaching the frame rate of acquisition.

Exporting Fits and Figures

Exporting Fitted Values

After a fit has been performed, the fitted values and the log-likelihood value will be displayed in the "Fitted Parameters Output" box in the GUI, but will also be saved in the current Matlab workspace (if using the non-standalone version). The name of the variables containing these values are indexed based on the position of the selected PDF in the drop down list. For example, the values produced by fitting to the "Single Exp" function will be stored in the variable `fittedVals1`, and the log-likelihood will be in `loglik1` because "Single Exp" is the first PDF listed. Fits to "Triple Exp", the third PDF will be stored in `fittedVals3`, and `loglik3`, and so on. When a new fit is performed using the same PDF (even if the dead-time, guesses, data, etc have changed), the old values will be overwritten with the latest fits.

In the standalone version (and Matlab installed version), the fitted values can be saved to a text file, along with the PDF, and data file name, by clicking the [**Save Fit Values**] button. This will save all fitted values for the current data, including models that have been fit to the data, but are currently not being displayed. The user will be prompted for a file location and name.

If bootstrapping is performed, the fitted values from each round of bootstrapping will be stored in a similarly indexed variable beginning with BS_out_ (e.g. BS_out_2 or BS_out_3), and the corresponding confidence intervals will be stored in a variable reflecting both the PDF index and the confidence level (e.g. For a 90% confidence intervals for a fit to a Triple Exp PDF, `BS_3_ConfInt_90`)

Exporting Figures

Once the GUI plot contains the data and fits that you desire (see Plotting Data), you may click the Pop Out Figure button, which will create a new Matlab figure containing all of the same information and axes as the plot in the GUI. Matlab's standard methods of editing or saving the figure can then be employed. If using the Standalone version of the program, it is recommended to pop out the figure, go to File, Save As, and then save the figure as a vector format file (e.g .eps) which can then be fully edited in software such as Adobe Illustrator.

Weighted Fits

Starting in Version 1.3, MEMLET includes functions for performing weighted fits where each data point does not contribute equally to the parameter estimates. This can be useful when you are fitting data from multiple experiments under the same conditions, but the experiments have produced different numbers of data points. If you would like the fitted parameters to equally weight data from each experiment (instead of weighting each data point equally as is done by default), you can use the weighted fit option. This option can also be used for other, more complex ways of weighting data.

There are weighted versions of each of the 'built-in' PDFs included with MEMLET. Users are also able to create custom weighted PDFs (See Weighting Theory and Implementation below, along with Custom PDFs, and Multidimensional Fits sections).

Performing a weighted Fit

- Load your weighted data as two columns with the data in the first column and the weights in the second column (see Loading Multiple Columns in a Data Variable). For help in creating the weight column, see below.
- Select a weighted PDF by first clicking the "Show PDFs" menu button, and then "show Weighted PDFs". Select the PDF you would like to use that is designated by "Weight" at the end
- Click **[Fit Data]**

Note on Plotting Weighted Fit results:

Depending on the data and weighting, the plotted fit may not look like a good fit to the data, since the data is displayed without weighting, but the fit accounts for the weights provided. We leave it up to the user to decide how to best present the data and the fit. Only histogram and cumulative distributions plots are currently supported for weighted fits.

Choosing the weights

Each data point must have a corresponding weight. If your data consists of a vector of 1000 points, you need to create a weights vector that is 1000 points as well. The relative value of each weight gives its contribution to the fit (points with larger corresponding weights affect the fitted parameters more). It is highly recommended that the sum of the weights adds to the total number of data points, as described in the example below.

Important Note for Log-likelihood Ratio Testing

It is absolutely required for log-likelihood ratio testing that sum of all the weights for a given fit add to the total number of points in the fit. This ensures that the optimized log-likelihood value is comparable to that for an unweighted fit. The value of log-likelihood determines the significance value for model comparison and will give incorrect results if the weights are not properly normalized in this way.

Weights Example

Three experiments were performed under the same conditions which resulted in 500, 250, and 250 data points each. The researcher wants to equally weight the results from each experiment, since there is no reason why the values from the first experiments are better predictor of the parameters of interest than those from the second and third experiments. The researcher can combine all three datasets into on column vector by using a command such as:

```
dataAll=[data1; data2 ;data3]
```

Weights can then be generated such that data2 and data3 have weights twice that of data1 and so that the sum of all weights is equal to 1000 (the total number of data points). For our simple example, we can consider how to set the weights manually, then we will look at a general method.

We know for our data that data2 and data3 points should each count twice as much as data1 points, since there are twice as many data1 points as there are either data2 or data3. Thus, we could choose weights such that weight1=1, and weight2=weight3=2. We then need to create a column vector with these weights so that each point in our dataset has a value for weight:

```
weights1=1*ones(500,1); weights2=2*ones(250,1); weights3=2*ones(250,1);
```

We can then combine this into one vector

```
weightsAll=[weights1;weights2;weight3]
```

However, now we need to make sure the sum of all the weights is equal to 1000, our total number of points. Currently is equal to 1500. We can divide the entire vector by 1.5 to ensure the total sum if 1000

```
weightsAllNormalized=weightsAll/1.5;
```

Now we can put the data into MEMLET by typing the following in the *Enter Variable Name* box:

```
[dataAll weightsAllNormalized];
```

Generalized Example

The most general way to do this is to use code similar to that shown below. First, determine the size of each data set

```
nDataSets=3; lenAll= length(dataAll); len1= length(data1); len2= length(data2); len2= length(data2);
```

Next, create a weight vector for each dataset so that it equally contributes to the fit. If we want each data set to contribute equally, the total fractional weight of all points in each data set is 1/nDataSets, so the fractional weight for each point in data1 is (1/(nDataSets))/len1 or 1/(nDataSets*len1). To normalize the weights so that the sum of all weights is lenAll, we just multiply by lenAll:

```
weight1 = lenAll/(nDataSets*len1); weight2 = lenAll/(nDataSets*len2); weight3 = lenAll/(nDataSets*len3);
```

we then create the arrays with a weight for each data point.

```
weights1 = weight1*ones(len1,1); weights2 = weight2*ones(len2,1); weights3 = weight3*ones(len3,1);
```

and then combined the weights together as before:

```
weightsAll = [weights1; weights2; weight3]
```

Weighting Theory and Implementation

To achieve proper weighting, all that is required is raising the likelihood value for each point (result of the PDF) to the weight value for that point as an exponent. This exponentiation is necessary because the log of likelihood is minimized, and thus the weights become multiplicative factors for determining the likelihood. Thus doubling the weight value in this scenario leads to it contributing twice as much to the final **log**-likelihood.

Weighted log-likelihood= weight*log(likelihood)=log(likelihood^weight)

Any PDF can be made into a weighted PDF by just adding " .^w " to the end of the equation and then adding w as a data variable. The weights will be treated as if they were a data variable. (see Multidimensional Fits)

Weighting Global Fits

Global Fits can also be weighted. See the section under Global Fits

Bootstrapping

Bootstrapping can be used to estimate the confidence intervals for the fitted values by repeatedly fitting "bootstrapped" data sets produced by randomly selecting (with replacement) data points from the original data set until the same number of data points as the original data are selected. The bootstrapped data will thus contain some of the original data points multiple times, and will be missing other data points. Fitting many such data sets (e.g. 100-1000 sets), and by examining the fitted values returned, confidence intervals can be estimated.

Note: When first performing bootstrapping, you may receive a request to allow MEMLET or Matlab to pass through your computer's firewall. You may deny this request with no loss of functionality. It is caused by the parallel computing code use for bootstrapping, but no network access is actually required.

Running the Bootstrapping

To perform bootstrapping, first load your data and fit it to your desired PDF. After the fit is performed, and the appropriate PDF is selected in the drop down menu, enter the number of rounds of bootstrapping in the # of Iterations box toward the bottom right of the GUI panel. The number of rounds necessary depends on the size of the confidence interval you desire (in practice, 100 rounds should be considered a minimum in most cases, with 500-1000 being more common). Then click, **[Run Bootstrapping].** The word "Fitting …" should appear across the plot. Because the bootstrapping process utilizes the parallel computing capabilities of Matlab, the Current Iteration box will only update about every 10 iterations depending on the number of cores being utilized (see Parallel Computing Tips below). Once all rounds have finished, the confidence intervals will be displayed in the "Confidence Intervals" box at the lower right. The results may also be visualized in several other ways (see below).

Calculation of Confidence Intervals

The program calculates confidence intervals by taking the confidence level (α, i.e. α=0.90 for 90% confidence intervals), and returning the values corresponding to the (1-α)/2 and 1-(1-a)/2 percentile levels (e.g. the 95% percentile values and 5% percentile values). For examples, for a fit to a single exponential, the fitted value maybe be k1=12.5 per second, with a 90% confidence interval of 11.2- 16.2 per second. This indicates that 90% of the fitted values from the bootstrapping were between 11.2 and 16.2 per second.

The confidence intervals are computed for each fitted variable independently, and thus if two of the fitted variables show high degrees of correlation with each other, it is possible that the confidence intervals returned may not accurately describe the joint confidence region of the data (for more information and examples, see texts such as Numerical Recipes in C, section 15.6, Confidence Limits on Estimated Model Parameters). See below for suggestions on how to visualize this type of correlation and confidence regions.

Visualizing the Bootstrapped Results

Bootstrapped Bounds on the Displayed Fit

When visualizing the fit using either a histogram or cumulative frequency distribution, the confidence area of the fitted curve can be visualized on the plot as well. This is done after bootstrapping has been performed by entering the desired confidence value (e.g. 0.9 for 90% confidence intervals) in the Confidence Int box, then clicking the **[Plot Bootstrap Bounds]** button. Depending on the PDF, this process may take a while, because the PDF will be evaluated at each set of fitted values from the bootstrapping process. At every point along the x-axis, the (1-α)/2 and 1-(1-a)/2 percentiles of the possible values of the PDF from the bootstrapped fits will be displayed and the minimum and maximum values from these evaluations will be plotted. This gives a good visualization of the range of fits returned from the bootstrapping.

Visualizing the Distribution of Fitted Variables

Another way to visualize the results of the bootstrapping is to plot a distribution of the fitted values which are returned. This can be accomplished by clicking the **[Plot Variable Distr]** button after running the bootstrapping algorithm. This creates a new figure for each fitted variable, and displays a histogram showing all of the fitted values. Usually, the distribution is approximately Gaussian, but severe deviations from Gaussian distributions may be informative. In particular, if there is more

than one local minimum where the fitted value might fall, this will be displayed as multiple peaks in the bootstrapping distribution(s) and may lead to over-estimation of the confidence intervals. See Bootstrapping Details below for information about apparent bi-stability in multi-component distributions.

Correlation between Multiple Fitted Variables

To determine the amount of correlation between two fitted variables, the bootstrapped values may be plotted as a scatter plot, using commands similar to:

```
figure; plot(BS_out_2(:,2), BS_out_2(:,3), '.');
```

which will present a scatter plot showing how the values of the two rates, k1 (x-axis), and k2 (y-axis) correlate after bootstrapping has been run on a double exponential fit. Confidence regions can be constructed which contain, e.g., 90% of the points on the 2D graph. However, the shape of such regions is not always straightforward. Standard methods of determining correlation (least squares fitting to a line, Matlab's corr.m function, etc.) may be used to quantify the amount of correlation between the bootstrapped fits. Further methods and discussion for detailed correlation analysis are given in Johnson et al. (FitSpace Explorer: an algorithm to evaluate multidimensional parameter space in fitting kinetic data Anal. Biochem., 387 (2009), pp. 30–41).

Parallel Computing Tips

The multiple fitting requirement of bootstrapping takes minutes of time to run. For newer versions of Matlab, the bootstrapping routine should automatically take advantage of the local parallel computing capabilities of Matlab and the computing hardware (e.g. multiple cores). However, it may be useful to modify the default profile in Matlab's Cluster Profile Manager to change the number of Matlab "workers" that can run locally. In general, performance will increase as the number of workers increases up to the number of cores in your machine (assuming there is enough RAM available). To manually start a local pool of machines with n number of workers, enter parpool(n) (ex. parpool (8), for eight workers) into the Matlab command window before starting bootstrapping. In addition, it is possible to use a remote cluster to perform these computations, see the Matlab documentation for more details.

The number of iterations that must occur before the Current Iteration indicator is updated is set in the MEMLET.m file. This parameter, called `loopsize`, is set to be twice the number of parallel workers in the parallel pool. In general, the larger the `loopsize`, the more efficiently the parallel computation will occur. Thus, it may increase the efficiency of the process to increase the value of `loopsize` at the expense of frequent updates to the Current Iteration indicator (implying patience and trust). The simplest way to find the `loopsize` variable is to search MEMLET.m for `loopsize`. The most efficient values of `loopsize` are integer multiples of the number of computer cores available in the local machine.

Bootstrapping Caveats and Details

After each bootstrapped data set is created, it is fitted in a similar way the entire data set was originally fitted with the following differences:

1. The original fitted parameters from the entire dataset are used as the guesses for the bootstrap fits
2. The annealing temperature parameter is reduced to 0.5 in order to prevent undesirable instability ("jumping") of fitted values in multicomponent fits

The jumping mentioned above can be understood by an example of a data set which is described by a combination of two exponential distributions with rates of 100 s^{-1} and 20 s^{-1}, where A gives the relative amplitude of the process occurring at 100 s^{-1}. The program does not currently include a way to ensure that in the Double Exponential PDF, k1 will always be larger than k2 or vice versa. Therefore, when fitting the data, sometimes k1 may be found as 100 $s^{-1}$and k2=20 $s^{-1}$and other times k1=20 $s^{-1}$and k2=100 s^{-1}. Both are correct and well constrained fits. However, during bootstrapping, if the annealing temperature is too high (and/or the two component rates are similar), the bootstrapped values of k1 will sometimes correspond to the faster rate and sometimes to the slower rate. The program does not currently implement any constraints to prevent this because it is very model- and parameter-dependent. Instead the annealing temperature has been lowered for the bootstrapping process and the raw bootstrapped values have been made available to the user. This type of parameter jumping should be apparent from unreasonably large confidence intervals and dual- or multiple-peaked bootstrapping parameter histograms.

It is possible to modify the annealing temperature or other fitting parameters for the bootstrapping process in the MLEAnnealBoot.m file. This may be useful in certain circumstances, for instance the TimeLimit parameter or tolerance

parameters maybe be changed to speed up the bootstrapping process or make it more reliable if deemed useful in a particular case.

Model Comparison

The program allows the users to easily test the significance of models with more adjustable fitting parameters compared to a simpler model (ex. test the significance of a double exponential vs. single exponential). This is done by using the "Log-likelihood Model Testing" box in the lower left corner of the GUI.

Note: The simpler model being compared must be a constrained version of the complex model. This means that a double exponential can be compared to a single exponential, but a single Gaussian (2 free parameters) cannot be compared to a single exponential (1 free parameter) with this method. See below for Examples of Constrained Variables.

Performing the Test

1. Fit the data using the model with the larger number of parameters (ex. fit to the double exponential PDF if comparing double and single exponentials).
2. In the Variables to Constrain box, enter the variables which are constrained in the simplified model. These should be entered as pairs of the variable's name and the constrained value with each pair separated by a comma (ex. A=0, k1=1, to constrain the variables A and k1 to values of 0 and 1 respectively)
   Important Note:
   When entering constrained variables, it is important to keep two things in mind:
   A. The number of variables entered into the box must be equal to the total number of variables which do not contribute to the simplified model. For example, in a double exponential PDF function $Ak_1\exp(-k_1t) + (1-A)k_2\exp(-k_2t)$, setting A=0 will effectively reduce the number of free parameters to 1 (since k1 will no longer contribute since the only term containing k1 is multiplied by zero). If only A=0 is entered into the Variables to Constrain box, the program will still try to fit k1 (resulting in random values) and will not calculate the correct p-value since it will assume that the models only differ by one free parameter. Thus it is necessary to enter both A=0 and k1=0 into the Variables to Constrain box.
   B. The constrained values must be set to values which will result in a valid output from the PDF. For instance, you cannot constrain the `sig` (standard deviation) term in a Gaussian function to be equal to 0, which would cause the program to return an error (since in the Gaussian PDF the `sig` variable is in a denominator).

3. Click **[Test Constrained Model]**, the word "Fitting…" will appear on the plot, and then the constrained fit will be plotted and the values reported in the Alternative Fitted Values box.
4. The log-likelihood value of the constrained model will be displayed along with a p-value indicating the confidence level at which the less complex model can be rejected (taking into account the number of additional free parameters). For example: a p-value of 0.04 indicates a 96% confidence level that the model with fewer free parameters is a significantly less good fit to the data. A p-value of 1 would offer no justification at all for using the model with more free parameters

Examples of Constrained Variables

Unconstrained PDF	Constrained PDF	Constrained Variables	Change in Degrees of Freedom
Double Exp PDF	Single Exponential PDF	A=1, (k2=0)	2 (A is constrained and k2 no longer contributes)
Double Gaussian	Single Gaussian	A=1, (mu2=0, sig2=1)	3 (A is constrained and mu2 and sig2 no longer contribute)
Bell's Equation	Single Exponential (Force independent)	d=0	1
Gamma Distribution	Exponential Distribution	k (shape)=1	1
Gamma Distribution	2 sequential Exponential processes	k (shape)=2	1

Model Testing Theory

The theory behind model testing using the Log-likelihood ratio test has been long established (see Wilks 1938 and van der Vaart 1998 for more details and proofs). In summary, a model can be compared to a constrained version of the same model by examining the ratio of the likelihoods. In such cases, twice the log of the ratio of the likelihoods is expected to be approximately chi-squared distributed with the number of degrees of freedom equal to the difference in the number free parameters in each model. This approximation has error terms on the order of $1/\sqrt{n}$, where n is the number of data points being fit. In practice, because the program finds the log-likelihoods and not the likelihoods directly, the quantity $2(LL_{complex}-LL_{simplified})$ is approximately chi-squared distributed. Thus a p-value may be determined from the cumulative density function of the chi-squared distribution with degrees of freedom given by the change in number of free parameters.

Global Fits

If it is hypothesized that values from multiple datasets are described by the same probability density functions which share some, but not all, of the same values for the fitted parameters, the program is able to perform a global fit. (Ex: The on-rate of a substrate when experiments are performed at several substrate concentrations)

Preparing the Data

Currently, performing global fits is only supported when all the datasets are combined into either one Matlab variable or one file. Each dataset should be in its own column. If the datasets contain different numbers of points, the datasets must be combined into a cell or padded. See Multiple Columns of Data for more information about how to create these cells for use in the program.

Performing the Global Fit

To perform the global fit, check the Global Fit checkbox. This will cause a new entry box to appear into which you may type the names of the variables which are unique to each dataset. For example, in two datasets described by a Gaussian distribution, you may know that the peak positions are the same in each dataset, but the widths of the distributions could differ. In this case, you would type `sig` into the "Unique Global Variable" box. When you click Fit data, the program will fit all datasets, automatically creating new fitted variables for each data set for the unique global variables specified. For example, sig_1 and sig_2 will represent the standard deviation of the first and second datasets respectively.

Working with the Results of a Global Fit

Once the global fit has been performed, the fitted variables appear in the "Fitted Parameters Output" box. The variables also are stored in the workspace (see Exporting Fitted Values) in the same order as presented in the "Fitted Parameters Output" box.

The fits may be visualized for each dataset by selecting the desired dataset using the Select Column drop down menu. Next, click **[Clear Plot], [Plot Data**], then **[Replot Fit]** to plot this particular data set and fitted curve. The program does not allow multiple datasets or individual and global fits to be plotted simultaneously.

[Added in V 1.2]

Popping out the Figure for Global Fits

To pop out the figure for each dataset when doing global fitting, first load and fit the data as described above. Then click the [**Pop Out Figure]** button. You can then click **Yes** to create a new plot for each dataset with the currently selected fit. The datasets are plotted in the order of the datafile/structure and does not necessarily start with the currently displayed dataset, i.e. the lowest numbered figure will show the first dataset (first column).

If you would like to plot multiple fits for each dataset, you should clear the plot, select the dataset using the column selector, plot data, then select the first fit you would like to plot, click **[Replot Fit]**, select the next fit you would like to plot, click **[Replot Fit]** and repeat until all the desired fits are plotted. Then click [**Pop Out Figure]** and answer **No** to only pop out the current graph. Using the replot fit functionality prevents the user from needing to running the fitting procedure after switching datasets, since this can often take a long time for large, global fits.

Weighted Global Fits

To perform weighted global fits, a column of weights should be generated for each dataset as described in the Weighted Fits section. The weights columns are placed immediate after each data set like they are another data variable for that dataset as described in the Global Fits with Multidimensional Data.

In order to perform log-likelihood ratio testing, it is important or keep in mind that the sum of all the weights across all datasets should be equal to the number of points in all the datasets. Implementing and interpreting these weighted global fits should be done careful, and we encourage the user to seek proper advice if they have any questions.

Multidimensional Data

The use of PDFs with more than one data variable within a given dataset is available. An example of this type of dataset is an optical trap experiment in which the force (an independent variable) is set stochastically by Brownian fluctuations and the duration of events (the dependent variable) depends on force.

Preparing the Data

For a multidimensional data set, the data should be arranged in columns with each column corresponding to one of the data variables. For more information about this, see Loading Data with Multiple Columns.

Important: If the data contains a minimum or maximum detectable event length (t_{min} or t_{max}), then the variable subject to these constraints should be the first column of the dataset.

Entering a Multidimensional PDF

A multidimensional PDF will have more than one data variable in it. For instance, the provided Bell's Equation PDF contains the variables t (duration) and F (force), both of which vary within the set of events. A custom PDF may be written which has two or more data variables. The data variables should be listed in the Data Variables box separated by commas. The order of the listed data variables should correspond to the order of the columns in the data (i.e. for Bell Equation PDF, event durations should be placed in column 1 and the forces in column 2, since the variables are listed as "t,F" in the Data Variables box.) As mentioned above, if using the t_{min} or t_{max} feature of the program, the first variable should be the one subjected to these constraints.

The handling of t_{min} and t_{max} in Multidimensional Fits

When using the t_{min} feature of the program with a multidimensional data set, the program will remove all data points which correspond to observations in which the value in the **first column** is less than the specified t_{min}. For instance, in the data set given below of durations and forces, if a t_{min} of 20 ms is imposed on the data, the entire row which corresponds to a data point with a duration below 20 ms will be removed (shaded rows) for plotting and fitting purposes. Data subject to a t_{max} are handled in an analogous way.

t (duration in second)	F (forces in pN)
0.107	1.42
0.013	0.28
0.25	0.19
0.015	0.85
0.25	1.21

Plotting the Results of a Multidimensional Fit

Because of the nature of multidimensional fits, there are many ways in which to plot the results. For two dimensional data sets, an x-y plot can be used to visualize the results. In this case, the expectation value for the y-axis variable is calculated at a range of points along the x-axis to yield a line fitting the data. This may or may not provide a useful display, so users should explore possible methods, such as contour plots or two-dimensional groupings of data points, to plot the results.

Global Fits with Multidimensional Data Sets

Global fits can be performed on a multidimensional data set in two ways. The first is to place each data set into separate groups of columns, and then to perform an explicit global fit in the program (checking the Global Fit? box). This method should be used when the data collected in each experiment is two dimensional (i.e. force and duration are collected in each experiment, and the experiment is repeated under different conditions.) A second way, described does not explicitly use the global fit feature, but may be easier to perform if only one data variable is collected during a given experiment.

Explicit Global Fit for Truly Multidimensional Data

In cases where multiple experiments are performed at various conditions, but where multiple variable data values are collected in each run, the data should be arranged as follows, assuming that the data variables for each set are x and y, and x_1 and y_1 are the data from the first data set, x_2 and y_2 are the data from the second data set and so on:

x_1	y_1	x_2	y_2	x_3	y_3

If the datasets contain different numbers of data points, then cells must be used for each column if loading data from a Matlab variable;(see Loading Data with Multiple Columns). Cells should not contain more than one column of data (ex. create a cell like this: `cell = {x1,y1,x2,y2,…}` not `cell = {[x1 y1],[x2 y2], ..}`).

After the data is loaded, the global Fit box should be checked, and the instructions for performing Global Fits can be followed.

Custom PDFs

It is possible to enter a custom PDF to fit your data in two ways:

- Enter a PDF manually for one-time use
- Add a PDF to the drop down list to be used multiple times

Note: Any custom PDF must be properly normalized, so that the integrated area of the PDF is equal to one. If it is not normalized, the Maximum likelihood method will fail and incorrect and unstable parameter values will be returned. For more information, see the Theory Section of Accounting for Minimum or Maximum Detectable Event Size.

To enter a PDF manually for one-time use, select "Other" from the Select PDF drop down list, then type the desired PDF into the PDF box. The PDF should use Matlab operators and functions (+,-,*, /, sin(), exp(), etc.). (Multiplication, division, and exponentiation will be automatically assumed to be element-wise, so the dot operator [eg ".*", "./", ".^"] is not necessary and, in fact, will generate an error.). For complex functions, it is recommended to write the function in its own m-file somewhere in the Matlab path, then to reference that function in the PDF list box (for an example, the Triple Exponential Function is defined this way).

After entering in the PDF, both the data variables and fitting variables must be entered in their respective boxes. Each variable may consist of letters, numbers, and underscores. Variables should be separated by commas. Data Variables correspond to the experimental data being loaded, while Fitting Variables are the parameters that the program will adjust to optimize the fit. The number of data variables must be equal to the number of data columns (see Entering a Multidimensional PDF for more info). There can be an arbitrary number of fitting variables, but each variable in the PDF must be listed in one of the variable boxes and *vice versa*.

Including a t_{min} or t_{max} in a custom PDF

By applying the renormalization principle described in the Working with Deadtimes section, custom PDFs can be written which take into account the instrument's minimum detectable event. The keyword "tmin" should be used for its value in the custom PDF to take advantage of the t_{min} box in the graphical user interface. Similarly, the keyword "tmax" should be used for t_{max}. Examples of this notation can be seen in the PDFlist.m file by looking at how the built in PDF's are listed.

Adding a Custom PDF to the PDF list

Custom PDFs can be added to the PDF list by created a specially formatted .m file and placing it in the "PDFs" subfolder of the program. The program will scan this folder while loading and will import all properly formatted .m files in the folder into the PDF list. The Program will scan the Extra PDFs and Camera PDFs subfolders if the corresponding box in the Show PDFs menu is checked. A properly formatted m file will take one or more input arguments, and when called with one input argument will return a structure which contains the following fields: name, PDF, dataVar, fitVar, ub ,lb, guess.

This is illustrated below in the Simple_Custom_PDF_Example.m file, stored in the folder PDFs/ExtraPDFs. The highlighted sections of this file can be modified to create your own custom PDF that can be accessed in the program's PDF list when the file has been moved into the "PDFs" folder of the program.

```
function metaData=Simple_Custom_PDF_Example(arg1)
%the function should have atleast one argument and should return the
%metaData structure with the following fields:

% The metaData variable contains all the information about the PDF
% replace **Simple Example** with a name for the PDF to appear in the drop down list
  metaData=struct('name','Simple Example',...
% replace **k1*exp(-k1*t)** with the functional form of the PDF as a string,
            'PDF',    'k1*exp(-k1*x)',...
% replace **x** with a comma separated string of all the data Variables
            'dataVar','x',...
% replace **k1** with a comma separated string of all the fitting Variables
            'fitVar', 'k1',...
 % replace **100** with a comma separated string of the default upper bounds in the same
order as fitVar above
            'ub',    '100',...
% replace **0** with a comma separated string of the default lower bounds in the same
order as fitVar above
```

```
            'lb',    '0' ,...
% replace **1** with a comma separated string of the default guesses in the same order as
fitVar above
            'guess','1');
```

For more complex functions, including those that specify corrections for either a tmin or tmax, a more sophisticated m-file can be created. A template for such a file is provided as "Full_Custom_PDF_Template.m" in the ExtraPDFs folder. In addition, tripExpPDF.m can be viewed as a fully functioning example of such a file. It is important to note that when the main program reads the PDF information stored in the m. file, it calls the function with one argument. This argument is `limtype`, which specifies whether a tmin, tmax, or both have been specified in the GUI. Its value is given by the chart below:

`Limtype` value	**Number Specified in GUI**
0	Neither
1	tmin
2	tmax
3	Both tmin and tmax

This information is necessary when creating a PDF which handles the tmin and tmax cases. If users have questions about creating these custom PDF m files after looking at the examples and templates, please email memletinfo@gmail.com

A variety of PDFs have been provided in the PDFs/ExtraPDFs folder. To access these extra PDFs from the dropdown menu, simply move or copy the .m files from the ExtraPDFs folder into the PDFs folder. The custom .m files can be shared with other users, and may be distributed from the MEMLET website in the future.

Editing the PDF List

In addition to adding a PDF to the list, it is also possible to modify the initial guesses or limits of existing PDF's by finding the entry corresponding to the PDF of interest in the PDFList.m file and changing the values desired. Do so with care, because erroneous values which return Inf or NaN values could make the PDF unusable.

Command Line Access

The simulated annealing MLE fitting feature of the program can be accessed using custom Matlab scripts instead of the graphical user interface by calling the MLEFitCL.m function. This function is capable of fitting the supplied data to a given PDF subject to specified bounds and initial guesses. The input and outputs of this function are described below. The last argument is optional, and may specify an integer (NumBoot) to perform bootstrapping with numBoot round, or can be a string of variables (UniqueGlobalVar), specifying the unique global variables to be used in a global fit.

```
[fittedVals, logLikli] =  MLEFitCL(data, userPDF, dataVar, fitVar, lb, ub,
guess,annealTemp, NumBoot or UniqueGlobalVar)

Output Arguments:
fittedVals: an array of the fitted values from the MLE which is the same
length as fittedVars. If bootstrapping is employed, it is a matrix with
the width of fittedVars and length of the number of rounds of
bootstrapping requested. The fitted values of each round of boostrapping
are returned.

logLikli: The log-likelihood returned by the fit. This can be used for
log-likelihood ratio testing. For bootstrapping a column vector of
 log-likelihoods from each round is returned.

Input Arguments:

data: a column vector of data values. Global fitting is not supported in this command
line function.

userPDF: a string containing the PDF to be fit to the data, this string
should contain all the fit and data variables given.

dataVar: a string containing the data variable(s) separated by commas

fitVar: a string containing the fitting variable(s) separated by commas

lb: the lower bounds for the fitted variables (string with values separated by commas,
the number of values must equal to the number of fitted variables)

ub: the upper bounds for the fitted variables (string with values separated by commas,
the number of values must equal to the number of fitted variables)

guess: the initial guess for the fitted variables (numeric array with a length equal
to the number of fitted variables, or string with values separated by commas, the number
of values must equal to the number of fitted variables)

annealTemp: The initial annealing temperature

NumBoot (varargin): optional argument specifying the number of rounds of
bootstrapping to be performed.
```

This function can also be used along with the PDFList.m function which will return one of the built-in PDFs with default guesses and bounds as shown in the example below, which fits the data in the variable *data* to a double exponential PDF stored in the PDFList.m file with the default bounds and guesses.

```
[userPDF, dataVar, fitVar, lb,ub, guess]=PDFList('Double Exp (Independent)');
annealTemp=25;
[fittedVals, logLikli]= MLEFitCL(data, userPDF, dataVar, fitVar, lb,ub,
guess,annealTemp);
```

This function only fits the data and does not create plots or store the fitted parameters, but they may be accessed as normal within the Matlab environment.