LineFit: a Matlab Toolbox for 1D Line Fitting

Zhang Jiang
(Dated: October 29, 2016)

CONTENTS

I. Overview
II. Installation
III. Usage
   A. Script Mode
      1. Fitting a double-peak profile
      2. Customized or combined curve models
   B. GUI Mode
Appendices
   A. List of linefit properties
   B. List of linefit methods
   C. List of linefit built-in curve models
References

I. OVERVIEW

Linefit is a toolbox for 1D line fitting. In addition to over 30 built-in curve models, it allows combing those built-in model or customizing new models. As a class object, linefit’s properties and methods allow one to customize fitting options, easily switch on/off parameters for fitting, evaluate models, and calculate model properties. It provides both script mode for batch processing and GUI mode for interactive fitting.

Linefit works on Matlab 2015b or later. It requires Matlab Optimization Toolbox for lsqcurvefit solver.

II. INSTALLATION

Add the linefit directory to Matlab search path either from the Set Path in Matlab Home tab or run >>addpath <your path>/linefit in the command window. If Voigt distribution function is used, one needs add the Faddeeva sub-directory.

III. USAGE

We use ‘curve’ to denote the model for fitting and ‘bkgd’ to denote the model for background. They are stored as fields in private property linefit.ModelBase

CurveModel : [1x1 struct]
BkgdModel : [1x1 struct]

Each field itself is a structure consisting of model function handle, number of parameters, start values, lower and upper bounds, fit flags, and etc. For example, CurveModel has default fields:

ModelFcnHandle : @linefit.lorentzian
NOFParams : [3 1]
StartParams : [3x1 double]
LowerBounds : [3x1 double]
UpperBounds : [3x1 double]
FitFlags : [3x1 double]
ModelName : ‘Cauchy/Lorentzian’
ModelUsage : ‘lorentzian(x,[amp,x0,gamma])’
FitStdErrors : [3x1 double]

FitFlags is a logical matrix with 1 for fitting and 0 fixed parameters. FitStdErrors is a place holder to store the standard deviations after fitting; and its initial value is set to -1 (also an indicator for non-fitting parameters). ModelBase is a private property that cannot be directly changed by users. Instead, the curve and background models can be indirectly changed to other built-in models by setting properties CurveModelIndex and BkgdModelIndex (see Appendix A for details). After a new linefit object is created and loaded with 2-column data into property Data, ModelBase is automatically copied to a public property Model, whose field values such as start values, lower and upper bounds, and fit flags can be directly changed. Such way of managing models allows an easy setup for models with multiple curves of the same curve type (e.g. crystallography profiles with multiple peaks). This is achieved simply by adding or reducing number of curves in the Model.CurveMode (via method addcurves and reducecurves) while keeping the base curve model unchanged in ModelBase.

There is no better way to demonstrate than going through examples.

A. Script Mode

1. Fitting a double-peak profile

Create a linefit object and load data

foo=load(‘demo_doublpeak.dat’)
a=linefit(foo)
Instead of default settings of Lorentzian curve mode and no background, we two Gaussian curves and constant background.

```matlab
a.CurveModelIndex=4 % Gaussian
a.BkgdModelIndex=1 % constant background
a=addcurves(a,1) % add a curve to the end
```

After method `addcurves` appends another Gaussian curve to the model, a new `linefit` is returned and assigned back to the original object `a`.

To preview the model with start parameter values,

```matlab
plot(a) or plot(a,'preview')
```

`Linefit` uses built-in Matlab solver `lsqcurvefit`. The optimization options are identical to those for `lsqcurvefit`, and can be accessed or changed in property `FitOptions`.

To start fitting,

```matlab
a=startfit(a)
```

Fit results, such as fitted parameters and their standard deviations, squared norm of the residual, residuals, exit flag (reason to stop fitting), output (information about fitting process), Jacobian at the solution, and etc., are temporarily saved in `FitOutput` waiting for acceptance or rejection.

```matlab
plot(a,'fit')
```

plots the the fitted curve on the raw data, as well as the residuals.

```matlab
a=acceptfit(a)
```

```matlab
a.CustomCurveModel=linefit.createmodel(@(x,p)p(1)*exp(-abs(x-p(2)).^3/p(3)),3,[1,0,5],[0,-1,0],[100,10,Inf],[1,1,1])
```

where the 1st argument is an anonymous function handle whose expression must be vectorized (e.g. using the ‘.’ for the power, multiplication, division etc), the 2nd argument is the total number of parameters, and the 3rd-6th arguments are start values, lower bounds, upper bounds, and fit flags. The 3rd-6th arguments can be empty [ ] to use the default settings: random values for start values, -Inf’s for lower bounds, Inf’s for upper bounds, and 1s for fit flags. One can also create a customized model by working out the function in a Matlab .m file. Let’s say file `myfun.m` has the function defined by the following lines

```matlab
function y=myfun(x,p)
y=p(1)*exp(-abs(x-p(2)).^3/p(3));
```

This file has to be in the current working directory or wherever Matlab can see it from its path settings. We then create the model by

```matlab
a.CustomCurveModel=linefit.createmodel(@(x,p)myfun,3,[1,0,5],[0,-1,0],[100,10,Inf],[1,1,1])
```

The customized model is then loaded to `ModelBase` by

```matlab
a.CurveModelIndex=-2;
```
We then add one more curve of the same model and start the fitting

```matlab
a = addcurves(a, 1);
```
```matlab
a = a.startfit;
```

Note that analytical model properties are not defined for customized models; instead, use numerical method.

If one Lorentzian and one Gaussian are desired to model the double peak data, we can combine these two built-in models by creating a multi-model

```matlab
a.CustomCurveModel = a.creatmodel([1, 4], [], [], [], [], []);
```

where the 1st argument [1, 4] are the index for Lorentzian and Gaussian. The number of parameters is automatically set to the total number of parameters of these two built-in models; thus the 2nd argument is left empty. The other arguments are also left empty to use the default settings. We then load this multi-model to ModelBase by setting the model index to -1

```matlab
a.CurveModelIndex = -1;
```

If you get to this step by continuing the last example where two curves are used, you need to reduce the number of curves to one, because the multi-model itself is a double-peak profile although it is considered as a one curve model.

```matlab
a = reducecurves(a, 2)
```

where the 2nd argument is the index(es) of the current curves to be removed in Model.CurveModel.

Customized background model can be created in the same way as the customized curve model, but one cannot combine built-in background models. Multiple background models of the same type are not allowed either.

**B. GUI Mode**

Although linefit toolbox is fully functional in the script model, a GUI interface (linefitgui) provides a faster access if there are not too many data sets to be managed. Let's demonstrate with an example. Start GUI and load data

```matlab
foo = load('demo_xtal.data');
```
```matlab
linefitgui(foo)
```

One also start GUI first by linefitgui and load data from there. Complete the fitting by following the steps in Fig. 1.

To use customized model or multi-mode in GUI mode, enter the arguments only separated by commas in the Customization edit boxes, e.g.,

```matlab
@(x,p)p(1)*exp(-abs(x-p(2)).^3/p(3)),
3,[1,0,5],[0,-1,0],[100,10,Inf],[1,1,1]
```
or

```matlab
[1,4],[],[],[],[],[]
```

There is no need to enter the method name createmodel. Then use the Select Model popup menu to change the model index to customized model or multi-model.

**Appendices**

**Appendix A: List of linefit properties**

In Matlab command window, type linefit to view the default properties of a linefit object.

```matlab
Data : []
CurveModelIndex : 1
BkgdModelIndex : 0
CustomCurveModel : [1x1 struct]
CustomBkgdModel : [1x1 struct]
Model : [1x1 struct]
FlagNonNegBkgd : 1
FlagLogScale : 0
FitOptions : [1x1 optim.options.Lsqcurvefit]
FitOutput : [1x1 struct]
ModelList : [1x30 struct]
PeaksFound : []
ModelBase : [1x1 struct]
NOfModelCurves : 1
```

Public properties that are independent and can be set or accessed by users are

- **Data**: (Public) Input data. Must be a 2-column double or single type data [XData,YData].
- **CurveModelIndex**: (Public) Index of curve models. In Matlab command window, type linefit.getmodellist to display all models, usage and initial parameters.
- **BkgdModelIndex**: (Public) Index of background models. -2/-1/[0]/1/2/3.../N: Custom/Power/[No]/Constant/Linear/Quadratic/.../Polynomial of degree N.
- **CustomCurveModel**: (Public) Structure to store custom or multi curve model. Can be created with createmodel method.
- **CustomBkgdModel**: (Public) Structure to store custom background model. Can be created with createmodel method.
- **Model**: (Public) Complete fit model including both curve and background models (CurveModel and BkgdModel).

- **FlagNonNegBkgd**: (Public) Flag to force a non-negative background. 0/[1]: no/[yes].

- **FlagLogScale**: (Public) Flag to fit in log scale for YData. [0]/[1]: [no]/[yes].

- **FitOptions**: (Public) Optimization options for solver lsqcurvefit. See Matlab help or documentation for lsqcurvefit.

- **FitOutput**: (Public) Structure to store fitting result.
• Modellist: (Private) Definition of built-in models.
• PeaksFound: (Private) Peak value and positions \([Y, X]\) for the initialization of peak-shaped models.
• ModelBase: (Private) Base curve model and background model for fitting. ModelBase.CurveModel is replicated to Model.CurveModel if multiple identical curves are used (NOfModelCurves). ModelBase can be changed only via CurveModelIndex.
• NOfModelCurves: Number of curves. It can be increased or decreased by methods addcurves and reducecurves.

Appendix B: List of linefit methods

Dynamic methods can be called in two ways: obj.MethodName(argument1,...) or MethodName(obj,argument1,...). The 2nd way will be adopted below. Static methods can be called like directly called by linefit.MethodName(argument1,...) without first creating an object.

• linefit (dynamic): Create a new linefit object.
  obj=linefit creates an empty object.
  obj=linefit(data) creates an object with data and default settings.

  obj2=addcurves(obj1) adds one curve to the end.
  obj2=addcurves(obj1,n) adds n curves to the end.
  obj2=addcurves(obj1,n,pos) adds n curves to position pos. pos must be a non-zero integer. The value of pos (i.e. |pos|) is the position; the sign of pos is the direction (i.e. + means an insertion after |pos| and - means an insertion before |pos|).

  This method creates a new object obj2. Setting obj2 to obj1 as the output argument avoids creating a new object.

• reducecurves (dynamic): Reduce curves from Model.CurveModel.
  obj2=reducecurves(obj1) reduces the last one curve in the list end.
  obj2=reducecurves(obj1,index) reduces curves of position index. index must be a scalar or a vector, e.g., [2,5:7] removes curve #2, #5, #6, and #7.

• sortcurves (dynamic): Sort the order of curves in Model.CurveModel.
  obj2=sortcurves(obj1,row) sorts the curves with default mode='ascend' according to the values in row number row, i.e. the index position of the parameters. For example, row=2 sorts the order of the two curves by the values of the parameters values in p(2) in the example in Section II A 2.
  obj2=sortcurves(obj1,row,mode) sorts with mode='ascend' or 'descent'.

• startfit (dynamic): Start fitting.
  obj2=startfit(obj1) starts fitting and stores fit result in obj2.FitOutput.

• acceptfit (dynamic): Accept and pass fit result to fields StartParams and FitStdErrors in Model.CurveModel and Model.BkgdModel.
  obj2=acceptfit(obj1) returns to a new object obj2.

• evalmodel (dynamic): Evaluate Model.
  y=evalmodel(obj) evaluates using XDATA in obj.Data and returns values to y.
  y=evalmodel(obj,XV) evaluates on a vector XV.
  [y,iy]=evalmodel(obj,...) also returns the values for each curve to iy.
  [y,iy,bkgd]=evalmodel(obj,...) also returns the background values to bkgd.

• plot (dynamic): Plot model calculations with start parameters or fitted parameters.
  plot(obj) preview model calculations using parameters in obj.Model.
  plot(obj,mode) plots model calculations when mode='preview', and fitting result when mode='fit'. Default is mode='preview'. The 'fit' mode also plots the residual and labels the Pearson's \(\chi^2\), which are defined by

\[
\text{Residual}(x_i) = f(x_i) - y_i
\]

\[
\chi^2 = \sum_i \frac{[f(x_i) - y_i]^2}{y_i}
\]

  hlines=plot(obj,...) returns the handle of the plotted data line.

• findpeakauto (dynamic): Automatically search for peaks in Data.
  obj2=findpeakauto(obj1) uses function peakfinder.m to find peaks\(\text{\[P\]}\) and stores them to obj2.peaksfound.
  obj2=findpeakauto(obj1, arg1, arg2...) uses arguments defined in peakfinder.
• **findpeakmanual** (dynamic): Manually and interactively search for peaks.
  
  `obj2=findpeakmanual(obj1)` uses interactive tool `getpts` (requires Matlab Image Processing Toolbox) to identify peaks.

• **getmodelprops** (dynamic): Get curve model properties such as mean, variance, FWHM, skewness, and kurtosis.

  `s=getmodelprops(obj)` gets curve model properties using analytical method and returns to a structure `s`.

  `s=getmodelprops(obj,XV)` numerically calculates model properties using `XV`. `XV` must be a vector. Suppose function `f(x)` is normalized, i.e., `\int f(x)dx = 1`. Mean value is then calculated by

  \[
  \mu = \int x f(x)dx;
  \]

  the variance (the 2nd central momentum) is calculated by

  \[
  \sigma^2 = \int (x - \mu)^2 f(x)dx;
  \]

  skewness is calculated by

  \[
  \gamma_1 = \frac{\mu_3}{\sigma^3},
  \]

  where \( \mu_3 = \int (x - \mu)^3 f(x)dx \) is the 3rd central momentum, and \( \sigma = \sqrt{\mu_2} \) is the standard deviation; and excess kurtosis is calculated by

  \[
  \gamma_2 = \frac{\mu_4}{\sigma^4} - 3,
  \]

  where \( \mu_4 = \int (x - \mu)^4 f(x)dx \) is the 4th central momentum.

• **applypeaks** (dynamic): Pass peak values and positions in PeaksFound to `Model.CurveModel.StartParams`, and set the number of curves to the number of peaks found.

  `obj2=applypeaks(obj1)` passes to the 1st and 2nd input parameters (peak amplitudes and positions) for built-in curve models #1-18 (peak-shaped models).

  `obj2=applypeaks(obj1,rows)` passes to rows of `Model.CurveModel.StartParams`. `rows` must be a two-element vector with the 1st value for peak amplitude row and 2nd value for peak location row. For example, `rows=[3,4]` means that, in the model, the 3rd parameter is the peak amplitude and the 4th parameter is the peak location (x value).

• **createmodel** (static): Create model structure for `linefit` object

  \[
  f(4,[2,1,3])=3.7296\]

  \[
  x=4\text{ and } p=[2,1,3], \text{i.e. } f(4,[2,1,3])=3.7296\]

  For a multi-model (`CurveModelIndex=-1`) where the `Model.CurveModel` is constructed as a sum of built-in models, use

  `s=createmodel(models,[],startparams,lowerbounds,upperbounds,fitflags)` where the `models` is a list of indexes of built-in models, e.g., `models=[1,4,13]` constructs a multi-model using the sum of Lorentzian (model #1), Gaussian (model #4) and Voigt (model #13). The default of `nofp` is automatically set to the total number of parameters of these built-in models.

  `s=createmodel(...,fitflags,modelname)` takes a name (string) for the model. It is automatically set to empty if not specified, or forced to be built-in model indexes when `CurveModelIndex=-1`.

  `s=createmodel(...,modelname, modelusage)` takes usage instructions (string) for the model.

• **lineprops** (static): Numerically calculate line properties.

  `s=obj.lineprops(data)` or `s=linefit.lineprops(data)` calculates properties for a 2-column data. It outputs a structure `s` with fields: `s.area` for numerically integrated area (using Matlab `trapz` function); `s.peak` for the peak X, Y values, and position index; `s.mean`, `s.variance`, `s.skewness`, `s.kurtosis`, and `s.fwhm` for mean, variance, skewness, kurtosis, and full-width-of-half-maximum (FWHM); and `s.fwhm_x` for the center position of FWHM.

• **getmodellist** (static): Get the list of built-in models with name, usage, initial parameters, bounds and etc.

  `s=obj.getmodellist` or `s=linefit.getmodellist` returns to a table `s`. 

- skewness is calculated by
- \( \text{obj2}=\text{applypeaks} (\text{obj1}, \text{rows}) \) passes to rows of `Model.CurveModel.StartParams`. `rows` must be a two-element vector with the 1st value for peak amplitude row and 2nd value for peak location row. For example, `rows=[3,4]` means that, in the model, the 3rd parameter is the peak amplitude and the 4th parameter is the peak location (x value).
Appendix C: List of linefit built-in curve models

These functions are defined as static methods, which can be called the same way as other linefit static methods. General usage of these functions is \( y = \text{linefit}.\text{function}_\text{name}(x, p) \) or \([y, \text{props}] = \text{linefit}.\text{function}_\text{name}(x, p)\), where \( p \) is the parameter vector defined below, and \( \text{props} \) is a structure for model properties, such as mean, variance and etc.

Note that not all functions have analytical formulas for mean, variance, skewness, kurtosis, or FWHM. Although one may force a calculation of these properties by numerically evaluating the model (see linefit method getmodelprops), properties such as skewness and kurtosis may not be mathematically defined at all.

Most distribution functions are normalized unless explicitly claimed.

There are 31 built-in models. Models #1-18 are distribution functions with support \((-\infty, \infty)\); models #19-28 are distribution functions with half-infinity support; the rest of the models are other profile functions.

1. lorentzian \((x, [A, x0, gamma])\) Lorentzian or Cauchy distribution function
   \( A \): amplitude, \( x0 \): location, \( gamma \) \((\gamma > 0)\): scale or HWHM (half-width-of-half-maximum).
   
   \[
   f(x) = A \frac{1}{\pi \gamma} \left[ 1 + \left( \frac{x - x0}{\gamma} \right)^2 \right]^{-1}
   \]
   
   mode : \( x0 \)
   median : \( x0 \)
   fwhm : \( 2\gamma \)

2. ilorentzian \((x, [A, x0, gamma])\) Intermediate Lorentzian distribution function
   \( A \): amplitude, \( x0 \): location, \( gamma \) \((\gamma > 0)\): scale.
   
   \[
   f(x) = A \frac{\sqrt{2} - 1}{2\gamma} \left[ 1 + (2^2 - 1) \left( \frac{x - x0}{\gamma} \right)^2 \right]^{-\frac{3}{2}}
   \]
   
   mode : \( x0 \)
   median : \( x0 \)
   fwhm : \( 2\gamma \)

3. mlorentzian \((x, [A, x0, gamma])\) Modified Lorentzian distribution function
   \( A \): amplitude, \( x0 \): location, \( gamma \) \((\gamma > 0)\): scale.
   
   \[
   f(x) = A \frac{2\sqrt{2} - 1}{\pi \gamma} \left[ 1 + \left( \sqrt{2} - 1 \right) \left( \frac{x - x0}{\gamma} \right)^2 \right]^{-2}
   \]
   
   mode : \( x0 \)
   median : \( x0 \)
   fwhm : \( 2\gamma \)

4. gaussian \((x, [A, x0, sigma])\) Gaussian or normal distribution function
   \( A \): amplitude, \( x0 \): location, \( sigma \) \((\sigma > 0)\): scale or standard deviation.
   
   \[
   f(x) = A \frac{1}{\sigma \sqrt{2\pi}} \exp \left[ -\frac{(x - x0)^2}{2\sigma^2} \right]
   \]
   
   mean : \( x0 \)
   variance : \( \sigma^2 \)
   skewness : 0
   kurtosis : 0
   mode : \( x0 \)
   median : \( x0 \)
   fwhm : \( 2\sqrt{2\log(2)}\sigma \)

5. generalizednorm1 \((x, [A, a, b, c])\) Generalized normal distribution function version 1
   \( A \): amplitude, \( a \): location, \( b > 0 \): scale, \( c > 0 \): shape.
   
   \[
   f(x) = A \frac{c}{2\Gamma(1/c)} \exp \left[ -\left( \frac{|x - a|}{b} \right)^c \right]
   \]
   
   mean : \( a \)
   variance : \( \frac{b^c \Gamma(3/c)}{\Gamma(1/c)} \)
   skewness : 0
   kurtosis : \( \frac{\Gamma(5/c) \Gamma(1/c)}{\Gamma(3/c)^2} - 3 \)
   mode : \( a \)
   median : \( a \)

6. laplace \((x, [A, a, b])\) Laplace distribution function
   \( A \): amplitude, \( a \): location, \( b > 0 \): scale.
   
   \[
   f(x) = A \frac{1}{2b} \exp \left( -\frac{|x - a|}{b} \right)
   \]
   
   mean : \( a \)
   variance : \( 2b^2 \)
   skewness : 0
   kurtosis : 3
   mode : \( a \)
   median : \( a \)

7. logistic \((x, [A, a, b])\) Logistic distribution function
   \( A \): amplitude, \( a \): location, \( b > 0 \): scale.
   
   \[
   f(x) = A \frac{1}{b} \exp \left( -\frac{x - a}{b} \right) \left[ 1 + \exp \left( -\frac{x - a}{b} \right) \right]^{-2}
   \]
mean : a
variance : \(\pi^2 b^2 / 3\)
skewness : 0
kurtosis : 7/5
mode : a
median : a

9. **pseudovoigt\(TCH(x,[A,a,b,c])\)** Pearson type VII or student’s t distribution function\(^{[17]}\)

\(A\): amplitude, \(a\): location, \(b > 0\): scale, \(c > 0\): shape (degree of freedom).

\[
f(x) = A \frac{\Gamma(\frac{c+1}{2})}{b\sqrt{\pi}c\Gamma(\frac{c}{2})} \left[ 1 + \frac{1}{c} \left( \frac{x-a}{b} \right)^2 \right]^{-\frac{c+1}{2}}
\]

mean : a for \(c > 1\), otherwise undefined
variance : \(\frac{\sigma^2}{c-2}\) for \(c > 2\), infinite for \(1 < c \leq 2\), otherwise undefined
skewness : 0 for \(c > 3\), otherwise undefined
kurtosis : \(\frac{6}{c-4}\) for \(c > 4\), infinite for \(2 < c \leq 4\), otherwise undefined
mode : a
median : a

with \(\sigma_G = \Gamma / (2\sqrt{2\log(2)})\); \(\eta_L\) is the contribution from the Lorentzian to the total Voigt

\[
\eta_L = 1.36603 \frac{\Gamma_{LV}}{\Gamma} - 0.47719 \left( \frac{\Gamma_{LV}}{\Gamma} \right)^2 + 0.1116 \left( \frac{\Gamma_{LV}}{\Gamma} \right)^3.
\]

fwhm : \(\Gamma\)

10. **pseudovoigt\(IAT(x,[A,x0,\text{sigma},\text{gamma}])\)** Pseudo-Voigt type IAT distribution function\(^{[9]}\)

\(A\): amplitude, \(x_0\): location, \(\text{sigma} (\sigma > 0)\): scale (standard deviation) of the Gaussian for convolution, \(\text{gamma} (\gamma > 0)\): scale (HWHM) of the Lorentzian for convolution.

\[
f(x) = A[(1 - \eta_L - \eta_I - \eta_P)f_G(x; \sigma_G) + \eta_L f_L(x; \gamma_L) + \eta_I f_I(x; \gamma_I) + \eta_P f_P(x; \gamma_P)],
\]

where \(f_G(x; \sigma_G)\) is a Gaussian with FWHM \(\Gamma_G = 2\sqrt{2\log(2)}\sigma_G\); \(f_L(x; \gamma_L)\) is a Lorentzian with HWHM \(\gamma_L\). FWHM \(\Gamma_L = 2\gamma_L\), and contribution \(\eta_L\); \(f_I(x; \gamma_I)\) is an irrational function (equivalent to the intermediate Lorentzian) defined by

\[
f_I(x; \gamma_I) = \frac{1}{2\gamma_I} \left[ 1 + \left( \frac{x-x_0}{\gamma_I} \right)^2 \right]^{-\frac{3}{2}},
\]

with FWHM \(\Gamma_I = 2\sqrt{2^{3/2} - 1}\gamma_I\) and contribution \(\eta_I\); and \(f_P(x; \gamma_P)\) is an inverse squared hyperbolic cosine function defined by

\[
f_P(x; \gamma_P) = \frac{1}{2\gamma_P \cosh^2 \left( \frac{x-x_0}{\gamma_P} \right)},
\]

with FWHM \(\Gamma_P = 2\log(\sqrt{2} + 1)\gamma_P\) and contribution \(\eta_P\). These FWHMs are calculated via

\[
\Gamma_i = (\Gamma_{GV} + \Gamma_{LV})(\omega_i, \text{ where } \Gamma_{GV} = 2\sqrt{2\log(2)}\sigma, \Gamma_{LV} = 2\gamma, \text{ and } i \in \{G, L, I, P\}). With \rho \text{ defined as } \rho \equiv \Gamma_{LV}/(\Gamma_{LV} + \Gamma_{GV}), w_i \text{ and } \eta_i \text{ are calculated via}
\]

\[
w_G = 1 - \rho \sum_{i=0}^{N} a_i \rho^i,
\]

\[
w_L = 1 - (1 - \rho) \sum_{i=0}^{N} b_i \rho^i,
\]

\[
w_I = \sum_{i=0}^{N} c_i \rho^i,
\]

\[
w_P = \sum_{i=0}^{N} d_i \rho^i,
\]

\[
\eta_L = \rho \left[ 1 + (1 - \rho) \sum_{i=0}^{N} f_i \rho^i \right],
\]

\[
\eta_P = \rho \left[ 1 + (1 - \rho) \sum_{i=0}^{N} g_i \rho^i \right],
\]

\[
\eta_I = \rho \left[ 1 + (1 - \rho) \sum_{i=0}^{N} h_i \rho^i \right],
\]

\[
\eta_G = \rho \left[ 1 + (1 - \rho) \sum_{i=0}^{N} j_i \rho^i \right].
\]
\[ \eta_t = \rho(1 - \rho) \sum_{i=0}^{N} g_i \rho^i, \]

\[ \eta_t = \rho(1 - \rho) \sum_{i=0}^{N} h_i \rho^i, \]

where \( N = 6 \), and polynomial coefficients \( a_i, b_i, c_i, d_i, f_i, g_i \) and \( h_i \) are

<table>
<thead>
<tr>
<th></th>
<th>( a_i )</th>
<th>( b_i )</th>
<th>( c_i )</th>
<th>( d_i )</th>
<th>( f_i )</th>
<th>( g_i )</th>
<th>( h_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.66</td>
<td>-0.42179</td>
<td>1.19913</td>
<td>1.10186</td>
<td>-0.30165</td>
<td>0.25437</td>
<td>1.01579</td>
</tr>
<tr>
<td>1</td>
<td>0.15021</td>
<td>-1.25693</td>
<td>1.43021</td>
<td>-0.47745</td>
<td>-1.38927</td>
<td>-0.14107</td>
<td>1.50429</td>
</tr>
<tr>
<td>3</td>
<td>4.74052</td>
<td>-23.45651</td>
<td>47.06071</td>
<td>-2.76622</td>
<td>7.21045</td>
<td>-23.59717</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8.48252</td>
<td>-16.50453</td>
<td>57.92559</td>
<td>-4.05475</td>
<td>-32.83023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-2.95553</td>
<td>3.19974</td>
<td>-17.80614</td>
<td>9.76947</td>
<td>-10.02142</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

fwhm : \( \Gamma \) (same as \texttt{pseudovoigtTCH})

11. \texttt{pseudovoigtLLHGD(x, [A, x0, sigma, gamma])} Pseudo-Voigt type LLHGD distribution function[11]

\( A \): amplitude, \( x0 \): location, \( \texttt{sigma} (\sigma > 0) \): scale (standard deviation) of the Gaussian for convolution, \( \texttt{gamma} (\gamma > 0) \): scale (FWHM) of the Lorentzian for convolution.

\[ f(x; \Gamma) = c_G f_G(x; \sigma_G) + c_L f_L(x; \gamma_L), \]

where \( \Gamma \) is the approximated FWHM of Voigt[11]

\[ \Gamma = 0.5346 \Gamma_{LV} + \sqrt{0.2165975 \Gamma_{LV}^2 + \Gamma_{GV}^2}, \]

with \( \Gamma_{LV} = 2\gamma \) and \( \Gamma_{GV} = 2\sqrt{2\log(2)}\sigma; f_G(x; \sigma_G) \) is a Gaussian with \( \sigma_G = \Gamma/(2\sqrt{2\log(2)}) \); \( f_L(x; \gamma_L) \) is a Lorentzian with \( \gamma_L = \Gamma/2 \); and coefficients \( c_G \) and \( c_L \) are given by

\[ c_G = 0.32460 - 0.61825d + 0.17681d^2 + 0.12109d^3, \]
\[ c_L = 0.68188 + 0.61293d - 0.18384d^2 - 0.11568d^3, \]

where \( d = (\Gamma_{LV} - \Gamma_{GV})/(\Gamma_{LV} + \Gamma_{GV}) \).

fwhm : \( \Gamma \)

12. \texttt{uniform(x, [A, a, b])} Uniform distribution function[12]

\( A \): amplitude, \( a \): location, \( b \) \((b > a)\): scale (upper bound)

\[ f(x) = \begin{cases} \frac{A}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise} \end{cases} \]

mean : \((a + b)/2\)

variance : \((b - a)^2/12\)

skewness : 0

kurtosis : \(-6/5\)

median : \((a + b)/2\)

13. \texttt{voigt(x, [A, x0, sigma, gamma])} Voigt distribution function[13]

\( A \): amplitude, \( x0 \): location, \( \texttt{sigma} (\sigma > 0) \): scale (standard deviation) of the Gaussian for convolution, \( \texttt{gamma} (\gamma > 0) \): scale (FWHM) of the Lorentzian for convolution.

\[ f(x) = \int_{-\infty}^{\infty} f_G(x'; \sigma) f_L(x - x'; \gamma) dx' = \frac{\text{Re}[w(z)]}{\sigma \sqrt{2\pi}}, \]

where \( f_G \) and \( f_L \) are a Gaussian and a Lorentzian, \( z = \frac{x-x_0+ix}{\sigma \sqrt{2}} \), and \( \text{Re}[w(z)] \) is the real part of the Faddeeva function (a.k.a. Kramp function) which is a scaled complex complementary error function defined as

\[ w[z] = \exp(-z^2) \text{erfc}(-iz) = \text{erf}(-iz) \]
\[ = \exp(-z^2) \left[ 1 + \frac{2z}{\sqrt{\pi}} \int_{0}^{\infty} \exp(t^2) dt \right] \]

fwhm : \( \Gamma \) (same as \texttt{pseudovoigtTCH})

14. \texttt{skewlaplace(x, [A, a, b, c])} Skew-Laplace distribution function[14]

\( A \): amplitude, \( a \): location, \( b > 0 \): scale, \( c > 0 \): scale.

\[ f(x) = A \frac{1}{b + c} \begin{cases} \exp \left( \frac{x-a}{b} \right), & \text{for } x \leq a \\ \exp \left( \frac{-a-x}{c} \right), & \text{for } x > a \end{cases} \]

mean : \( a - b + c \)

variance : \( b^2 + c^2 \)
17. skewpvsb(x, [A, x0, fwhm0, eta, a]) Asymmetric pseudo-Voigt type SB distribution function

A: amplitude, x0: location, fwhm0 ( Γ₀ > 0): FWHM of symmetric Voigt, eta (γ ≥ 0): contribution of Lorentzian, a: degree of asymmetry, b: location shift of sigmoidal function.

This distribution has identical formula as skewpvsB, except that the modified FWHM is

Γ𝑀 = \frac{2Γ₀}{1 + \exp[-a(x - x₀)]}.

Note that Γ𝑀 is not FWHM of the distribution; the FWHM has to be evaluated numerically. Also note that this distribution is not normalized, i.e. the amplitude A is not the total area of the distribution.

18. expdist(x, [A, a, b]) Exponential distribution function

A: amplitude, x0: location, μ = \exp(-\frac{x-a}{b}) is the beta function (a.k.a. the Euler integral of the first kind).

Note that Γ𝑀 is not FWHM of the distribution; the FWHM has to be evaluated numerically. Also note that this distribution is not normalized, i.e. the amplitude A is not the total area of the distribution.
A: amplitude, a: location, b>0: scale.

\[ f(x) = A \begin{cases} 
0, & \text{for } x < a \\
\frac{1}{b} \exp \left( -\frac{x-a}{b} \right), & \text{for } x \geq a 
\end{cases} \]

mean : \( a + b \)

variance : \( b^2 \)

skewness : 2

kurtosis : 6

mode : a

median : a + b \log(2)

21. `gammadist(x,[A,a,b,c])` Gamma distribution function

A: amplitude, a: location, b>0: scale, c>0: shape

\[ f(x) = A \begin{cases} 
0, & \text{for } x \leq a \\
\frac{1}{\Gamma(c)} \left( \frac{x-a}{b} \right)^{c-1} \exp \left( -\frac{x-a}{b} \right), & \text{for } x > a 
\end{cases} \]

mean : \( a + bc \)

variance : \( b^2c \)

skewness : \( 2/\sqrt{c} \)

kurtosis : \( 6/c \)

mode : \( a + b(c-1) \) for \( c \geq 1 \), otherwise \( a \)

median : \( a + (b/2)/[\text{erfc}^{-1}(1/2)]^2 \)

where \( \text{erfc}^{-1}(x) \) is the inverse error function.

22. `inversegamma(x,[A,x0,a,b])` Inverse gamma distribution function

A: amplitude, x0: location, a>0: scale, b>0: shape.

\[ f(x) = A \begin{cases} 
0, & \text{for } x \leq x_0 \\
\frac{1}{\Gamma(b)} (x-x_0)^{-b-1} \exp \left( -\frac{a}{x-x_0} \right), & \text{for } x > x_0 
\end{cases} \]

mean : \( x_0 + \frac{a}{b+1} \) for \( b > 1 \)

variance : \( \frac{a^2}{(b-2)(b-1)} \) for \( b > 2 \)

skewness : \( \frac{4\sqrt{b-2}}{b-3} \) for \( b > 3 \)

kurtosis : \( \frac{30b-66}{(b-4)(b-3)} \) for \( b > 4 \)

mode : \( x_0 + \frac{a}{b+1} \)

median : \( x_0 + \exp(a) \)

23. `inversenorm(x,[A,x0,a,b])` Inverse normal distribution (a.k.a Wald distribution) function

A: amplitude, x0: location, a>0: location and scale, b>0: scale.

\[ f(x) = A \begin{cases} 
0, & \text{for } x \leq x_0 \\
\sqrt{\frac{b}{2\pi(x-x_0)^3}} \exp \left( -\frac{b}{2(x-x_0)^2} \right), & \text{for } x > x_0 
\end{cases} \]

mean : \( x_0 + a \)

variance : \( a^3/b \)

skewness : \( 3\sqrt{a/b} \)

kurtosis : \( 15a/b \)

mode : \( x_0 + \frac{a}{b} (\sqrt{9a^2 + 4b^2} - 3a) \)

24. `levy(x,[a,b])` Lévy distribution function

A: amplitude, a: location, b>0: scale.

\[ f(x) = A \begin{cases} 
0, & \text{for } x \leq a \\
\sqrt{\frac{b}{2\pi(x-a)^3}} \exp \left( -\frac{b}{2(x-a)^2} \right), & \text{for } x > a 
\end{cases} \]

mean : \( \infty \)

variance : \( \infty \)

mode : \( a + b/3 \)

median : \( a + (b/2)/[\text{erfc}^{-1}(1/2)]^2 \)

25. `logcauchy(x,[A,x0,a,b])` Log-Cauchy distribution function

A: amplitude, x0: location, a: log-location, b>0: scale.

\[ f(x) = A \begin{cases} 
0, & \text{for } x \leq x_0 \\
\frac{1}{\pi(x-x_0)} \exp \left( -\frac{b}{\log(x-x_0)-a}^2 \right), & \text{for } x > x_0 
\end{cases} \]

variance : \( \infty \)

median : \( x_0 + \exp(a) \)

26. `lognorm(x,[A,x0,a,b])` Log-normal distribution

A: amplitude, x0: location, a: log-location, b>0: scale. Both a and b are measured in log space.

\[ f(x) = A \begin{cases} 
0, & \text{for } x \leq x_0 \\
\frac{1}{(x-x_0)\sqrt{2\pi}} \exp \left( -\frac{\log(x-x_0)-a}^2 \right), & \text{for } x > x_0 
\end{cases} \]

mean : \( x_0 + \exp(a + b^2/2) \)

variance : \( [\exp(b^2) - 1] \exp(2a + b^2) \)

skewness : \( [\exp(b^2) + 2] \sqrt{\exp(b^2) - 1} \)

kurtosis : \( \exp(4b^2) + 2 \exp(3b^2) + 3 \exp(2b^2) - 6 \)

mode : \( x_0 + \exp(a - b^2) \)

median : \( x_0 + \exp(a) \)
27. \texttt{paretoI(x,[A,a,b])} Pareto type I distribution function.\(^{223}\)
A: amplitude, a \(> 0\): location and scale, b \(> 0\): scale.

\[
f(x) = \begin{cases} 
 0, & \text{for } x < a \\
 0, & \text{for } x \geq a \\
 \frac{ab}{b-a} \cdot \frac{b}{c-c^2}, & \text{for } x \geq a
\end{cases}
\]

mean : \(\frac{ab}{b-a}\) for \(b > 1\), otherwise \(\infty\)

variance : \(\frac{a^2 b}{(b-1)(b-2)}\) for \(b > 2\), \(\infty\) for \(b \in (0, 2]\)

skewness : \(\frac{2(1+b)}{b^3 - 2b}\) for \(b > 3\)

kurtosis : \(\frac{6(b^3+b^2-6b-2)}{b(b-3)(b-4)}\) for \(b > 4\)

mode : \(a\)

median : \(a \sqrt{b}\)

28. \texttt{weibull(x,[A,a,b,c])} Weibull distribution function.\(^{224}\)
A: amplitude, a: location, b > 0: scale, c > 0: shape.

\[
f(x) = \begin{cases} 
 0, & \text{for } x < a \\
 \frac{c}{b} \left(\frac{x-a}{b}\right)^{c-1} \exp[-\left(\frac{x-a}{b}\right)^c], & \text{for } x \geq a
\end{cases}
\]

mean : \(a + b G(1)\)

variance : \([-G^2(1) + G(2)] b^2\)

\(G(n) = \frac{\Gamma(n)}{n!}\)

29. \texttt{atan(x,[A,a,b])} Inverse tangent function
A: amplitude, a: location, b: scale.

\[
f(x) = A \tan \left(\frac{x-a}{b}\right)
\]

30. \texttt{erf(x,[A,a,b])} Error function
A: amplitude, a: location, b: scale.

\[
f(x) = A \text{erf} \left(\frac{x-a}{b}\right)
\]

31. \texttt{powerlaw(x,[A,a,b])} Power-law function
A: amplitude or scale, a: location, b: shape.

\[
f(x) = \begin{cases} 
 0, & \text{for } x \leq a \\
 (x-a)^{-b}, & \text{for } x > a
\end{cases}
\]