LineFit: a Matlab Toolbox for 1D Line Fitting

Zhang Jiang
(Dated: October 29, 2016)

CONTENTS

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[1] 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

```
a.CuveModelIndex=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

`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,

```
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.

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

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

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

accepts fit result and updates start values (`StartParams`) and standard deviations (`FitStdErrors`) for both `CurveModel` and `BkgdModel`. Nothing needs to be done to reject.

You can then get the model properties (if defined for the model) such as mean, variance, median, skewness, kurtosis, and etc, either analytically or numerically,

```
getmodelprops(a)  % analytical method
getmodelprops(a,linspace(-100,100,1000))
                  % numerical method
```

You may evaluate the model (calculate the y values) on provided x data or any list of x data by

```
y=evalmodel(a)  % on provided x data
y=evalmodel(a,-100:0.1:100)
                % on other x data
```

Note that because `getmodelprops` and `evalmodel` use the start values (see Appendix B), you need to accept fit result first if you wish to get the properties and model evaluations on fit result.

2. *Customized or combined curve models*

Now we demonstrate how to create a customized or combined curve model.

Still use the data from the last example. Suppose the customized curve model is

$$f(x) = A\exp\left(-|x-x_0|^3/b\right)$$

We need to first create the custom model (see Appendixe B for usage details) by

```
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

```
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

```
a.CustomCurveModel=linefit.creatmodel(@(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 setting `CurveModeIndex` to -2.

```
a.CurveModelIndex=-2;
```

We then add one more curve of the same model and start the fitting

```
a=addcurves(a,1);
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

```
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

```
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.

```
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

```
foo=load('demo_xtal.data');
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.,

```
@(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

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

There is no need to enter the method name `createmodel`. Then use the Select Model popupmenu 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.

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.

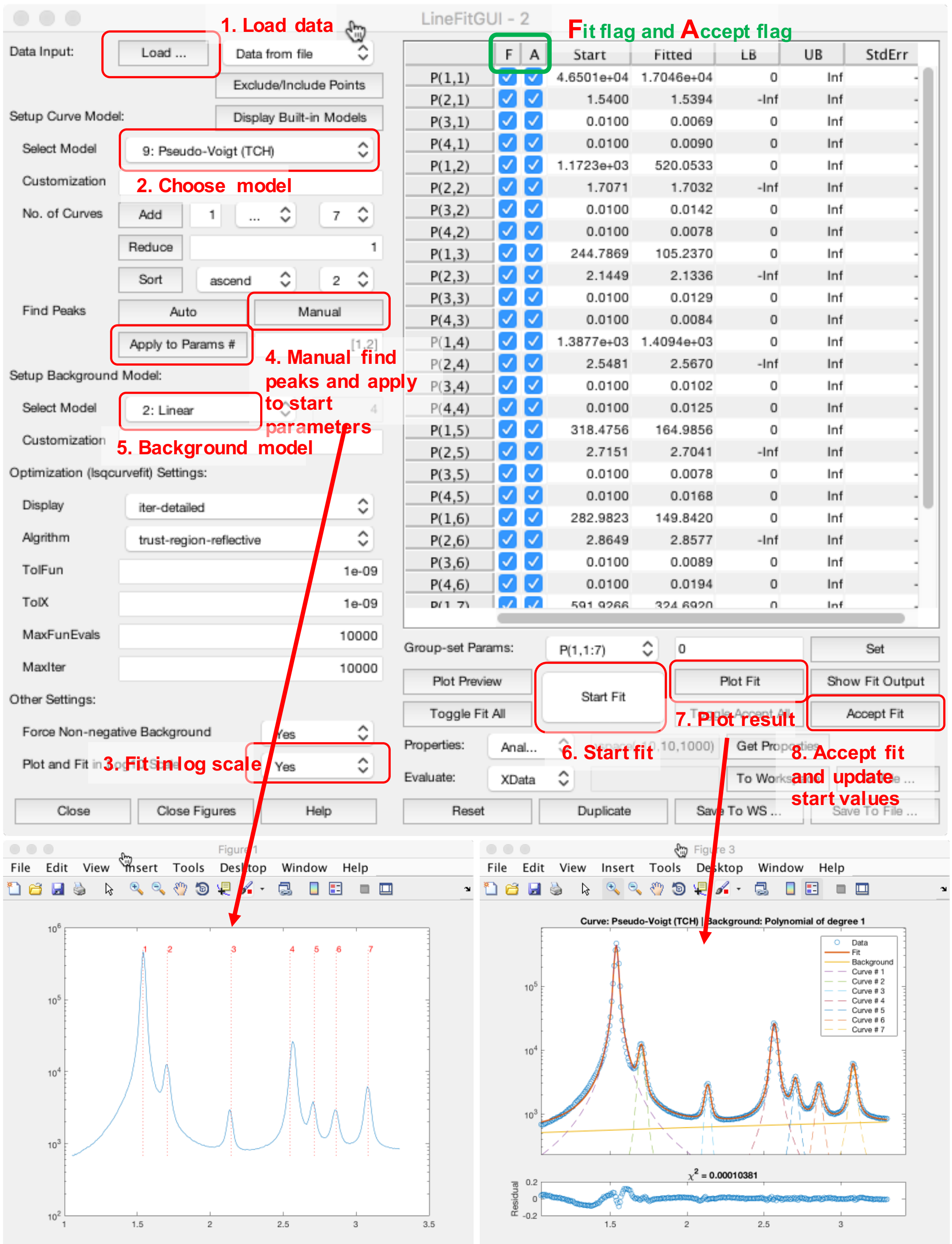

FIG. 1. Top: GUI; Bottom left: manually find peaks; Bottom right: plot after fitting.

- `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 the 1st way for dynamic methods; they can also be 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.

- `addcurves` (dynamic): Add curves of identical `ModelBase.CurveModel` to `Model.CurveModel`.

  `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 III 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=accetpfit(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

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

  `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[2] 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, kurtosis, and etc.

  `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 x values `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 with Matlab numerical integration function `trapz`,

  $$\mu = \int xf(x)dx;$$

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

  $$\mu_2 = \int (x-\mu)^2 f(x)dx;$$

  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

  `s=linefit.createmodel(fcn_handle,nofp, startparams,lowerbounds,upperbounds,fitflags)`. `fcn_handle` is the function handle for the model. `nofp` is the number of total parameters in the model. `startparams`, `lowerbounds`, and `upperbounds` are initial parameters, lower and upper bounds for the fitting. `fitflags` is logical flags (false/true or 0/1) to indicate whether the parameter is to be fixed or fitted. `startparams`, `lowerbounds`, `upperbounds`, and `fitflags` must be either vector of dimension `nofp`×1, or empty `[]` for default values (random values for `startparams`, -Inf, Inf and 1 for `lowerbounds`, `upperbounds`, and `fitflags`).

  `fcn_handle` must have the format `@(x,p)UserFcn(x,p)` where `x` is the input x value, `p` is a vector of parameters; or it can be an anonymous function, e.g., `f=@(x,p)p(1)+p(2)*x+p(3)*sin(x)` (which gives a value of 3.7296 when x=4 and p=[2,1,3], 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 `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, inital parameters, bounds and etc.

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

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=linefit.function_name(x,p)` or `[y,props]= linefit.function_name(x,p)`, where `p` is the parameter vector defined below, and `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).

$$f(x) = A\frac{1}{\pi\gamma}\left[1+\left(\frac{x-x_0}{\gamma}\right)^2\right]^{-1}$$

$$\begin{aligned} \text{mode} &: x_0 \\ \text{median} &: x_0 \\ \text{fwhm} &: 2\gamma \end{aligned}$$

2. `ilorentzian(x,[A,x0,gamma])` Intermediate Lorentzian distribution function[3]

   `A`: amplitude, `x0`: location, `gamma` ($\gamma > 0$): scale.

$$f(x) = A\frac{\sqrt{2^{\frac{2}{3}}-1}}{2\gamma}\left[1+(2^{\frac{2}{3}}-1)\left(\frac{x-x_0}{\gamma}\right)^2\right]^{-\frac{3}{2}}$$

$$\begin{aligned} \text{mode} &: x_0 \\ \text{median} &: x_0 \\ \text{fwhm} &: 2\gamma \end{aligned}$$

3. `mlorentzian(x,[A,x0,gamma])` Modified Lorentzian distribution function[4]

   `A`: amplitude, `x0`: location, `gamma` ($\gamma > 0$): scale.

$$f(x) = A\frac{2\sqrt{\sqrt{2}-1}}{\pi\gamma}\left[1+(\sqrt{2}-1)\left(\frac{x-x_0}{\gamma}\right)^2\right]^{-2}$$

$$\begin{aligned} \text{mode} &: x_0 \\ \text{median} &: x_0 \\ \text{fwhm} &: 2\gamma \end{aligned}$$

4. `gaussian(x,[A,x0,sigma])` Gaussian or normal distribution function

   `A`: amplitude, `x0`: location, `sigma` ($\sigma > 0$): scale or standard deviation.

$$f(x) = A\frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]$$

$$\begin{aligned} \text{mean} &: x_0 \\ \text{variance} &: \sigma^2 \\ \text{skewness} &: 0 \\ \text{kurtosis} &: 0 \\ \text{mode} &: x_0 \\ \text{median} &: x_0 \\ \text{fwhm} &: 2\sqrt{2\log(2)}\sigma \end{aligned}$$

5. `generalizednorm1(x,[A,a,b,c])` Generalized normal distribution function version 1[5]

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

$$f(x) = A\frac{c}{2b\Gamma(1/c)}\exp\left[-\left(\frac{|x-a|}{b}\right)^c\right]$$

$$\begin{aligned} \text{mean} &: a \\ \text{variance} &: \tfrac{b^2\Gamma(3/c)}{\Gamma(1/c)} \\ \text{skewness} &: 0 \\ \text{kurtosis} &: \tfrac{\Gamma(5/c)\Gamma(1/c)}{\Gamma(3/c)^2} - 3 \\ \text{mode} &: a \\ \text{median} &: a \end{aligned}$$

6. `laplace(x,[A,a,b])` Laplace distribution function[6]

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

$$f(x) = A\frac{1}{2b}\exp\left(-\frac{|x-a|}{b}\right)$$

$$\begin{aligned} \text{mean} &: a \\ \text{variance} &: 2b^2 \\ \text{skewness} &: 0 \\ \text{kurtosis} &: 3 \\ \text{mode} &: a \\ \text{median} &: a \end{aligned}$$

7. `logistic(x,[A,a,b])` Logistic distribution function[6]

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

$$f(x) = A\frac{1}{b}\exp\left(-\frac{x-a}{b}\right)\left[1+\exp\left(-\frac{x-a}{b}\right)\right]^{-2}$$

$$\begin{aligned} \text{mean} &: a \\ \text{variance} &: \pi^2 b^2/3 \\ \text{skewness} &: 0 \\ \text{kurtosis} &: 7/5 \\ \text{mode} &: a \\ \text{median} &: a \end{aligned}$$

8. `pearsonVII(x,[A,a,b,c])` Pearson type VII or student' t distribution function[6,7]

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

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

$$\begin{aligned} \text{mean} &: a \text{ for } c>1, \text{ otherwise undefined} \\ \text{variance} &: \tfrac{b^2 c}{c-2} \text{ for } c>2,\ \infty \text{ for } 1<c\le 2, \\ &\quad \text{otherwise undefined} \\ \text{skewness} &: 0 \text{ for } c>3, \text{ otherwise undefined} \\ \text{kurtosis} &: \tfrac{6}{c-4} \text{ for } c>4,\ \infty \text{ for } 2<c\le 4, \\ &\quad \text{otherwise undefined} \\ \text{mode} &: a \\ \text{median} &: a \end{aligned}$$

9. `pseudovoigtTCH(x,[A,x0,sigma,gamma])` Pseudo-Voigt type TCH distribution function[8]

   `A`: amplitude, `x0`: location, `sigma` ($\sigma>0$): scale (standard deviation) of the Gaussian for convolution, `gamma` ($\gamma>0$): scale (HWHM) of the Lorentzian for convolution.

$$f(x;\Gamma) = A\left[(1-\eta_L)f_G(x;\gamma_G)+\eta_L f_L(x;\sigma_L)\right],$$

   where Γ is the approximated FWHM of Voigt given by

$$\begin{aligned}\Gamma = (&\Gamma_{GV}^5 + 2.69269\Gamma_{GV}^4\Gamma_{LV} + 2.42843\Gamma_{GV}^3\Gamma_{LV}^2 \\ &+ 4.47163\Gamma_{GV}^2\Gamma_{LV}^3 + 0.07842\Gamma_{GV}\Gamma_{LV}^4 + \Gamma_{LV}^5)^{\frac{1}{5}}.\end{aligned}$$

   with $\Gamma_{GV} = 2\sqrt{2\log(2)}\sigma$ and $\Gamma_{LV} = 2\gamma$; $f_L(x)$ is a Lorentzian given by

$$f_L(x;\gamma_L) = \frac{1}{\pi\gamma_L}\left[1+\left(\frac{x-x_0}{\gamma_L}\right)^2\right]^{-1},$$

   with $\gamma_L = \Gamma/2$; $f_G(x)$ is a Gaussian given by

$$f_G(x) = \frac{1}{\sqrt{2\pi\sigma_G^2}}\exp\left[-\frac{(x-x_0)^2}{2\sigma_G^2}\right],$$

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

$$\eta_L = 1.36603\frac{\Gamma_{LV}}{\Gamma} - 0.47719\left(\frac{\Gamma_{LV}}{\Gamma}\right)^2 + 0.11116\left(\frac{\Gamma_{LV}}{\Gamma}\right)^3.$$

$$\text{fwhm} : \Gamma$$

10. `pseudovoigtIAT(x,[A,x0,sigma,gamma])` Pseudo-Voigt type IAT distribution function[9]

    `A`: amplitude, `x0`: location, `sigma` ($\sigma>0$): scale (standard deviation) of the Gaussian for convolution, `gamma` ($\gamma>0$): scale (HWHM) of the Lorentzian for convolution.

$$\begin{aligned} f(x) = A[&(1-\eta_L-\eta_I-\eta_P)f_G(x;\sigma_G) \\ &+\eta_L f_L(x;\gamma_L)+\eta_I f_I(x;\gamma_I)+\eta_P f_P(x;\gamma_P)],\end{aligned}$$

    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 γ_L, FWHM $\Gamma_L = 2\gamma_L$, and contribution η_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 $\Gamma_I = 2\sqrt{2^{2/3}-1}\gamma_I$ and contribution η_I; and $f_P(x;\gamma_P)$ is an inverse squared hyperbolic cosine function defined by

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

    with FWHM $\Gamma_P = 2\left[\log(\sqrt{2}+1)\right]\gamma_P$ and contribution η_P. These FWHMs are calculated via $\Gamma_i = (\Gamma_{GV}+\Gamma_{LV})w_i$, where $\Gamma_{GV} = 2\sqrt{2\log(2)}\sigma$, $\Gamma_{LV} = 2\gamma$, and $i \in \{G,L,I,P\}$. With ρ defined as $\rho \equiv \Gamma_{LV}/(\Gamma_{LV}+\Gamma_{GV})$, w_i and η_i are calculated via

$$\begin{aligned} w_G &= 1-\rho\sum_{i=0}^{N} a_i\rho^i, \\ w_L &= 1-(1-\rho)\sum_{i=0}^{N} b_i\rho^i, \\ w_I &= \sum_{i=0}^{N} c_i\rho^i, \\ w_P &= \sum_{i=0}^{N} d_i\rho^i, \\ \eta_L &= \rho\left[1+(1-\rho)\sum_{i=0}^{N} f_i\rho^i\right], \end{aligned}$$

$$\eta_I = \rho(1-\rho)\sum_{i=0}^{N} g_i\rho^i,$$

$$\eta_I = \rho(1-\rho)\sum_{i=0}^{N} h_i\rho^i,$$

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

i	a_i	b_i	c_i	d_i	f_i	g_i	h_i
0	0.66	-0.42179	1.19913	1.10186	-0.30165	0.25437	1.01579
1	0.15021	-1.25693	1.43021	-0.47745	-1.38927	-0.14107	1.50429
2	-1.24984	10.30003	-15.36331	-0.68688	9.3155	3.23653	-9.21815
3	4.74052	-23.45651	47.06071	2.76622	-24.10743	-11.09215	23.59717
4	-9.48291	29.14158	-73.61822	-4.55466	34.96491	22.10544	-39.71134
5	8.48252	-16.50453	57.92559	4.05475	-21.18862	-24.12407	32.83023
6	-2.95553	3.19974	-17.80614	-1.26571	3.7029	9.76947	-10.02142

fwhm : Γ (same as `pseudovoigtTCH`)

11. `pseudovoigtLLHGD(x,[A,x0,sigma,gamma])` Pseudo-Voigt type LLHGD distribution function[10]

    `A`: amplitude, `x0`: location, `sigma` ($\sigma > 0$): scale (standard deviation) of the Gaussian for convolution, `gamma` ($\gamma > 0$): scale (HWHM) of the Lorentzian for convolution.

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

where Γ 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 : Γ

12. `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. `voigt(x,[A,x0,sigma,gamma])` Voigt distribution function[1,13]

    `A`: amplitude, `x0`: location, `sigma` ($\sigma > 0$): scale (standard deviation) of the Gaussian for convolution, `gamma` ($\gamma > 0$): scale (HWHM) of the Lorentzian for convolution.

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

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

$$\begin{aligned} w[z] &= \exp(-z^2)\mathrm{erfc}(-iz) = \mathrm{erfcx}(-iz) \\ &= \exp(-z^2)\left[1 + \frac{2i}{\sqrt{\pi}}\int_0^z \exp(t^2)dt\right] \end{aligned}$$

fwhm : Γ (same as `pseudovoigtTCH`)

14. `skewlaplace(x,[A,a,b,c])` Skew-Laplace distribution function[6,14]

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

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

mean : $a - b + c$

variance : $b^2 + c^2$

$$\text{skewness}: \tfrac{2(c^3-b^3)}{(c^2+b^2)^{(3/2)}}$$
$$\text{kurtosis}: \tfrac{9b^4+6b^2c^2+9c^4}{(c^2+b^2)^2}-3$$
$$\text{mode}: a$$
$$\text{median}: a+b\log\left(\tfrac{b+c}{2b}\right)$$

15. `skewlogistic(x,[A,a,b,c])` Skew-logistic or generalized logistic type I distribution function[6,15]

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

$$f(x)=A\frac{c}{b}\exp\left(-\frac{x-a}{b}\right)\left[1+\exp\left(-\frac{x-a}{b}\right)\right]^{-(c+1)}$$

$$\text{mean}: a+[\gamma+\psi(c)]b$$
$$\text{variance}: \left[\tfrac{\pi^2}{6}+\psi^{(1)}(c)\right]b^2$$
$$\text{skewness}: \tfrac{\psi^{(2)}(c)-\psi^{(2)}(1)}{\left[\tfrac{\pi^2}{6}+\psi^{(1)}(c)\right]^{3/2}}$$
$$\text{kurtosis}: \tfrac{\psi^{(3)}(c)+\psi^{(1)}(c)\left[\pi^2+3\psi^{(1)}(c)\right]+\tfrac{3\pi^4}{20}}{\left[\tfrac{\pi^2}{6}+\psi^{(1)}(c)\right]^2}-3$$
$$\text{mode}: a+b\log(c)$$
$$\text{median}: a-b\log(2^{1/c}-1)$$

    where γ is the Euler-Mascheroni constant, $\psi(x)$ is the digamma function, and $\psi^{(i)}(x)$ is the polygamma function.

16. `skewnorm(x,[A,a,b,c])` Skew-normal distribution function[6,16]

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

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

$$\text{mean}: a+b\delta\sqrt{\tfrac{2}{\pi}}$$
$$\text{variance}: b^2\left(1-\tfrac{2\delta^2}{\pi}\right)$$
$$\text{median}: \tfrac{4-\pi}{2}\tfrac{(\delta\sqrt{2/\pi})^3}{(1-2\delta^2/\pi)^{3/2}}$$
$$\text{kurtosis}: 2(\pi-3)\tfrac{(\delta\sqrt{2/\pi})^4}{(1-2\delta^2/\pi)^2}$$

    where $\delta=\frac{c}{\sqrt{1+c^2}}$.

17. `skewpvSB(x,[A,x0,fwhm0,eta,a])` Asymmetric pseudo-Voigt type SB distribution function[17]

    `A`: amplitude, `x0`: location, `fwhm0` ($\Gamma_0>0$): FWHM of symmetric Voigt, `eta` ($\eta\geq 0$): contribution of Lorentzian, `a`: degree of asymmetry.

$$f(x)=(1-\eta)f_G(x;\sigma_G)+\eta f_L(x;\gamma_L),$$

    where $f_G(x;\sigma_G)$ is a Gaussian with $\sigma_G=\Gamma_M/(2\sqrt{2\log(2)})$ and $f_L(x;\gamma_L)$ is a Lorentzian with $\gamma_L=\Gamma_M/2$, where Γ_M is the modified FWHM written as

$$\Gamma_M=\frac{2\Gamma_0}{1+\exp\left[-a(x-x_0)\right]}.$$

    Note that Γ_M is not FWHM of the distribution function; 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. `skewpvSSG(x,[A,x0,fwhm0,eta,a,b])` Asymmetric pseudo-Voigt type SSG distribution function[18]

    `A`: amplitude, `x0`: location, `fwhm0` ($\Gamma_0>0$): FWHM of symmetric Voigt, `eta` ($\eta\geq 0$): 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

$$\Gamma_M=\frac{2\Gamma_0}{1+\exp\left[-a(x-x_0-b)\right]}.$$

    Note that Γ_M is not FWHM of the distribution function; 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.

19. `burrXII(x,[A,x0,a,b,c])` Burr type VII distribution function[6]

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

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

$$\text{mean}: x_0+\mu_1$$
$$\text{variance}: -\mu_1^2+\mu_2$$
$$\text{skewness}: \tfrac{2\mu_1^3-3\mu_1\mu_2+\mu_3}{(-\mu_1^2+\mu_2)^{3/2}}$$
$$\text{kurtosis}: \tfrac{-3\mu_1^4+6\mu_1^2\mu_2-4\mu_1\mu_3+\mu_4}{(-\mu_1^2+\mu_2)^2}-3$$
$$\text{mode}: x_0+a\left(\tfrac{b-1}{bc+1}\right)^{1/b}$$
$$\text{median}: x_0+a(2^{1/c}-1)^{1/b}$$

    where $\mu_r=a^rc\mathrm{B}\left(\frac{bc-r}{b},\frac{b+r}{b}\right)$ are moments and $\mathrm{B}(x,y)=\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$ is the beta function (a.k.a. the Euler integral of the first kind).

20. `expdist(x,[A,a,b])` Exponential distribution function[6]

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

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

$$\begin{aligned} \text{mean} &: a + b \\ \text{variance} &: b^2 \\ \text{skewness} &: 2 \\ \text{kurtosis} &: 6 \\ \text{mode} &: a \\ \text{median} &: a + b\log(2) \end{aligned}$$

21. `gammadist(x,[A,a,b,c])` Gamma distribution function[6]

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

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

$$\begin{aligned} \text{mean} &: a + bc \\ \text{variance} &: b^2 c \\ \text{skewness} &: 2/\sqrt{c} \\ \text{kurtosis} &: 6/c \\ \text{mode} &: a + b(c-1) \text{ for } c \geq 1, \\ & \quad \text{otherwise } a \end{aligned}$$

22. `inversegamma(x,[A,x0,a,b])` Inverse gamma distribution function[6,19]

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

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

$$\begin{aligned} \text{mean} &: x_0 + \frac{a}{b-1} \text{ for } b > 1 \\ \text{variance} &: \frac{a^2}{(b-2)(b-1)^2} \text{ for } b > 2 \\ \text{skewness} &: \frac{4\sqrt{b-2}}{b-3} \text{ for } b > 3 \\ \text{kurtosis} &: \frac{30b-66}{(b-4)(b-3)} \text{ for } b > 4 \\ \text{mode} &: x_0 + \frac{a}{b+1} \end{aligned}$$

23. `inversenorm(x,[A,x0,a,b])` Inverse normal distribution (a.k.a Wald distribution) function[6,20]

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

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

$$\begin{aligned} \text{mean} &: x_0 + a \\ \text{variance} &: a^3/b \\ \text{skewness} &: 3\sqrt{a/b} \\ \text{kurtosis} &: 15a/b \\ \text{mode} &: x_0 + \frac{a}{2b}\left(\sqrt{9a^2+4b^2} - 3a\right) \end{aligned}$$

24. `levy(x,[a,b])` Lévy distribution function[6]

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

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

$$\begin{aligned} \text{mean} &: \infty \\ \text{variance} &: \infty \\ \text{mode} &: a + b/3 \\ \text{median} &: a + (b/2)/\left[\text{erfc}^{-1}(1/2)\right]^2 \end{aligned}$$

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

25. `logcauchy(x,[A,x0,a,b])` Log-Cauchy distribution function[21]

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

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

$$\begin{aligned} \text{variance} &: \infty \\ \text{median} &: x_0 + \exp(a) \end{aligned}$$

26. `lognorm(x,[A,x0,a,b])` Log-normal distribution[6,22]

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

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

$$\begin{aligned} \text{mean} &: x_0 + \exp(a + b^2/2) \\ \text{variance} &: \left[\exp(b^2) - 1\right]\exp(2a + b^2) \\ \text{skewness} &: \left[\exp(b^2) + 2\right]\sqrt{\exp(b^2) - 1} \\ \text{kurtosis} &: \exp(4b^2) + 2\exp(3b^2) + 3\exp(2b^2) - 6 \\ \text{mode} &: x_0 + \exp(a - b^2) \\ \text{median} &: x_0 + \exp(a) \end{aligned}$$

27. `paretoI(x,[A,a,b])` Pareto type I distribution function[6,23]

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

$$f(x) = A \begin{cases} 0, & \text{for } x < a \\ \frac{ba^b}{x^{b+1}}, & \text{for } x \geq a \end{cases}$$

$$\begin{aligned} \text{mean} &: \tfrac{ab}{b-1} \text{ for } b > 1, \text{ otherwise } \infty \\ \text{variance} &: \tfrac{a^2 b}{(b-1)^2(b-2)} \text{ for } b > 2, \infty \text{ for } b \in (0, 2] \\ \text{skewness} &: \tfrac{2(1+b)}{b-3}\sqrt{\tfrac{b-2}{b}} \text{ for } b > 3 \\ \text{kurtosis} &: \tfrac{6(b^3+b^2-6b-2)}{b(b-3)(b-4)} \text{ for } b > 4 \\ \text{mode} &: a \\ \text{median} &: a\sqrt[b]{2} \end{aligned}$$

28. `weibull(x,[A,a,b,c])` Weibull distribution function[6,24]

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

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

$$\begin{aligned} \text{mean} &: a + bG(1) \\ \text{variance} &: \left[-G^2(1) + G(2)\right] b^2 \\ \text{skewness} &: \tfrac{2G^3(1) - 3G(1)G(2) + G(3)}{[-G^2(1)+G(2)]^{3/2}} \\ \text{kurtosis} &: \tfrac{-3G^4(1) + 6G^2(1)G(2) - 4G(1)G(3) + G(4)}{[-G^2(1)+G(2)]^2} \\ \text{mode} &: a \text{ for } c \leq 1,\ a + b\sqrt[c]{\frac{c-1}{c}} \text{ for } c > 1 \\ \text{median} &: a + b\sqrt[c]{2} \end{aligned}$$

    where $G(n) = \Gamma\left(\frac{c+n}{n}\right)$.

29. `atan(x,[A,a,b])` Inverse tangent function

    `A`: amplitude, `a` : location, `b`: scale.

$$f(x) = A\text{atan}\left(\frac{x-a}{b}\right)$$

30. `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. `powerlaw(x,[A,a,b])` Power-law function

    `A`: amplitude or scale, `a` : location, `b`: shape.

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

---

[1] Faddeeva is a package to compute complex error functions. See README file for compiling instructions for your platform. The package can be downloaded from `https://www.mathworks.com/matlabcentral/fileexchange/38787-faddeeva-package--complex-error-functions`.

[2] (), matlab Central File ID: 25500. `https://www.mathworks.com/matlabcentral/fileexchange/`.

[3] C. Khattak and D. Cox, J. App. Cryst. **10**, 405 (1977).

[4] G. Malmros and J. O. Thomas, J. Appl. Cryst. **10**, 7 (1977).

[5] `https://en.wikipedia.org/wiki/Generalized_normal_distribution`.

[6] M. McLaughlin, *Compendium of Common Probability Distributions*, 2nd ed. (2014) `http://www.causascientia.org/math_stat/Dists/Compendium.pdf`.

[7] `https://en.wikipedia.org/wiki/Student's_t-distribution` ().

[8] P. Thompson, D. E. Cox, and J. B. Hastings, J. Appl. Cryst. **20**, 79 (1987).

[9] T. Ida, M. Ando, and H. Toraya, J. Appl. Cryst. **33**, 1311 (2000).

[10] Y. Liu, J. Lin, G. Huang, Y. Guo, and C. Duan, J. Opt. Soc. Am. B **18**, 666 (2001).

[11] J. J. Olivero and R. L. Longbothum, J. Quant. Spectrosc. Radiat. Transfer **17**, 233 (1977).

[12] `https://en.wikipedia.org/wiki/Uniform_distribution_(continuous)`.

[13] G. P. M. Poppe and C. M. J. Wijers, ACM Trans. Math. Soft. **16**, 38 (1990).

[14] `https://en.wikipedia.org/wiki/Asymmetric_Laplace_distribution` ().

[15] `https://en.wikipedia.org/wiki/Generalized_logistic_distribution` ().

[16] `https://en.wikipedia.org/wiki/Skew_normal_distribution` ().

[17] A. Stancik and E. Brauns, Vib. Spectr. **47**, 66 (2008).

[18] M. Schmid, H.-P. Steinrück, and J. M. Gottfried, Surf. Interface Anal. **46**, 505 (2014).

[19] `https://en.wikipedia.org/wiki/Inverse-gamma_distribution` ().

[20] `https://en.wikipedia.org/wiki/Inverse_Gaussian_distribution` ().

[21] `https://en.wikipedia.org/wiki/Log-Cauchy_distribution` ().

[22] `https://en.wikipedia.org/wiki/Log-normal_distribution` ().

[23] `https://en.wikipedia.org/wiki/Pareto_distribution`.

[24] `https://en.wikipedia.org/wiki/Weibull_distribution`.