Demonstrate the use of functions in polyfitter

Synopsis
polyfitterdemo()
include "polyfitter.g"

Description
Demonstrate the use of the functions in polyfitter. Inputs and results are sent to the logger.

See Also
polyfitter

Returns
Bool: T for success

1.5.23 polyfittertest - Function

Package utility
Module mathematics

Test the polyfitter class

Synopsis
polyfittertest()
include "polyfitter.g"

Description
This class purports to test the polyfitter class. In fact, all it does currently is run polyfitterdemo.

See Also
polyfitter polyfitterdemo

Returns
Bool: T for success

1.5.24 sinusoidfitter - Tool

Package **utility**
Module **mathematics**

Non-linear least-squares fit sinusoid to data
include "sinusoidfitter.g"

Constructors

Functions

eval	Evaluate a sinusoid (usually the result of a fit)
fit	Fit a sinusoid to data
setstate	Set the state of the fitter

Description

`sinusoidfitter` is a class that does a non-linear least-squares fit of a sinusoid of a specified order to data. An optional estimate of the errors in the data may be provided. The state of the fitter, which includes the parameters of the sinusoid, estimated errors, the covariance matrix, the number of iterations actually done, and whether the fit converged or not, is available at any time and is set whenever the fit function is invoked.. A function to evaluate the fitted sinusoid is also provided. The controlling values of the fitter (initial guesses for the parameters, the maximum number of iterations, and the convergence criteria) can be set by the user.
At present, only fits to real data without constraints are supported.
The fits are carried out with double precision.
The form of the sinusoid is: $A\cos\left(2\pi(x - x_0)/p\right)$
It is best to remove at a mean value from the data before the fit since the sinusoid has zero mean.
The fit is not particularly robust against aliasing so care should be taken to inspect the result at all times. The fitter does best when the initial period guess is longer than the suspected best fit.
The computation is actually performed in a C++ executable, not in Glish. Details about the algorithms are available in the documentation of the underlying C++ fitting classes.
`sinusoidfitterdemo()` and `sinusoidfittertest()` functions are available.

Example

First we need to get access to the sinusoid fitting declarations and create a fitting tool:

```
include "sinusoidfitter.g"
fitter := sinusoidfitter()
```

Now lets manufacture some data from the sinusoid

$$y = 3\cos 2\pi(x-8)/10$$

```
x := [0:99]/5.0  # 0,0.2,0.4,0.6,0.8,1.0,1.2,...,19.8
y := 3*cos(2*pi*(x-8)/10)
```

The fitter needs a reasonable initial guess (this is obviously contrived, since we know what the sinusoid is).

```
guess := [=];
guess.amplitude := 1.0
guess.period := 13.0
guess.x0 := 0.0
fitter.setstate(guess)
```

Now, lets perform the fit:

```
ok := fitter.fit(state, x, y)
# and examination of the state record shows that the fit
# has not yet converged.  However, the current values of
# the sinusoid parameters appear to be converging.
# So we attempt the fit again.  Since we have not provided
# any new initial guess via the setstate member, the fit
# will use the most recent values as its starting point.
ok := fitter.fit(state, x, y)
# the fit has now converged according to the state record.
```

The state argument is a record which will contain the state of the fitter at the end of the fit (either maxiter will have been reached or the fit will have converged). The other two arguments, `x` and `y`, are both input arguments and are not changed by using this function. We did not provide an error estimate σ, so the software assumes the errors are identical for each data point and $\sigma = 1$. The state record will indicate if the fit converged or not. The arguments of the fit function are:

ok The return parameter. It is false only when some severe error has occurred. If a fit has merely failed to converge, the state record will show that and the return value will still be true.

state The state of the fitter after the fit. See the description of the fit function for a full listing of all of the fields in the state record.

x,y The data - x and y must be vectors having the same number of elements.

sigma An optional argument indicating the errors in the data values, y.
When absent, it is assumed that all y values have the same error.

The other function in this class is one to `eval`uate a sinusoid. It is the sinusoid given by the current state of the fitter. Usually this will be the sinusoid you have just fit, but it could be any sinusoid you specify. For example, to find out the largest deviation of the fit values from the data values we could do the following:

```
ok := fitter.eval(y2, x)
print max(abs(y2 - y))
```

ok The return value is True if the fit succeeds, False otherwise. (Output)

y2 The sinusoid evaluated at `x`. (Output)

x Where to evaluate the sinusoid. (Input)

See Also
sinusoidfitterdemo sinusoidfittertest

sinusoidfitter.fit - Function

Package	**utility**
Module	**mathematics**
Tool	**sinusoidfitter**

Fit a sinusoid to data

Synopsis
fit(state, x, y, sigma)

Arguments

state	The state of the fitter after the fit. (Output)
x	(Input)
y	Must be the same length as "x". (Input)
sigma	Errors in the "y" values. (Input) Default: 1.0

Returns
Bool — did the function call succeed

Description
Fit a sinusoid to data. If sigma does not have the same number of elements as x and y (which is the case with the default value for this argument) then the fitter assumes that all y values have the same sigma. The state record has the following fields:

converged A boolean. When true, the fit has converged.

curriter The number of iterations actually used in the most recent fit.

maxiter The maximum number of iterations to perform during any invocation of the fit function. The default value is 10

criteria The convergence criteria. If the change in the value of χ^2 between two subsequent iteration is less than this value, then the fit is considered to have converged. The default value is 0.001

chisq The χ^2 for this fit.

amplitude The amplitude of the sinusoid.

period The `x` distance between peaks of the sinusoid.

x0 The `x` location of one peak of the sinusoid.

amplitude_error The 1-sigma error in the amplitude, from the covariance matrix.

period_error The 1-sigma error in the period, from the covariance matrix.

x0_error The 1-sigma error in x0, from the covariance matrix.

covariance The covariance matrix, the diagonal is used to determine the errors in the parameters.

---

sinusoidfitter.eval - Function

Package	**utility**
Module	**mathematics**
Tool	**sinusoidfitter**

Evaluate a sinusoid (usually the result of a fit)

Synopsis
eval(y, x)

Arguments

y	Values of the evaluated sinusoid. (Output)
x	The independent variable. (Input)

Returns
Bool — did the evaluation succeed

Description
Evaluate the sinusoid indicated by the current state of the fitter. Normally this is the most recent result of a use of fit, however you may use the setstate function to evaluate any sinusoid.

---

sinusoidfitter.setstate - Function

Package	**utility**
Module	**mathematics**
Tool	**sinusoidfitter**

Set the state of the fitter

Synopsis
setstate(state)

Arguments

state	The state record indicated values in the fitter to change (Input).

Returns
Bool — did the setstate succeed

Description
Change the state of the fitter. Use this to indicate the initial sinusoid parameters and the convergence criteria and maximum number of iteration. See the description for the fit function for a full list of the fields in the state record. Only the maxiter, criteria, amplitude, period, and x0 fields may be used to set the state of the fitter. Any other fields are ignored (and the fact that they have been ignored is logged).

1.5.25 sinusoidfitterdemo - Function

Package utility
Module mathematics

Demonstrate the use of functions in sinusoidfitter

Synopsis
sinusoidfitterdemo()
include "sinusoidfitter.g"

Description
Demonstrate the use of the functions in sinusoidfitter. Inputs and results are sent to the logger.

See Also
sinusoidfitter

Returns
Bool: T for success

1.5.26 sinusoidfittertest - Function

Package utility
Module mathematics

Test the sinusoidfitter class

Synopsis
sinusoidfittertest()
include "sinusoidfitter.g"

Description
This class purports to test the sinusoidfitter class. In fact, all it does currently is run sinusoidfitterdemo.

See Also
sinusoidfitter sinusoidfitterdemo

Returns
Bool: T for success

1.5.27 gaussian1d - Function

Package utility
Module mathematics

Calculate a one-dimensional Gaussian

Synopsis
gaussian1d(x, height, center, fwhm)
include "gaussian.g"

Description

The gaussian1d function evaluates one-dimensional Gaussians. You can specify the peak height of the Gaussian, the center of the Gaussian and the width of the Gaussian.

The width of the Gaussian is defined to be the full width at half-maximum (FWHM). If you prefer to use "natural" widths, a constant `fwhm_to_natural` is provided[3]. To get a natural width, multiply the FWHM by this constant [4]. Similarly the FWHM is derived by dividing the natural width by this constant.

You can evaluate the Gaussian at a number of positions by passing the function a vector for "x" instead of a scalar.

Arguments

---

[3]The natural width is σ in the expression $\exp(\frac{-(x-x_0)^2}{\sigma^2})$

[4] $\frac{1}{\sqrt{\ln(16)}}$

x	in	values at which to evaluate the Gaussian Allowed:double Default: no default
height	in	maximum value of the Gaussian Allowed:double Default: 1.0
center	in	location of the maximum value of the Gaussian Allowed:double Default: 0.0
fwhm	in	width of the gaussian Allowed:double Default: 1/fwhm_to_natural

Returns
The "y" values or fail

Example
This simple example calculates and plots a Gaussian.

```
include 'gaussian.g';
x := 1:100;
y := gaussian1d(x, height=3, center=30, fwhm=20);
```

And then plot the example, marking the peak and FWHM of the Gaussian.

```
include 'pgplotter.g';
pg := pgplotter();
pg.env(1, 100, 0, 3, 0, 0);
pg.line(x,y);                    # Plot the Gaussian
pg.line([30,30], [0, 3]);        # Show the center

# Show the FWHM
pg.line([30-20/2, 30-20/2, 30+20/2, 30+20/2],   # x
[0,       3/2,     3/2,     0]);        # y
```

See Also
gaussian2d

1.5.28 gaussian2d - Function

Package utility
Module mathematics

Calculate two-dimensional Gaussian

Synopsis
gaussian2d(x, y, height, center, fwhm, pa)

include "gaussian.g"

Description

The gaussian2d function evaluates two-dimensional Gaussians. You can specify the peak height of the Gaussian, the center of the Gaussian, the width of the major and minor axes and the position-angle of the Gaussian.

The width of the Gaussian is defined to be the full width at half-maximum (FWHM). If you prefer to use "natural" widths a constant `fwhm_to_natural` is provided[5]. To get a natural width, multiply the FWHM by this constant [6]. Similarly the FWHM is derived by dividing the natural width by this constant.

The "position angle" is the angle, measured counter-clockwise, between the "x" axis and the major axis of the Gaussian. This angle is in radians.

The major axis of the Gaussian is considered to be the first axis of the specified FWHM, whether or not that width is in fact wider than the second axis. If you care about this, you should be careful to ensure that the first width is not smaller than the second.

You can evaluate the Gaussian at a number of positions by passing the function a vector for "x" and "y" instead of just scalars.

Arguments

[5] The natural width is σ in the expression $\exp(\frac{-(x-x_0)^2}{\sigma^2})$

[6] $\frac{1}{\sqrt{\ln(16)}}$