Package	**utility**
Module	**mathematics**
Tool	**gauss1dfitter**

Set the convergence criteria

Synopsis
setcriteria(criteria)

Arguments

criteria	New criteria Allowed:double

Returns
boolean

1.5.21 polyfitter - Tool

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

Least-squares fit polynomials to data
include "polyfitter.g"

Constructors

Functions

eval	Evaluate a polynomial (usually the result of a fit)
fit	Fit a polynomial of a specified order to data
multifit	Fit several sets of data to polynomials

Description

`polyfitter` is a class that does a linear least-squares fit of a polynomial of a

specified order to data. An optional estimate of the errors in the data may be provided. Besides the coefficients, χ^2 and the estimated errors in the coefficient are output. A function to evaluate the fit polynomial is also provided.
At present, only fits to real data without constraints are supported.
The fits are carried out with double precision.
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.
`polyfitterdemo()` and `polyfittertest()` functions are available.

Example

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

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

Now lets manufacture some data from the polynomial: $y = 3x^2 + 5$

```
x := 1:10          # 1,2,3,4,5,6,7,8,9,10
y := 3*x*x + 5     # 8, ..., 305
```

Now, lets perform the fit:

```
ok := fitter.fit(coeff, coefferrs, chisq, x, y, order=2)
```

All the parameters up to (but not including) `x` are output parameters, the others are all input errors. We did not provide an error estimate σ, so the software assumes the errors are identical for each data point and $\sigma = 1$. The variables have the following meaning:

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

coeff A vector containing the estimated (fit) coefficients. Coefficient zero (*i.e.*, the constant term) is first, the highest order coefficient is last.

coefferrs The estimated errors in the fit coefficients.

chisq The χ^2 for this fit.

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

order What order polynomial do we want to fit? We specify it this way because we have "skipped over" the defaulted `sigma` argument. You can spell out all arguments this way if you like.

One can also fit several independent y vectors with a single call using the multifit function.
The other function in this class is one to `evaluate` a polynomial. Usually it will be the polynomial or polynomialsyou have just fit, but it could be any polynomial 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, coeff)
print max(abs(y2 - y))
```

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

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

x Where to evaluate the polynomial. (Input)

coeff A vector containing the coefficients to use. Coefficient zero (*i.e.*, the constant term) is first, the highest order coefficient is last. (Input)

The coeff parameter in the eval function could be the result of a call to fit or multifit. If coeff is a vector, it will return a single polynomial evaluated at x. If coeff is a matrix (two dimensions), it will return N polynomials evaluated at x, where N is the number of elements along the second dimension of coeff.

See Also
polyfitterdemo polyfittertest

polyfitter.fit - Function

Package	**utility**
Module	**mathematics**
Tool	**polyfitter**

Fit a polynomial of a specified order to data

Synopsis
fit(coeff, coefferrs, chisq, x, y, sigma, order)

Arguments

coeff	The coefficients of the fit. The constant (0th-order) term is first, highest order term last. (Output)
coefferrs	Estimated errors of the fit. (Output)
chisq	χ^2 for the fit. (Output)
x	(Input)
y	Must be the same length as "x". (Input)
sigma	Errors in the "y" values. (Input) Default: 1.0
order	Order of the highest term in the polynomial to fit. (Input) Allowed:$0 \leq order \leq 10$ Default: 1

Returns
Bool — did the fit succeed

Description
Fit a polynomial to data.

---

polyfitter.multifit - Function

Package	**utility**
Module	**mathematics**
Tool	**polyfitter**

Fit several sets of data to polynomials

Synopsis
multifit(coeff, coefferrs, chisq, x, y, sigma, order)

Arguments

coeff	The coefficients of the fit. The constant (0th-order) term is first, highest order term last. coeff[,1] are the coefficients for the first fit, coeff[,2] for the second, and so on. (Output)
coefferrs	Estimated errors of the fit. coefferrs[,i] are the errors for the ith fit. (Output)
chisq	χ^2 for the fit. chisq[i] is the chi-square for the i'th fit. (Output)
x	One dimensional array. (Input)
y	Must be the same length as "x" in its first axis, the length of the second axis is the number of fits to do. (Input)
sigma	Errors in the "y" values. If one dimensional, the sigmas are assumed to be the same for all fits. (Input) Default: 1.0
order	Order of the highest term in the polynomial to fit. (Input) Allowed:$0 \leq order \leq 10$ Default: 1

Returns
Bool — did the fit succeed

Description
Fits a polynomial to multiple data sets (which must share the same x however)[2]. Basically, this is the same as `fit` (without an error message - errors are in the log messages), except the dimensionality of all arguments is one higher. This function runs in C++ at the same speed as `fit`, however it minimizes the communication over the Glish bus, which would limit you to 100 fits or less per second.

Example

```
  include "polyfitter.g"
T
  x := 1:100
  y := array(0, 100, 3)
  y[,1] := 5                    # constant
  y[,2] := 2*x                  # linear
  y[,3] := 10*x*x               # square
  fitter := polyfitter()
```

[2]This restriction could be relaxed. Let us know if it would be useful to do so.

```
  fitter.multifit(coeff, coefferrs, chisq, x, y, order=2)
1
  coeff[,1]                     # the first fit
[5 -2.98023e-08 0]
  coeff[,2]                     # the second fit
[0.00126371 2 3.93629e-08]
  coeff[,3]                     # the third fit
[0.00238887 -0.000247818 10]
```

---

polyfitter.eval - Function

Package	**utility**
Module	**mathematics**
Tool	**polyfitter**

Evaluate a polynomial (usually the result of a fit)

Synopsis
eval(y, x, coeff)

Arguments

y	Values of the evaluated polynomial. If coeff has two dimensions, y[,1] is the evaluation using coeff[,1], y[,2] corresponds to coeff[,2] and so on.(Output)
x	The independent variable. (Input)
coeff	The coefficients in the order as emitted by fit or multifit.(Input)

Returns
Bool — did the evaluation succeed

Description
Evaluate a polynomial. Normally you will be evaluating the result of a fit, but you can evaluate a polynomial from any other source as well.

1.5.22 polyfitterdemo - Function

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