curvefit

1.1.0

Generated by Doxygen 1.8.11

Contents

1 Main Page

1.1 Introduction

CURVEFIT is a library for fitting functions to sets of data.

Author

Jason Christopherson

Version

1.1.0

2 Modules Index

2.1 Modules List

Here is a list of all documented modules with brief descriptions:

3 Data Type Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

4 Data Type Index

4.1 Data Types List

Here are the data types with brief descriptions:

5 Module Documentation

5.1 curvefit_c_binding Module Reference

curvefit_c_binding

Data Types

- type c_linear_interp

  A C compatible type encapsulating a linear_interp object.
- type c_lowess_smoothing

  A C compatible type encapsulating a lowess_smoothing object.
- type c_nonlinear_regression

  A C compatible type encapsulating a nonlinear_regression object.
- type c_polynomial_interp

  A C compatible type encapsulating a polynomial_interp object.
- type c_spline_interp

  A C compatible type encapsulating a spline_interp object.
- type cnonlin_reg_helper

  A type for helping to interface between a C function pointer, and the nonlinear_regression type.
- interface creg_fcn

  Describes a routine for finding the coefficients of a function of one variable.

Functions/Subroutines

- pure logical(c_bool) function is_monotonic_c (n, x)

  Tests to see if an array is montonically increasing or decreasing.
- subroutine get_linear_interp (obj, li)

  Retrieves the linear_interp object from the C compatible c_linear_interp data structure.
- subroutine lininterp_init_c (obj, n, x, y, err)

  Initializes a new c_linear_interp object.
- subroutine lininterp_free_c (obj)

  Frees resources held by a c_linear_interp object.
- subroutine lininterp_interp_c (obj, n, x, y)

  Performs a linear interpolation to determine the points y that for the requested indendent variable values in x.
- integer(i32) function lininterp_get_pt_count_c (obj)

  Gets the number of points used by the interpolation object.
- subroutine lininterp_get_pts_c (obj, n, x, y)

  Gets a copy of the data points stored by the interpolation object.
- subroutine get_polynomial_interp (obj, interp)

  Retrieves the polynomial_interp object from the C compatible c_polynomial_interp data structure.
- subroutine polyinterp_init_c (obj, n, x, y, order, err)

  Initializes a new c_polynomial_interp object.
- subroutine polyinterp_free_c (obj)

  Frees resources held by a c_polynomial_interp object.
- subroutine polyinterp_interp_c (obj, n, x, y)

  Performs a polynomial interpolation to determine the points y that for the requested indendent variable values in x.
- integer(i32) function polyinterp_get_pt_count_c (obj)

  Gets the number of points used by the interpolation object.
- subroutine polyinterp_get_pts_c (obj, n, x, y)

  Gets a copy of the data points stored by the interpolation object.
- subroutine get_spline_interp (obj, interp)

  Retrieves the spline_interp object from the C compatible c_spline_interp data structure.
- subroutine splineinterp_init_c (obj, n, x, y, ibcbeg, ybcbeg, ibcend, ybcend, err)

  Initializes a new c_spline_interp object.
- subroutine splineinterp_free_c (obj)

  Frees resources held by a c_spline_interp object.
- subroutine splineinterp_interp_c (obj, n, x, y)

  Performs a spline interpolation to determine the points y that for the requested indendent variable values in x.
- integer(i32) function splineinterp_get_pt_count_c (obj)

  Gets the number of points used by the interpolation object.
- subroutine splineinterp_get_pts_c (obj, n, x, y)

  Gets a copy of the data points stored by the interpolation object.
- subroutine splineinterp_diff1_c (obj, n, x, y)

  Computes the interpolated first derivative.
- subroutine splineinterp_diff2_c (obj, n, x, y)

  Computes the interpolated second derivative.
- pure real(dp) function mean_c (n, x)

  Computes the mean of a data set.
- real(dp) function median_c (n, x, srt)

  Computes the median of a data set.
- pure real(dp) function variance_c (n, x)

  Computes the sample variance of a data set.
- subroutine covariance_c (m, n, x, c, err)

Computes the covariance matrix of N data sets of M observations.
- pure real(dp) function stdev_c (n, x)

    Computes the corrected standard deviation of a data set.
- real(dp) function conf_int_c (n, x, alpha, use_t, err)

    Computes the confidence interval based upon a standard normal distribution.
- subroutine moving_average_c (n, x, npts, err)

    Applies a moving average to smooth a data set.
- real(dp) function linlsq_1var_c (n, x, y, err)

    Employs a least squares fit to determine the coefficient A in the linear system: $Y = A * X$.
- subroutine linlsq_nvar_c (m, n, npts, x, y, a, err)

    Employs a least squares fit to determine the coefficient A in the linear system: $Y = A * X$.
- subroutine get_lowess_smoothing (obj, ptr)

    Retrieves the lowess_smoothing object from the C compatible c_lowess_smoothing data structure.
- subroutine lowess_init_c (obj, n, x, y, srt, err)

    Initializes a new c_lowess_smoothing object.
- subroutine lowess_free_c (obj)

    Frees resources held by a c_lowess_smoothing object.
- subroutine lowess_smooth_c (obj, f, n, y, err)

    Performs the actual smoothing operation.
- integer(i32) function lowess_get_pt_count_c (obj)

    Gets the number of points used by the lowess_smoothing object.
- subroutine lowess_get_pts_c (obj, n, x, y)

    Gets a copy of the data points stored by the lowess_smoothing object.
- subroutine lowess_get_residual_c (obj, n, x)

    Gets the residuals from each data point.
- subroutine get_nonlinear_regression (obj, ptr)

    Retrieves the cnonlin_reg_helper object from the C compatible c_nonlinear_regression data structure.
- subroutine nlr_init_c (obj, n, x, y, fcn, ncoeff, err)

    Initializes a new c_nonlinear_regression object.
- subroutine nlr_free_c (obj)

    Frees resources held by a c_nonlinear_regression object.
- subroutine nlr_solve_c (obj, n, c, ib, err)

    Computes the solution to the nonlinear regression problem using the Levenberg-Marquardt method.
- integer(i32) function nlr_get_pt_count_c (obj)

    Gets the number of points used by the c_nonlinear_regression object.
- subroutine nlr_get_pts_c (obj, n, x, y)

    Gets a copy of the data points stored by the c_nonlinear_regression object.
- subroutine nlr_get_solver_params_c (obj, cntrl)

    Gets the nonlinear regression solver solution control parameters.
- subroutine nlr_set_solver_params_c (obj, cntrl)

    Sets the nonlinear regression solver solution control parameters.
- subroutine crh_fcn (this, x, f)

    Computes the residual between the supplied data set, and the function value given a set of coefficients.
- pure logical function crh_is_fcn_defined (this)

    Tests if the pointer to the function containing the equation to solve has been assigned.
- subroutine crh_set_fcn (this, fcn)

    Establishes a pointer to the routine containing the equations to solve.
- subroutine seb_c (n, applied, output, fullscale, rst, err)

    Computes the static error band of a data set.
- real(dp) function nonlin_c (n, applied, measured)

    Computes the best-fit nonlinearity of a data set.

- real(dp) function term_nonlin_c (n, applied, measured)

  Computes the terminal nonlinearity of a data set.
- real(dp) function hysteresis_c (n, applied, measured)

  Computes the hysteresis in an ascending/descending data set.
- real(dp) function rtz_c (n, applied, measured, tol)

  Computes the return to zero error in an ascending/descending data set.
- real(dp) function repeat_c (npts, ntests, applied, measured)

  Computes the repeatability of a sequence of tests.
- subroutine xtalk_c (npts, ndof, xerr, indices, xt, err)

  Computes the crosstalk errors for a multiple degree-of-freedom data set.
- subroutine split_c (n, x, na, ascend, nd, descend, nascend, ndescend, err)

  Splits a data set into ascending and descending components.

5.1.1 Detailed Description

curvefit_c_binding

Purpose

Provides C bindings to the curvefit library.

5.1.2 Function/Subroutine Documentation

5.1.2.1 real(dp) function curvefit_c_binding::conf_int_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), intent(in), value *alpha*, logical(c_bool), intent(in), value *use_t*, type(errorhandler), intent(inout) *err*)

Computes the confidence interval based upon a standard normal distribution.

Parameters

`in`	*n*	The number of data points.
`in`	*x*	An N-element array containing the data set.
`in`	*alpha*	The confidence level. This value must lie between zero and one such that: 0 < alpha < 1.
`in`	*use↩_t*	Set to true to use the t-distribution in the event of an unknown true standard deviation; else, set to true to use a normal distribution.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `alpha` is does not satisfy: 0 < alpha < 1.

Returns

The confidence interval as the deviation from the mean.

Remarks

The confidence interval, assuming a standard normal distribution, is as follows: mu +/- z ∗ s / sqrt(n), where mu = the mean, and s = the standard deviation. This routine computes the z ∗ s / sqrt(n) portion leaving the computation of the mean to the user.

Definition at line 847 of file curvefit_c_binding.f90.

5.1.2.2 subroutine curvefit_c_binding::covariance_c (integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(in) *x*, real(dp), dimension(n,n), intent(out) *c*, type(errorhandler), intent(inout) *err*)

Computes the covariance matrix of N data sets of M observations.

Parameters

`in`	*m*	The number of observations.
`in`	*n*	The number of data sets.
`in`	*x*	The M-by-N matrix of data.
`out`	*c*	The N-by-N matrix where the resulting covariance matrix will be written.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Definition at line 788 of file curvefit_c_binding.f90.

5.1.2.3 subroutine curvefit_c_binding::crh_fcn (class(cnonlin_reg_helper), intent(in) *this*, real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(out) *f*)

Computes the residual between the supplied data set, and the function value given a set of coefficients.

Parameters

`in`	*this*	The cnonlin_reg_helper object.
`in`	*x*	An N-element array containing the N coefficients.
`out`	*f*	An M-element array that, on output, contains the residual at each of the M data points.

Definition at line 1474 of file curvefit_c_binding.f90.

5.1.2.4 pure logical function curvefit_c_binding::crh_is_fcn_defined (class(cnonlin_reg_helper), intent(in) *this*)

Tests if the pointer to the function containing the equation to solve has been assigned.

Parameters

`in`	*this*	The cnonlin_reg_helper object.

Returns

Returns true if the pointer has been assigned; else, false.

Definition at line 1497 of file curvefit_c_binding.f90.

5.1.2.5 subroutine curvefit_c_binding::crh_set_fcn (class(cnonlin_reg_helper), intent(inout) *this*, procedure(creg_fcn), intent(in), pointer *fcn*)

Establishes a pointer to the routine containing the equations to solve.

Parameters

`in,out`	*this*	The cnonlin_reg_helper object.
`in`	*fcn*	The function pointer.

Definition at line 1509 of file curvefit_c_binding.f90.

5.1.2.6 subroutine curvefit_c_binding::get_linear_interp (type(**c_linear_interp**), intent(in), target *obj*, type(**linear_interp**), intent(out), pointer *li*)

Retrieves the linear_interp object from the C compatible c_linear_interp data structure.

Parameters

`in`	*obj*	The C compatible c_linear_interp object.
`out`	*li*	A pointer to the resulting linear_interp object. This pointer can be NULL dependent upon the state of `obj`.

Definition at line 138 of file curvefit_c_binding.f90.

5.1.2.7 subroutine curvefit_c_binding::get_lowess_smoothing (type(**c_lowess_smoothing**), intent(in), target *obj*, type(**lowess_smoothing**), intent(out), pointer *ptr*)

Retrieves the lowess_smoothing object from the C compatible c_lowess_smoothing data structure.

Parameters

`in`	*obj*	The C compatible c_lowess_smoothing object.
`out`	*ptr*	A pointer to the resulting lowess_smoothing object. This pointer can be NULL dependent upon the state of `obj`.

Definition at line 1005 of file curvefit_c_binding.f90.

5.1.2.8 subroutine curvefit_c_binding::get_nonlinear_regression (type(**c_nonlinear_regression**), intent(in), target *obj*, type(**cnonlin_reg_helper**), intent(out), pointer *ptr*)

Retrieves the cnonlin_reg_helper object from the C compatible c_nonlinear_regression data structure.

Parameters

`in`	*obj*	The C compatible c_nonlinear_regression object.
`out`	*ptr*	A pointer to the resulting cnonlin_reg_helper object. This pointer can be NULL dependent upon the state of `obj`.

Definition at line 1221 of file curvefit_c_binding.f90.

5.1.2.9 subroutine curvefit_c_binding::get_polynomial_interp (type(**c_polynomial_interp**), intent(in), target *obj*, type(**polynomial_interp**), intent(out), pointer *interp*)

Retrieves the polynomial_interp object from the C compatible c_polynomial_interp data structure.

Parameters

`in`	*obj*	The C compatible c_polynomial_interp object.
`out`	*interp*	A pointer to the resulting polynomial_interp object. This pointer can be NULL dependent upon the state of `obj`.

Definition at line 307 of file curvefit_c_binding.f90.

5.1.2.10 subroutine curvefit_c_binding::get_spline_interp (type(**c_spline_interp**), intent(in), target *obj*, type(**spline_interp**), intent(out), pointer *interp*)

Retrieves the spline_interp object from the C compatible c_spline_interp data structure.

Parameters

`in`	*obj*	The C compatible c_spline_interp object.
`out`	*interp*	A pointer to the resulting spline_interp object. This pointer can be NULL dependent upon the state of `obj`.

Definition at line 478 of file curvefit_c_binding.f90.

5.1.2.11 real(dp) function curvefit_c_binding::hysteresis_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *applied*, real(dp), dimension(n), intent(in) *measured*)

Computes the hysteresis in an ascending/descending data set.

Parameters

`in`	*n*	The number of data points.
`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.

Returns

The hysteresis error.

Definition at line 1610 of file curvefit_c_binding.f90.

5.1.2.12 pure logical(c_bool) function curvefit_c_binding::is_monotonic_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*)

Tests to see if an array is montonically increasing or decreasing.

Parameters

`in`	*n*	The number of elements in the array.
`in`	*x*	The array to test.

Returns

Returns true if `x` is monotonic; else, false.

Definition at line 122 of file curvefit_c_binding.f90.

5.1.2.13 subroutine curvefit_c_binding::lininterp_free_c (type(c_linear_interp), intent(inout), target *obj*)

Frees resources held by a c_linear_interp object.

Parameters

`in,out`	*obj*	The c_linear_interp object.

Definition at line 200 of file curvefit_c_binding.f90.

5.1.2.14 integer(i32) function curvefit_c_binding::lininterp_get_pt_count_c (type(c_linear_interp), intent(in), target *obj*)

Gets the number of points used by the interpolation object.

Parameters

`in`	*obj*	The c_linear_interp object.

Returns

The number of points.

Definition at line 248 of file curvefit_c_binding.f90.

5.1.2.15 subroutine curvefit_c_binding::lininterp_get_pts_c (type(c_linear_interp), intent(in), target *obj,* integer(i32), intent(in), value *n,* real(dp), dimension(n), intent(out) *x,* real(dp), dimension(n), intent(out) *y*)

Gets a copy of the data points stored by the interpolation object.

Parameters

`in`	*obj*	The c_linear_interp object.
`in`	*n*	The size of the buffer arrays.
`out`	*x*	An N-element array where the x-coordinate data will be written.
`out`	*y*	An N-element array where the y-coordinate data will be written.

Remarks

If `n` is different than the actual number of points that exist, the lesser of the two values will be utilized. The interpolation object can be queried to determine the quantity of stored points.

Definition at line 278 of file curvefit_c_binding.f90.

5.1.2.16 subroutine curvefit_c_binding::lininterp_init_c (type(c_linear_interp), intent(out) *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), dimension(n), intent(in) *y*, type(errorhandler), intent(inout) *err*)

Initializes a new c_linear_interp object.

Parameters

`out`	*obj*	The c_linear_interp object to initialize.
`in`	*n*	The number of data points.
`in`	*x*	An N-element array containing the x-components of each data point. This array must be monotonic (ascending or descending only).
`in`	*y*	An N-element array containing the y-components of each data point.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if `x` is not monotonically increasing or decreasing.

Definition at line 173 of file curvefit_c_binding.f90.

5.1.2.17 subroutine curvefit_c_binding::lininterp_interp_c (type(c_linear_interp), intent(in), target *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), dimension(n), intent(out) *y*)

Performs a linear interpolation to determine the points `y` that for the requested indendent variable values in `x`.

Parameters

`in`	*obj*	The c_linear_interp object.
`in`	*n*	The number of points to interpolate.
`in`	*x*	An N-element array containing the values of the independent variable at which to interpolate.
`out`	*y*	An N-element array where the interpolated values can be written.

Definition at line 225 of file curvefit_c_binding.f90.

5.1.2.18 real(dp) function curvefit_c_binding::linlsq_1var_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), dimension(n), intent(inout) *y*, type(errorhandler), intent(inout) *err*)

Employs a least squares fit to determine the coefficient A in the linear system: Y = A $*$ X.

Parameters

`in`	*n*	The number of data points.
`in`	*x*	An N-element array containing the independent variable data.
`in,out`	*y*	An N-element array containing the dependent variable data corresponding to `x`. On output, the contents of this array are overwritten as it is used for storage purposes by the algorithm.

Parameters

`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are different sizes.

Returns

The scalar coefficient A.

Definition at line 923 of file curvefit_c_binding.f90.

5.1.2.19 subroutine curvefit_c_binding::linlsq_nvar_c (integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, integer(i32), intent(in), value *npts*, real(dp), dimension(n,npts), intent(inout) *x*, real(dp), dimension(m,npts), intent(in) *y*, real(dp), dimension(m,n), intent(out) *a*, type(errorhandler), intent(inout) *err*)

Employs a least squares fit to determine the coefficient A in the linear system: Y = A ∗ X.

Parameters

`in`	*m*	The number of dependent variables.
`in`	*n*	The number of independent variables.
`in,out`	*x*	An N-by-NPTS matrix containing the P data points of the N independent variables.
`in`	*y*	An M-by-NPTS matrix containing the P data points of the M dependent variables.
`out`	*a*	The M-by-N matrix where the resulting coefficient matrix A will be written.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if any of the matrix dimensions are not compatiable. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Remarks

The algorithm attempts to compute the coefficient matrix A as follows. Y ∗ X∗∗T = A ∗ X ∗ X∗∗T Y ∗ X∗∗T ∗ INV(X ∗ X∗∗T) = A This does require that X ∗ X∗∗T does not result in a singular matrix. To handle the situation where X ∗ X∗∗T is singular, the Moore-Penrose pseudo-inverse, computed by means of singular value decomposition, is utilized to still arrive at a solution that, at minimum, has a minimum Euclidean norm of its residual. Let: PINV(X) = X∗∗T ∗ INV(X ∗ X∗∗T), Then: A = Y ∗ PINV(X)

Definition at line 976 of file curvefit_c_binding.f90.

5.1.2.20 subroutine curvefit_c_binding::lowess_free_c (type(c_lowess_smoothing), intent(inout), target *obj*)

Frees resources held by a c_lowess_smoothing object.

Parameters

`in,out`	*obj*	The c_lowess_smoothing object.

Definition at line 1069 of file curvefit_c_binding.f90.

5.1.2.21 integer(i32) function curvefit_c_binding::lowess_get_pt_count_c (type(c_lowess_smoothing), intent(in) *obj*)

Gets the number of points used by the lowess_smoothing object.

Parameters

`in`	*obj*	The c_lowess_smoothing object.

Returns

The number of points.

Definition at line 1138 of file curvefit_c_binding.f90.

5.1.2.22 subroutine curvefit_c_binding::lowess_get_pts_c (type(c_lowess_smoothing), intent(in) *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(out) *x*, real(dp), dimension(n), intent(out) *y*)

Gets a copy of the data points stored by the lowess_smoothing object.

Parameters

`in`	*obj*	The c_lowess_smoothing object.
`in`	*n*	The size of the buffer arrays.
`out`	*x*	An N-element array where the x-coordinate data will be written.
`out`	*y*	An N-element array where the y-coordinate data will be written.

Remarks

If `n` is different than the actual number of points that exist, the lesser of the two values will be utilized. The lowess_smoothing object can be queried to determine the quantity of stored points.

Definition at line 1169 of file curvefit_c_binding.f90.

5.1.2.23 subroutine curvefit_c_binding::lowess_get_residual_c (type(c_lowess_smoothing), intent(in) *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(out) *x*)

Gets the residuals from each data point.

Parameters

`in`	*this*	The c_lowess_smoothing object.
`in`	*n*	The number of elements available in the buffer array `x`.
`out`	*x*	An N-element array where the residual data should be written.

Definition at line 1197 of file curvefit_c_binding.f90.

5.1.2.24 subroutine curvefit_c_binding::lowess_init_c (type(c_lowess_smoothing), intent(out) *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), dimension(n), intent(in) *y*, logical(c_bool), intent(in), value *srt*, type(errorhandler), intent(inout) *err*)

Initializes a new c_lowess_smoothing object.

Parameters

`out`	*obj*	The c_lowess_smoothing object.
`in`	*n*	The number of data points.
`in`	*x*	An N-element array containing the x-coordinate data. Ideally, the data set should be monotonically increasing; however, if it is not, it may be sorted by the routine, dependent upon the value of `srt`.
`in`	*y*	An N-element array containing the y-coordinate data.
`in`	*srt*	A logical flag determining if `x` should be sorted.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are not the same size. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Definition at line 1041 of file curvefit_c_binding.f90.

5.1.2.25 subroutine curvefit_c_binding::lowess_smooth_c (type(c_lowess_smoothing), intent(inout) *obj*, real(dp), intent(in), value *f*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(out) *y*, type(errorhandler), intent(inout) *err*)

Performs the actual smoothing operation.

Parameters

`in,out`	*obj*	The c_lowess_smoothing object.
`in`	*f*	Specifies the amount of smoothing. More specifically, this value is the fraction of points used to compute each value. As this value increases, the output becomes smoother. Choosing a value in the range of 0.2 to 0.8 usually results in a good fit. As such, a reasonable starting point, in the absence of better information, is a value of 0.5.
`in`	*n*	The size of the buffer `y`. Ideally, this parameter is equal to the number of points stored in `obj`; however, the routine will only traverse the minimum of the this parameter or the number of points stored in `obj`.
`out`	*y*	An N-element array to which the smoothed data will be written.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has been defined. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Definition at line 1106 of file curvefit_c_binding.f90.

5.1.2.26 pure real(dp) function curvefit_c_binding::mean_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*)

Computes the mean of a data set.

Parameters

`in`	*n*	The number of data points.
`in`	*x*	An N-element array containing the data set.

Returns

The mean of `x`.

Definition at line 728 of file curvefit_c_binding.f90.

5.1.2.27 real(dp) function curvefit_c_binding::median_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(inout) *x*, logical(c_bool), intent(in), value *srt*)

Computes the median of a data set.

Parameters

`in`	*n*	The number of data points.
`in,out`	*x*	The data set whose median is to be found. Ideally, the data set should be monotonically increasing; however, if it is not, it may be sorted by the routine, dependent upon the value of `srt`. On output, the array contents are unchanged; however, they may be sorted into ascending order (dependent upon the value of `srt`).
`in`	*srt*	A logical flag determining if `x` should be sorted.

Returns

The median of `x`.

Definition at line 747 of file curvefit_c_binding.f90.

5.1.2.28 subroutine curvefit_c_binding::moving_average_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(inout) *x*, integer(i32), intent(in), value *npts*, type(errorhandler), intent(inout) *err*)

Applies a moving average to smooth a data set.

Parameters

`in`	*n*	The number of data points.
`in,out`	*x*	On input, the signal to smooth. On output, the smoothed signal.
`in`	*npts*	The size of the averaging window. This value must be at least 2, but no more than the number of elements in `x`.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `npts` is less than 2, or greater than the length of `x`.
		• CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Definition at line 887 of file curvefit_c_binding.f90.

5.1.2.29 subroutine curvefit_c_binding::nlr_free_c (type(c_nonlinear_regression), intent(inout), target *obj*)

Frees resources held by a c_nonlinear_regression object.

Parameters

in,out	*obj*	The c_nonlinear_regression object.

Definition at line 1293 of file curvefit_c_binding.f90.

5.1.2.30 integer(i32) function curvefit_c_binding::nlr_get_pt_count_c (type(c_nonlinear_regression), intent(in) *obj*)

Gets the number of points used by the c_nonlinear_regression object.

Parameters

in	*obj*	The c_nonlinear_regression object.

Returns

The number of points.

Definition at line 1363 of file curvefit_c_binding.f90.

5.1.2.31 subroutine curvefit_c_binding::nlr_get_pts_c (type(c_nonlinear_regression), intent(in) *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(out) *x*, real(dp), dimension(n), intent(out) *y*)

Gets a copy of the data points stored by the c_nonlinear_regression object.

Parameters

in	*obj*	The c_nonlinear_regression object.
in	*n*	The size of the buffer arrays.
out	*x*	An N-element array where the x-coordinate data will be written.
out	*y*	An N-element array where the y-coordinate data will be written.

Remarks

If `n` is different than the actual number of points that exist, the lesser of the two values will be utilized. The c_nonlinear_regression object can be queried to determine the quantity of stored points.

Definition at line 1394 of file curvefit_c_binding.f90.

5.1.2.32 subroutine curvefit_c_binding::nlr_get_solver_params_c (type(c_nonlinear_regression), intent(in) *obj*, type(solver_control), intent(out) *cntrl*)

Gets the nonlinear regression solver solution control parameters.

Parameters

`in`	*obj*	The c_nonlinear_regression object.
`out`	*cntrl*	The solver_control object that, on output, will contain the current solver control parameters.

Definition at line 1421 of file curvefit_c_binding.f90.

5.1.2.33 subroutine curvefit_c_binding::nlr_init_c (type(c_nonlinear_regression), intent(out) *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), dimension(n), intent(in) *y*, type(c_funptr), intent(in), value *fcn*, integer(i32), intent(in), value *ncoeff*, type(errorhandler), intent(inout) *err*)

Initializes a new c_nonlinear_regression object.

Parameters

`out`	*obj*	The c_nonlinear_regression object.
`in`	*n*	The number of data points.
`in`	*x*	An N-element containing the independent variable values of the data set.
`in`	*y*	An N-element array of the dependent variables corresponding to `x`.
`in`	*fcn*	A pointer to the function whose coefficients are to be determined.
`in`	*ncoeff*	The number of coefficients in the function defined in `fcn`.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_INVALID_INPUT_ERROR: Occurs if `ncoeff` is less than or equal to zero.

Definition at line 1260 of file curvefit_c_binding.f90.

5.1.2.34 subroutine curvefit_c_binding::nlr_set_solver_params_c (type(c_nonlinear_regression), intent(inout) *obj*, type(solver_control), intent(in) *cntrl*)

Sets the nonlinear regression solver solution control parameters.

Parameters

`in,out`	*obj*	The c_nonlinear_regression object.
`in`	*cntrl*	The solver_control object that contains the current solver control parameters.

Definition at line 1446 of file curvefit_c_binding.f90.

5.1.2.35 subroutine curvefit_c_binding::nlr_solve_c (type(c_nonlinear_regression), intent(inout) *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(inout) *c*, type(iteration_behavior), intent(out) *ib*, type(errorhandler), intent(inout) *err*)

Computes the solution to the nonlinear regression problem using the Levenberg-Marquardt method.

Parameters

in,out	*obj*	The c_nonlinear_regression object.
in	*n*	The number of coefficients to determine.
in,out	*c*	On input, an array containing initial estimates of the coefficients. On output, the comptued coefficient values.
out	*ib*	An output parameter that allows the caller to obtain iteration performance statistics.
in,out	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_INVALID_OPERATION_ERROR: Occurs if no equations have been defined. • CF_INVALID_INPUT_ERROR: Occurs if the number of equations is less than than the number of variables. • CF_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized correctly. • CF_CONVERGENCE_ERROR: Occurs if the line search cannot converge within the allowed number of iterations. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_TOLERANCE_TOO_SMALL_ERROR: Occurs if the requested tolerance is to small to be practical for the problem at hand.

Definition at line 1333 of file curvefit_c_binding.f90.

5.1.2.36 real(dp) function curvefit_c_binding::nonlin_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *applied*, real(dp), dimension(n), intent(in) *measured*)

Computes the best-fit nonlinearity of a data set.

Parameters

in	*n*	The number of data points.
in	*applied*	An N-element array containing the values applied to the measurement instrument.
in	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.

Returns

The nonlinearity error.

Definition at line 1568 of file curvefit_c_binding.f90.

5.1.2.37 subroutine curvefit_c_binding::polyinterp_free_c (type(c_polynomial_interp), intent(inout), target *obj*)

Frees resources held by a c_polynomial_interp object.

Parameters

in,out	*obj*	The c_polynomial_interp object.

Definition at line 371 of file curvefit_c_binding.f90.

5.1.2.38 integer(i32) function curvefit_c_binding::polyinterp_get_pt_count_c (type(c_polynomial_interp), intent(in), target *obj*)

Gets the number of points used by the interpolation object.

Parameters

`in`	*obj*	The c_polynomial_interp object.

Returns

The number of points.

Definition at line 419 of file curvefit_c_binding.f90.

5.1.2.39 subroutine curvefit_c_binding::polyinterp_get_pts_c (type(c_polynomial_interp), intent(in), target *obj,* integer(i32), intent(in), value *n,* real(dp), dimension(n), intent(out) *x,* real(dp), dimension(n), intent(out) *y*)

Gets a copy of the data points stored by the interpolation object.

Parameters

`in`	*obj*	The c_polynomial_interp object.
`in`	*n*	The size of the buffer arrays.
`out`	*x*	An N-element array where the x-coordinate data will be written.
`out`	*y*	An N-element array where the y-coordinate data will be written.

Remarks

If `n` is different than the actual number of points that exist, the lesser of the two values will be utilized. The interpolation object can be queried to determine the quantity of stored points.

Definition at line 449 of file curvefit_c_binding.f90.

5.1.2.40 subroutine curvefit_c_binding::polyinterp_init_c (type(c_polynomial_interp), intent(out) *obj,* integer(i32), intent(in), value *n,* real(dp), dimension(n), intent(in) *x,* real(dp), dimension(n), intent(in) *y,* integer(i32), intent(in), value *order,* type(errorhandler), intent(inout) *err*)

Initializes a new c_polynomial_interp object.

Parameters

`out`	*obj*	The c_polynomial_interp object to initialize.
`in`	*n*	The number of data points.
`in`	*x*	An N-element array containing the x-components of each data point. This array must be monotonic (ascending or descending only).
`in`	*y*	An N-element array containing the y-components of each data point.
`in`	*order*	The order of the interpolating polynomial.

Parameters

`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if `x` is not monotonically increasing or decreasing. • CF_INVALID_INPUT_ERROR: Occurs if `order` is less than 1.

Definition at line 344 of file curvefit_c_binding.f90.

5.1.2.41 subroutine curvefit_c_binding::polyinterp_interp_c (type(c_polynomial_interp), intent(in), target *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), dimension(n), intent(out) *y*)

Performs a polynomial interpolation to determine the points `y` that for the requested indendent variable values in `x`.

Parameters

`in`	*obj*	The c_polynomial_interp object.
`in`	*n*	The number of points to interpolate.
`in`	*x*	An N-element array containing the values of the independent variable at which to interpolate.
`out`	*y*	An N-element array where the interpolated values can be written.

Definition at line 396 of file curvefit_c_binding.f90.

5.1.2.42 real(dp) function curvefit_c_binding::repeat_c (integer(i32), intent(in), value *npts*, integer(i32), intent(in), value *ntests*, real(dp), dimension(npts, ntests), intent(in) *applied*, real(dp), dimension(npts, ntests), intent(in) *measured*)

Computes the repeatability of a sequence of tests.

Parameters

`in`	*npts*	The number of data points per test.
`in`	*ntests*	The number of tests.
`in`	*applied*	An NPTS-by-NTEST matrix containing at least 2 columns (tests) of NPTS values applied to the measurement instrument.
`in`	*measured*	An NPTS-by-NTEST matrix containing the corresponding calibrated output from the instrument.

Returns

The largest magnitude deviation from the initial test.

Remarks

Repeatability is considered as the largest magnitude deviation of subsequent tests from the initial test. Noting that it is very likely that consecutive test points will vary slightly, test 2 through test N are linearly interpolated such that their test points line up with those from test 1.

Definition at line 1664 of file curvefit_c_binding.f90.

5.1.2.43 real(dp) function curvefit_c_binding::rtz_c (integer(i32), intent(in), value *n,* real(dp), dimension(n), intent(in) *applied,* real(dp), dimension(n), intent(in) *measured,* real(dp), intent(in), value *tol*)

Computes the return to zero error in an ascending/descending data set.

Parameters

`in`	*n*	The number of data points.
`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.
`in`	*tol*	An input argument that specifies the tolerance used in finding the matching zero data point.

Returns

The return to zero error.

Definition at line 1634 of file curvefit_c_binding.f90.

5.1.2.44 subroutine curvefit_c_binding::seb_c (integer(i32), intent(in), value *n,* real(dp), dimension(n), intent(in) *applied,* real(dp), dimension(n), intent(in) *output,* real(dp), intent(in), value *fullscale,* type(seb_results), intent(out) *rst,* type(errorhandler), intent(inout) *err*)

Computes the static error band of a data set.

Parameters

`in`	*n*	The number of data points.
`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*output*	An N-element array containing the values output by the instrument as a result of the values given in `applied`.
`in`	*fullscale*	The full scale measurement value for the instrument. The units must be consistent with those of `applied`.
`out`	*rst*	An seb_results object where the calculation results will be written.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `fullscale` is sufficiently close to zero to be considered zero. Sufficiently close in this instance is considered to be the square root of machine precision.

Definition at line 1537 of file curvefit_c_binding.f90.

5.1.2.45 subroutine curvefit_c_binding::splineinterp_diff1_c (type(c_spline_interp), intent(in), target *obj,* integer(i32), intent(in), value *n,* real(dp), dimension(n), intent(in) *x,* real(dp), dimension(n), intent(out) *y*)

Computes the interpolated first derivative.

Parameters

`in`	*obj*	The c_spline_interp object.
`in`	*n*	The number of points to interpolate.
`in`	*x*	An N-element array containing the values of the independent variable at which to interpolate.
`out`	*y*	An N-element array where the interpolated values can be written.

Definition at line 677 of file curvefit_c_binding.f90.

5.1.2.46 subroutine curvefit_c_binding::splineinterp_diff2_c (type(c_spline_interp), intent(in), target *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), dimension(n), intent(out) *y*)

Computes the interpolated second derivative.

Parameters

`in`	*obj*	The c_spline_interp object.
`in`	*n*	The number of points to interpolate.
`in`	*x*	An N-element array containing the values of the independent variable at which to interpolate.
`out`	*y*	An N-element array where the interpolated values can be written.

Definition at line 703 of file curvefit_c_binding.f90.

5.1.2.47 subroutine curvefit_c_binding::splineinterp_free_c (type(c_spline_interp), intent(inout), target *obj*)

Frees resources held by a c_spline_interp object.

Parameters

`in,out`	*obj*	The c_spline_interp object.

Definition at line 569 of file curvefit_c_binding.f90.

5.1.2.48 integer(i32) function curvefit_c_binding::splineinterp_get_pt_count_c (type(c_spline_interp), intent(in), target *obj*)

Gets the number of points used by the interpolation object.

Parameters

`in`	*obj*	The c_spline_interp object.

Returns

The number of points.

Definition at line 617 of file curvefit_c_binding.f90.

5.1.2.49 subroutine curvefit_c_binding::splineinterp_get_pts_c (type(c_spline_interp), intent(in), target *obj,* integer(i32), intent(in), value *n,* real(dp), dimension(n), intent(out) *x,* real(dp), dimension(n), intent(out) *y*)

Gets a copy of the data points stored by the interpolation object.

Parameters

in	*obj*	The c_spline_interp object.
in	*n*	The size of the buffer arrays.
out	*x*	An N-element array where the x-coordinate data will be written.
out	*y*	An N-element array where the y-coordinate data will be written.

Remarks

If `n` is different than the actual number of points that exist, the lesser of the two values will be utilized. The interpolation object can be queried to determine the quantity of stored points.

Definition at line 647 of file curvefit_c_binding.f90.

5.1.2.50 subroutine curvefit_c_binding::splineinterp_init_c (type(c_spline_interp), intent(out) *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), dimension(n), intent(in) *y*, integer(i32), intent(in), value *ibcbeg*, real(dp), intent(in), value *ybcbeg*, integer(i32), intent(in), value *ibcend*, real(dp), intent(in), value *ybcend*, type(errorhandler), intent(inout) *err*)

Initializes a new c_spline_interp object.

Parameters

out	*obj*	The c_spline_interp object to initialize.
in	*n*	The number of data points.
in	*x*	An N-element array containing the x-components of each data point. This array must be monotonic (ascending or descending only).
in	*y*	An N-element array containing the y-components of each data point.
in	*ibcbeg*	An input that defines the nature of the boundary condition at the beginning of the spline. If an invalid parameter is used, the code defaults to SPLINE_QUADRATIC_OVER_INTERVAL. • SPLINE_QUADRATIC_OVER_INTERVAL: The spline is quadratic over its initial interval. No value is required for `ybcbeg`. • SPLINE_KNOWN_FIRST_DERIVATIVE: The spline's first derivative at its initial point is provided in `ybcbeg`. • SPLINE_KNOWN_SECOND_DERIVATIVE: The spline's second derivative at its initial point is provided in `ybcbeg`. • SPLINE_CONTINUOUS_THIRD_DERIVATIVE: The third derivative is continuous at x(2). No value is required for `ybcbeg`.
in	*ybcbeg*	If needed, the value of the initial point boundary condition. If not needed, this parameter is ignored.

Parameters

`in`	*ibcend*	An input that defines the nature of the boundary condition at the end of the spline. If an invalid parameter is used, the code defaults to SPLINE_QUADRATIC_OVER_INTERVAL. • SPLINE_QUADRATIC_OVER_INTERVAL: The spline is quadratic over its final interval. No value is required for `ybcend`. • SPLINE_KNOWN_FIRST_DERIVATIVE: The spline's first derivative at its initial point is provided in `ybcend`. • SPLINE_KNOWN_SECOND_DERIVATIVE: The spline's second derivative at its initial point is provided in `ybcend`. • SPLINE_CONTINUOUS_THIRD_DERIVATIVE: The third derivative is continuous at x(n-1). No value is required for `ybcend`.
`in`	*ybcend*	If needed, the value of the final point boundary condition. If not needed, this parameter is ignored.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if `x` is not monotonically increasing or decreasing.

Definition at line 539 of file curvefit_c_binding.f90.

5.1.2.51 subroutine curvefit_c_binding::splineinterp_interp_c (type(c_spline_interp), intent(in), target *obj*, integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, real(dp), dimension(n), intent(out) *y*)

Performs a spline interpolation to determine the points `y` that for the requested indendent variable values in `x`.

Parameters

`in`	*obj*	The c_spline_interp object.
`in`	*n*	The number of points to interpolate.
`in`	*x*	An N-element array containing the values of the independent variable at which to interpolate.
`out`	*y*	An N-element array where the interpolated values can be written.

Definition at line 594 of file curvefit_c_binding.f90.

5.1.2.52 subroutine curvefit_c_binding::split_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*, integer(i32), intent(in), value *na*, real(dp), dimension(na), intent(out) *ascend*, integer(i32), intent(in), value *nd*, real(dp), dimension(nd), intent(out) *descend*, integer(i32), intent(out) *nascend*, integer(i32), intent(out) *ndescend*, type(errorhandler), intent(inout) *err*)

Splits a data set into ascending and descending components.

Parameters

`in`	*n*	The number of data points in `x`.
`in`	*x*	An N-element array containing the data set to split.
`in`	*na*	The capacity of `ascend`.
`out`	*ascend*	An array where the ascending points will be written. Ensure this array is appropriately sized to accept all the ascending points (it can be oversized).
`in`	*nd*	The capacity of `descend`.
`out`	*descend*	An array where the descending points will be written. Ensure this array is appropriately sized to accept all the descending points (it can be oversized).
`out`	*nascend*	The actual number of values written into `ascend`.
`out`	*ndescend*	The actual number of values written into `descend`.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if either `ascend` or `descend` is too small to actually accept all of the necessary data.

Remarks

The routine operates by finding the first occurrence where the data set is no longer monotonic, and then copies everything prior to that value, along with the the inflection value, into the output ascending data array. The routine then searches for either a change in direction, or a value that matches the first value in the ascending data set within some tolerance to determine the bounds on the descending data set. Once the bounds are determined, the descending data set is copied from the original array and placed in the output descending data array. This then means that any remaining data in the original data set that lies after either of the aforementioned sets is ignored.

Example

```
Given the following array X,
 X:
  0.0000000000000000
  0.38905000686645508
  0.77815997600555420
  0.97268998622894287
  1.1671400070190430
  1.5559999942779541
  1.9448399543762207
  0.97259998321533203
  -9.9999997473787516E-006

This routine splits the array into the following ascending and
descending arrays.

ASCENDING:
  0.0000000000000000
  0.38905000686645508
  0.77815997600555420
  0.97268998622894287
  1.1671400070190430
  1.5559999942779541
  1.9448399543762207

DESCENDING:
  1.9448399543762207
  0.97259998321533203
  -9.9999997473787516E-006
```

Definition at line 1782 of file curvefit_c_binding.f90.

5.1.2.53 pure real(dp) function curvefit_c_binding::stdev_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*)

Computes the corrected standard deviation of a data set.

Parameters

`in`	*n*	The number of data points.
`in`	*x*	An N-element array containing the data set.

Returns

The standard deviation of `x`.

Definition at line 814 of file curvefit_c_binding.f90.

5.1.2.54 real(dp) function curvefit_c_binding::term_nonlin_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *applied*, real(dp), dimension(n), intent(in) *measured*)

Computes the terminal nonlinearity of a data set.

Parameters

`in`	*n*	The number of data points.
`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.

Returns

The nonlinearity error.

Definition at line 1589 of file curvefit_c_binding.f90.

5.1.2.55 pure real(dp) function curvefit_c_binding::variance_c (integer(i32), intent(in), value *n*, real(dp), dimension(n), intent(in) *x*)

Computes the sample variance of a data set.

Parameters

`in`	*n*	The number of data points.
`in`	*x*	An N-element array containing the data set.

Returns

The variance of `x`.

Remarks

To avoid overflow-type issues, Welford's algorithm is employed. A simple illustration of this algorithm can be found `here`.

Definition at line 767 of file curvefit_c_binding.f90.

5.1.2.56 subroutine curvefit_c_binding::xtalk_c (integer(i32), intent(in), value *npts,* integer(i32), intent(in), value *ndof,* real(dp), dimension(npts, ndof), intent(in) *xerr,* integer(i32), dimension(2∗ndof), intent(in) *indices,* real(dp), dimension(ndof, ndof), intent(out) *xt,* type(errorhandler), intent(inout) *err*)

Computes the crosstalk errors for a multiple degree-of-freedom data set.

Parameters

`in`	*npts*	The number of data points in each degree of freedom.
`in`	*ndof*	The number of degrees of freedom.
`in`	*xerr*	An NPTS-by-NDOF matrix containing the measurement error values (computed such that XERR = X MEASURED - X APPLIED).
`in`	*indices*	A 2∗NDOF element array containing row indices defining the rows where each degree-of-freedom was applied in the data set `xerr`.
`out`	*xt*	An NDOF-by-NDOF matrix that, on output, will contain the crosstalk errors such that each loaded degree of freedom is represented by its own row, and each responding degree of freedom is represented by its own column.
`in,out`	*err*	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows. • CF_ARRAY_INDEX_ERROR: Occurs if any of the entries in `indices` are outside the row bounds of `xerr`.

Definition at line 1696 of file curvefit_c_binding.f90.

5.2 curvefit_calibration Module Reference

curvefit_calibration

Data Types

- interface crosstalk

  Computes the crosstalk errors for a multiple degree-of-freedom data set.
- interface hysteresis

  Computes the hysteresis in an ascending/descending data set.
- interface IDAMAX
- interface nonlinearity

  Computes the best-fit nonlinearity of a data set.
- interface repeatability

  Computes the repeatability of a sequence of tests.
- interface return_to_zero

  Computes the return to zero error in an ascending/descending data set.
- interface seb

  Computes the static error band of a data set.
- type seb_results

  Defines a container for static error band related information.
- interface split_ascend_descend

  Splits a data set into ascending and descending components.
- interface terminal_nonlinearity

  Computes the terminal nonlinearity of a data set.

Functions/Subroutines

- type(seb_results) function seb_1 (applied, output, fullscale, err)

  Computes the static error band of a data set.
- real(dp) function bf_nonlin (applied, measured, err)

  Computes the best-fit nonlinearity of a data set.
- real(dp) function term_nonlin (applied, measured, err)

  Computes the terminal nonlinearity of a data set.
- real(dp) function hysteresis_1 (xascend, ascend, xdescend, descend, err)

  Computes the hysteresis in an ascending/descending data set.
- real(dp) function hysteresis_2 (applied, measured, err)

  Computes the hysteresis in an ascending/descending data set.
- real(dp) function rtz_1 (applied, measured, tol, err)

  Computes the return to zero error in an ascending/descending data set.
- real(dp) function repeat_1 (applied, measured, err)

  Computes the repeatability of a sequence of tests.
- real(dp) function, dimension(size(xerr, 2), size(xerr, 2)) xtalk_1 (xerr, indices, err)

  Computes the crosstalk errors for a multiple degree-of-freedom data set.
- subroutine split_ascend_descend_1 (x, ascend, descend, nascend, ndescend, err)

  Splits a data set into ascending and descending components.

5.2.1 Detailed Description

curvefit_calibration

Purpose

To provide routines for computing calibration performance metrics commonly used to assess the fitness of a calibration curve fit.

References

- Wheeler, Anthony J., Ganji, Ahmad R., "Introduction to Engineering Experimentation," Third Edition, Prentice Hall.

5.2.2 Function/Subroutine Documentation

5.2.2.1 real(dp) function curvefit_calibration::bf_nonlin (real(dp), dimension(:), intent(in) *applied,* real(dp), dimension(:), intent(in) *measured,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the best-fit nonlinearity of a data set.

Parameters

`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The nonlinearity error.

Definition at line 211 of file curvefit_calibration.f90.

5.2.2.2 real(dp) function curvefit_calibration::hysteresis_1 (real(dp), dimension(:), intent(in) *xascend,* real(dp), dimension(:), intent(in) *ascend,* real(dp), dimension(:), intent(in) *xdescend,* real(dp), dimension(:), intent(in) *descend,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the hysteresis in an ascending/descending data set.

Parameters

`in`	*xascend*	An N-element array containing the ascending calibration points. This array must be monotonically increasing or decreasing.
`in`	*ascend*	An N-element array containing the sensor output to the calibration points in `xascend`.
`in`	*xdescend*	An M-element array containing the descending calibration points. This array must be monotonically increasing or decreasing.
`in`	*descend*	An M-element array containing the sensor output to the calibration points in `xdescend`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `xascend` and `ascend` are not the same size, or if `xdescend` and `descend` are not the same size. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if the calibration data is not monotonic in nature (either ascending or descending).

Returns

The hysteresis error.

Remarks

In order to account for slight variations between similar ascending and descending points, the algorithm used performs a linear interpolation between data points. The resulting interpolated value is then used to compute the reported hysteresis error.

Definition at line 353 of file curvefit_calibration.f90.

5.2.2.3 real(dp) function curvefit_calibration::hysteresis_2 (real(dp), dimension(:), intent(in) *applied,* real(dp), dimension(:), intent(in) *measured,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the hysteresis in an ascending/descending data set.

Parameters

`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.

Parameters

`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if the calibration data is not monotonic in nature (either ascending or descending). • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The hysteresis error.

Remarks

In order to account for slight variations between similar ascending and descending points, the algorithm used performs a linear interpolation between data points. The resulting interpolated value is then used to compute the reported hysteresis error.

Definition at line 444 of file curvefit_calibration.f90.

5.2.2.4 real(dp) function curvefit_calibration::repeat_1 (real(dp), dimension(:,:), intent(in) *applied,* real(dp), dimension(:,:), intent(in) *measured,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the repeatability of a sequence of tests.

Parameters

`in`	*applied*	An NPTS-by-NTEST matrix containing at least 2 columns (tests) of NPTS values applied to the measurement instrument.
`in`	*measured*	An NPTS-by-NTEST matrix containing the corresponding calibrated output from the instrument.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The largest magnitude deviation from the initial test.

Remarks

Repeatability is considered as the largest magnitude deviation of subsequent tests from the initial test. Noting that it is very likely that consecutive test points will vary slightly, test 2 through test N are linearly interpolated such that their test points line up with those from test 1.

Definition at line 608 of file curvefit_calibration.f90.

5.2.2.5 real(dp) function curvefit_calibration::rtz_1 (real(dp), dimension(:), intent(in) *applied*, real(dp), dimension(:), intent(in) *measured*, real(dp), intent(in), optional *tol*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the return to zero error in an ascending/descending data set.

Parameters

`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.
`in`	*tol*	An optional input that specifies the tolerance used in finding the matching data points. If no value is specified, the default value of the square root of machine precision times the largest magnitude value in `xcal` is used.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The return to zero error.

Definition at line 522 of file curvefit_calibration.f90.

5.2.2.6 type(seb_results) function curvefit_calibration::seb_1 (real(dp), dimension(:), intent(in) *applied*, real(dp), dimension(:), intent(in) *output*, real(dp), intent(in) *fullscale*, class(errors), intent(out), optional, target *err*) `[private]`

Computes the static error band of a data set.

Parameters

`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*output*	An N-element array containing the values output by the instrument as a result of the values given in `applied`.
`in`	*fullscale*	The full scale measurement value for the instrument. The units must be consistent with those of `applied`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `output` are not the same size. • CF_INVALID_INPUT_ERROR: Occurs if `fullscale` is sufficiently close to zero to be considered zero. Sufficiently close in this instance is considered to be the square root of machine precision.

Returns

The static error band information.

Definition at line 126 of file curvefit_calibration.f90.

5.2.2.7 subroutine curvefit_calibration::split_ascend_descend_1 (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(out) *ascend*, real(dp), dimension(:), intent(out) *descend*, integer(i32), intent(out) *nascend*, integer(i32), intent(out) *ndescend*, class(errors), intent(inout), optional, target *err*) [private]

Splits a data set into ascending and descending components.

Parameters

in	*x*	An N-element array containing the data set to split.
out	*ascend*	An array where the ascending points will be written. Ensure this array is appropriately sized to accept all the ascending points (it can be oversized).
out	*descend*	An array where the descending points will be written. Ensure this array is appropriately sized to accept all the descending points (it can be oversized).
out	*nascend*	The actual number of values written into `ascend`.
out	*ndescend*	The actual number of values written into `descend`.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if either `ascend` or `descend` is too small to actually accept all of the necessary data.

Remarks

The routine operates by finding the first occurrence where the data set is no longer monotonic, and then copies everything prior to that value, along with the the inflection value, into the output ascending data array. The routine then searches for either a change in direction, or a value that matches the first value in the ascending data set within some tolerance to determine the bounds on the descending data set. Once the bounds are determined, the descending data set is copied from the original array and placed in the output descending data array. This then means that any remaining data in the original data set that lies after either of the aforementioned sets is ignored.

Example

```
Given the following array X,
 X:
  0.0000000000000000
  0.38905000686645508
  0.77815997600555420
  0.97268998622894287
  1.1671400070190430
  1.5559999942779541
  1.9448399543762207
  0.97259998321533203
  -9.9999997473787516E-006

This routine splits the array into the following ascending and
descending arrays.

ASCENDING:
  0.0000000000000000
```

```
      0.38905000686645508
      0.77815997600555420
      0.97268998622894287
      1.1671400070190430
      1.5559999942779541
      1.9448399543762207

    DESCENDING:
      1.9448399543762207
      0.97259998321533203
      -9.9999997473787516E-006
```

Definition at line 1011 of file curvefit_calibration.f90.

5.2.2.8 real(dp) function curvefit_calibration::term_nonlin (real(dp), dimension(:), intent(in) *applied*, real(dp), dimension(:), intent(in) *measured*, class(errors), intent(inout), optional, target *err*) [private]

Computes the terminal nonlinearity of a data set.

Parameters

in	*applied*	An N-element array containing the values applied to the measurement instrument.
in	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The terminal nonlinearity error.

Definition at line 267 of file curvefit_calibration.f90.

5.2.2.9 real(dp) function, dimension(size(xerr, 2), size(xerr, 2)) curvefit_calibration::xtalk_1 (real(dp), dimension(:,:), intent(in) *xerr*, integer(i32), dimension(:), intent(in) *indices*, class(errors), intent(inout), optional, target *err*) [private]

Computes the crosstalk errors for a multiple degree-of-freedom data set.

Parameters

in	*xerr*	An NPTS-by-NDOF matrix containing the measurement error values (computed such that XERR = X MEASURED - X APPLIED).
in	*indices*	A 2∗NDOF element array containing row indices defining the rows where each degree-of-freedom was applied in the data set `xerr`.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `indices` is not 2∗NDOF in size. • CF_ARRAY_INDEX_ERROR: Occurs if any of the entries in `indices` are outside the row bounds of `xerr`.

Returns

A NDOF-by-NDOF matrix containing the crosstalk errors such that each loaded degree of freedom is represented by its own row, and each responding degree of freedom is represented by its own column.

Usage

The following program computes the crosstalk errors for a 2 DOF system. The applied data is as follows (there are 34 data points for each DOF).

```
0              0
3000           0
6000           0
7500           0
9000           0
12000          0
15000          0
7500           0
0              0
0              0
-3000          0
-6000          0
-7500          0
-9000          0
-12000         0
-15000         0
-7500          0
0              0
--------------------
0              0
0              67.79087067
0              135.5817413
0              203.3726196
0              271.1634827
0              338.9543762
0              203.3726196
0              0
0              0
0              -67.79087067
0              -135.5817413
0              -203.3726196
0              -271.1634827
0              -338.9543762
0              -203.3726196
0              0
```

The data output from the instrument under test is as follows.

```
0                  0
0.389050007        1.22E-03
0.778159976        2.59E-03
0.972689986        2.90E-03
1.167140007        3.14E-03
1.555999994        3.38E-03
1.944839954        3.56E-03
0.972599983        4.77E-03
-1E-05             -1.00E-05
0                  0
-0.388886005       2.10E-04
-0.777750015       5.10E-04
-0.972150028       6.90E-04
-1.166540027       8.80E-04
-1.555330038       1.30E-03
-1.944100022       1.78E-03
-0.971710026       5.80E-04
4.00E-05           3.00E-05
---------------------------
0                  0
-4.40E-04          0.271560013
-1.30E-03          0.543290019
-2.40E-03          0.815069973
-3.82E-03          1.086820006
-5.28E-03          1.358809948
```

```
-2.57E-03          0.815530002
1.50E-04           1.00E-05
0                  0
1.44E-03           -0.271450013
3.06E-03           -0.543120027
4.46E-03           -0.814930022
5.67E-03           -1.086799979
6.88E-03           -1.35879004
4.51E-03           -0.815479994
-2.00E-05          0
```

The code to compute the crosstalk errors given the above raw data is then as follows.

```
program main
  ! Parameters
  integer(i32), parameter :: npts = 34
  integer(i32), parameter :: ndof = 2

  ! Local Variables
  integer(i32) :: indices(2*ndof)
  real(dp), dimension(npts, ndof) :: xin, xout, xerr, xmeas
  real(dp), dimension(ndof, npts) :: xint, xmeast
  real(dp), dimension(ndof, ndof) :: c, ans, xt

  ! Initialization
  xin = reshape([0.0, 3000.0, 6000.0, 7500.0, 9000.0, 12000.0, &
      15000.0, 7500.0, 0.0, 0.0, -3000.0, -6000.0, -7500.0, -9000.0, &
      -12000.0, -15000.0, -7500.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, &
      0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, &
      0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, &
      0.0, 0.0, 0.0, 67.7908728, 135.5817456, 203.3726184, 271.1634912, &
      338.954364, 203.3726184, 0.0, 0.0, -67.7908728, -135.5817456, &
      -203.3726184, -271.1634912, -338.954364, -203.3726184, 0.0], &
      [npts, ndof])
  xout = reshape([0.0, 0.38905, 0.77816, 0.97269, 1.16714, 1.556, &
      1.94484, 0.9726, -1.0e-5, 0.0, -0.388886, -0.77775, -0.97215, &
      -1.16654, -1.55533, -1.9441, -0.97171, 4.0e-5, 0.0, -0.00044, &
      -0.0013, -0.0024, -0.00382, -0.00528, -0.00257, 0.00015, 0.0, &
      0.00144, 0.00306, 0.00446, 0.00567, 0.00688, 0.00451, -2.0e-5, &
      0.0, 0.00122, 0.00259, 0.0029, 0.00314, 0.00338, 0.00356, 0.00477,&
      -1.0e-5, 0.0, 0.00021, 0.00051, 0.00069, 0.00088, 0.0013, 0.00178,&
      0.00058, 3.0e-5, 0.0, 0.27156, 0.54329, 0.81507, 1.08682, 1.35881,&
      0.81553, 1.0e-5, 0.0, -0.27145, -0.54312, -0.81493, -1.0868, &
      -1.35879, -0.81548, 0.0], [npts, ndof])

  ! Compute the calibration gains
  xint = transpose(xin)
  xmeast = transpose(xout)
  c = linear_least_squares(xmeast, xint)
  xmeas = matmul(xout, transpose(c))
  xerr = xmeas - xin

  ! The indices are
  indices = [1, 17, 18, 34]

  ! Compute the crosstalk matrix
  xt = crosstalk(xerr, indices)
end program
```

The least squares fit generates the following matrix of calibration gains.

```
7713.710427        33.5206917
-0.214743728       249.4768498
```

The resulting measured values are then as follows.

```
0                  0
3001.05999         0.220815702
6002.58754         0.479040049
7503.146099        0.514603781
9003.085297        0.532721294
12002.64668        0.509090486
15002.05157        0.470495427
7502.514526        0.981144827
-0.077472309       -0.002492621
0                  0
-2999.74699        0.135900968
-5999.321307       0.294250127
-7498.860677       0.380902151
-8998.322469       0.470046784
-11997.32196       0.658317276
```

```
-14996.16495        0.861552092
-7495.47032         0.353365205
0.30955403          7.48E-03
----------------------------
0                   0
5.708846869         67.74803112
8.183633648         135.5385616
8.808803389         203.3416047
6.96458466          271.1372518
4.81985707          338.9927591
7.512893885         203.4564077
1.157391826         2.46E-03
0                   0
2.008550989         -67.72080332
5.398195074         -135.4965304
7.086130239         -203.3071324
7.306450061         -271.1326527
7.522743936         -338.9881361
7.453379661         -203.4443484
-0.154274205        4.29E-06
```

The crosstalk error matrix is then as follows.

```
0                   0.981144827
8.808803389         0
```

Definition at line 885 of file curvefit_calibration.f90.

5.3 curvefit_core Module Reference

curvefit_core

Data Types

- interface is_monotonic

    Tests to see if an array is montonically increasing or decreasing.
- interface reg_fcn

    Describes a routine for finding the coefficients of a function of one variable.

Functions/Subroutines

- pure logical function is_monotonic_dbl (x)

    Tests to see if an array is montonically increasing or decreasing.
- pure logical function is_monotonic_i32 (x)

    Tests to see if an array is montonically increasing or decreasing.

Variables

- integer, parameter, public dp = real64

    Defines a double-precision (64-bit) floating-point type.
- integer, parameter, public i32 = int32

    Defines a 32-bit signed integer type.
- integer, parameter, public cf_array_size_error = NL_ARRAY_SIZE_ERROR

    An error flag denoting an improperly sized array.
- integer, parameter, public cf_out_of_memory_error = NL_OUT_OF_MEMORY_ERROR

An error denoting that there is insufficient memory available.

- integer, parameter, public cf_no_data_defined_error = 300

  An error denoting that no data has been defined.
- integer, parameter, public cf_invalid_input_error = NL_INVALID_INPUT_ERROR

  An error flag denoting an invalid input.
- integer, parameter, public cf_nonmonotonic_array_error = 301

  An error flag denoting a non-monotonic array was given when a monotonic array was expected.
- integer, parameter, public cf_invalid_operation_error = NL_INVALID_OPERATION_ERROR

  An error resulting from an invalid operation.
- integer, parameter, public cf_convergence_error = NL_CONVERGENCE_ERROR

  An error resulting from a lack of convergence.
- integer, parameter, public cf_tolerance_too_small_error = NL_TOLERANCE_TOO_SMALL_ERROR

  An error indicating the user-requested tolerance is too small to be practical for the problem at hand.
- integer, parameter, public cf_array_index_error = 302

  An error indicating an array index was out of bounds.

5.3.1 Detailed Description

curvefit_core

Purpose

To provide core types and routines for the CURVEFIT library.

5.3.2 Function/Subroutine Documentation

5.3.2.1 pure logical function curvefit_core::is_monotonic_dbl (real(dp), dimension(:), intent(in) *x*)

Tests to see if an array is montonically increasing or decreasing.

Parameters

`in`	*x*	The array to test.

Returns

Returns true if `x` is monotonic; else, false.

Definition at line 105 of file curvefit_core.f90.

5.3.2.2 pure logical function curvefit_core::is_monotonic_i32 (integer(i32), dimension(:), intent(in) *x*) `[private]`

Tests to see if an array is montonically increasing or decreasing.

Parameters

`in`	*x*	The array to test.

Returns

Returns true if x is monotonic; else, false.

Definition at line 139 of file curvefit_core.f90.

5.4 curvefit_interp Module Reference

curvefit_interp

Data Types

- type interp_manager

  Describes an abstract base class allowing for interpolation of X-Y type data sets.
- interface interp_xy

  Defines the signature of a method used to interpolate a single value in an X-Y data set.
- type linear_interp

  Extends the interp_manager class allowing for linear, piecewise interpolation of a data set.
- type polynomial_interp

  Extends the interp_manager class allowing for polynomial interpolation of a data set.
- type spline_interp

  Extends the interp_manager class allowing for cubic spline interpolation of a data set.

Functions/Subroutines

- subroutine im_init (this, x, y, order, err)

  Initializes the specified interp_manager instance.
- integer function im_locate (this, pt, err)

  Attempts to locate the index in the array providing a lower bounds to the specified interpolation point.
- integer(i32) function im_hunt (this, pt, err)

  Attempts to locate the index in the array providing a lower bounds to the specified interpolation point. This method is typically more efficient than locate when the current index does not stray too far from the previous.
- real(dp) function im_perform (this, pt, err)

  Interpolates to obtain the function value at the specified independent variable.
- real(dp) function, dimension(size(pts)) im_perform_array (this, pts, err)

  Interpolates to obtain the function value at the specified independent variables.
- pure integer(i32) function im_get_num_pts (this)

  Gets the number of stored data points.
- pure real(dp) function im_get_x (this, ind)

  Gets the x component of the requested data point.
- pure real(dp) function im_get_y (this, ind)

  Gets the y component of the requested data point.
- real(dp) function li_raw_interp (this, jlo, pt)

  Performs the actual linear interpolation.
- subroutine pi_init (this, x, y, order, err)

  Initializes the specified polynomial_interp instance.
- real(dp) function pi_raw_interp (this, jlo, pt)

  Performs the actual interpolation.
- subroutine penta_solve (a1, a2, a3, a4, a5, b, x)

Solves a pentadiagonal system of linear equations. A pentadiagonal matrix is all zeros with the exception of the diagonal, and the two immediate sub and super-diagonals. The entries of row I are stored as follows: A(I,I-2) -> A1(I) A(I,I-1) -> A2(I) A(I,I) -> A3(I) A(I,I+1) -> A4(I) A(I,I+2) -> A5(I)

- real(dp) function si_raw_interp (this, jlo, pt)

  Performs the actual interpolation.
- subroutine si_second_deriv (this, ibcbeg, ybcbeg, ibcend, ybcend, err)

  Computes the second derivative terms for the cubic-spline model.
- subroutine si_init_1 (this, x, y, order, err)

  Initializes the specified spline_interp instance. The end points are considered free such that the interpolant is quadratic over both the initial and final intervals.
- subroutine si_init_2 (this, x, y, ibcbeg, ybcbeg, ibcend, ybcend, err)

  Initializes the specified spline_interp instance.
- real(dp) function si_diff1 (this, pt, err)

  Interpolates to obtain the first derivative value at the specified independent variable.
- real(dp) function, dimension(size(pts)) si_diff1_array (this, pts, err)

  Interpolates to obtain the first derivative value at the specified independent variables.
- real(dp) function si_diff2 (this, pt, err)

  Interpolates to obtain the second derivative value at the specified independent variable.
- real(dp) function, dimension(size(pts)) si_diff2_array (this, pts, err)

  Interpolates to obtain the second derivative value at the specified independent variables.

Variables

- integer(i32), parameter, public spline_quadratic_over_interval = 1000

  Indicates that the spline is quadratic over the interval under consideration (beginning or ending interval). This is equivalent to allowing a "natural" boundary condition at either the initial or final point.
- integer(i32), parameter, public spline_known_first_derivative = 1001

  Indicates a known first derivative at either the beginning or ending point.
- integer(i32), parameter, public spline_known_second_derivative = 1002

  Indicates a known second derivative at either the beginning or ending point.
- integer(i32), parameter, public spline_continuous_third_derivative = 1003

  Indicates a continuous third derivative at either the beginning or ending point.

5.4.1 Detailed Description

curvefit_interp

Purpose

To provide interpolation routines for X-Y data sets.

5.4.2 Function/Subroutine Documentation

5.4.2.1 pure integer(i32) function curvefit_interp::im_get_num_pts (class(interp_manager), intent(in) *this*) [private]

Gets the number of stored data points.

Parameters

in	*this*	The interp_manager object.

Returns

The number of data points.

Definition at line 506 of file curvefit_interp.f90.

5.4.2.2 pure real(dp) function curvefit_interp::im_get_x (class(interp_manager), intent(in) *this*, integer(i32), intent(in) *ind*) [private]

Gets the x component of the requested data point.

Parameters

in	*this*	The interp_manager object.
in	*ind*	The one-based index of the data point to retrieve.

Returns

The x component of the requested data point.

Definition at line 520 of file curvefit_interp.f90.

5.4.2.3 pure real(dp) function curvefit_interp::im_get_y (class(interp_manager), intent(in) *this*, integer(i32), intent(in) *ind*) [private]

Gets the y component of the requested data point.

Parameters

in	*this*	The interp_manager object.
in	*ind*	The one-based index of the data point to retrieve.

Returns

The y component of the requested data point.

Definition at line 535 of file curvefit_interp.f90.

5.4.2.4 integer(i32) function curvefit_interp::im_hunt (class(interp_manager), intent(inout) *this*, real(dp), intent(in) *pt*, class(errors), intent(inout), optional, target *err*) [private]

Attempts to locate the index in the array providing a lower bounds to the specified interpolation point. This method is typically more efficient than locate when the current index does not stray too far from the previous.

Parameters

in,out	*this*	The interp_manager instance.
in	*pt*	The interpolation point.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has yet been defined.

Returns

The array index below `pt`.

Definition at line 337 of file curvefit_interp.f90.

5.4.2.5 subroutine curvefit_interp::im_init (class(interp_manager), intent(inout) *this*, real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i32), intent(in), optional *order*, class(errors), intent(inout), optional, target *err*)

Initializes the specified interp_manager instance.

Parameters

in,out	*this*	The interp_manager instance.
in	*x*	An N-element array containing the independent variable data. The data in this array must be either monotonically increasing or decreasing.
in	*y*	An N-element array containing the dependent variable data.
in	*order*	The order of the interpolating polynomial. Notice, this parameter is optional; however, if not specified, a default of 1 is used.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are not the same size. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if `x` is not monotonically increasing or decreasing.

Definition at line 184 of file curvefit_interp.f90.

5.4.2.6 integer function curvefit_interp::im_locate (class(interp_manager), intent(inout) *this*, real(dp), intent(in) *pt*, class(errors), intent(inout), optional, target *err*) [private]

Attempts to locate the index in the array providing a lower bounds to the specified interpolation point.

Parameters

in,out	*this*	The interp_manager instance.

Parameters

in	*pt*	The interpolation point.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has yet been defined.

Returns

The array index below `pt`.

Definition at line 262 of file curvefit_interp.f90.

5.4.2.7 real(dp) function curvefit_interp::im_perform (class(interp_manager), intent(inout) *this*, real(dp), intent(in) *pt*, class(errors), intent(inout), optional, target *err*) `[private]`

Interpolates to obtain the function value at the specified independent variable.

Parameters

in,out	*this*	The interp_manager instance.
in	*pt*	The independent variable value to interpolate.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has yet been defined.

Returns

The interpolated value.

Definition at line 445 of file curvefit_interp.f90.

5.4.2.8 real(dp) function, dimension(size(pts)) curvefit_interp::im_perform_array (class(interp_manager), intent(inout) *this*, real(dp), dimension(:), intent(in) *pts*, class(errors), intent(inout), optional, target *err*) `[private]`

Interpolates to obtain the function value at the specified independent variables.

Parameters

in,out	*this*	The interp_manager instance.
in	*pts*	An M-element array containing the independent variable values to interpolate.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has yet been defined.

Returns

An M-element array containing the interpolated values.

Definition at line 479 of file curvefit_interp.f90.

5.4.2.9 real(dp) function curvefit_interp::li_raw_interp (class(linear_interp), intent(inout) *this*, integer(i32), intent(in) *jlo*, real(dp), intent(in) *pt*) [private]

Performs the actual linear interpolation.

Parameters

in,out	*this*	The linear_interp_mgr instance.
in	*jlo*	The array index below which `pt` is found in x.
in	*pt*	The independent variable value to interpolate.

Returns

The interpolated value.

Definition at line 554 of file curvefit_interp.f90.

5.4.2.10 subroutine curvefit_interp::penta_solve (real(dp), dimension(:), intent(in) *a1*, real(dp), dimension(:), intent(inout) *a2*, real(dp), dimension(:), intent(inout) *a3*, real(dp), dimension(:), intent(inout) *a4*, real(dp), dimension(:), intent(in) *a5*, real(dp), dimension(:), intent(inout) *b*, real(dp), dimension(:), intent(out) *x*) [private]

Solves a pentadiagonal system of linear equations. A pentadiagonal matrix is all zeros with the exception of the diagonal, and the two immediate sub and super-diagonals. The entries of row I are stored as follows: A(I,I-2) -> A1(I) A(I,I-1) -> A2(I) A(I,I) -> A3(I) A(I,I+1) -> A4(I) A(I,I+2) -> A5(I)

Parameters

in	*a1*	An N-element array as defined above.
in,out	*a2*	An N-element array as defined above. This array is overwritten by this routine during the solution process.
in,out	*a3*	An N-element array as defined above. This array is overwritten by this routine during the solution process.
in,out	*a4*	An N-element array as defined above. This array is overwritten by this routine during the solution process.
in	*a5*	An N-element array as defined above.
in,out	*b*	An N-element array containing the right-hand-side. This array is overwritten by this routine during the solution process.
out	*x*	An N-element array that, on output, contains the solution to the linear system.

- `Spline Library`

Definition at line 732 of file curvefit_interp.f90.

5.4.2.11 subroutine curvefit_interp::pi_init (class(polynomial_interp), intent(inout) *this*, real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i32), intent(in), optional *order*, class(errors), intent(inout), optional, target *err*) `[private]`

Initializes the specified polynomial_interp instance.

Parameters

`in,out`	*this*	The polynomial_interp instance.
`in`	*x*	An N-element array containing the independent variable data. The data in this array must be either monotonically increasing or decreasing.
`in`	*y*	An N-element array containing the dependent variable data.
`in`	*order*	The order of the interpolating polynomial. Notice, this parameter is optional; however, if not specified, a default of 1 is used.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are not the same size. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_INVALID_INPUT_ERROR: Occurs if `order` is less than 1. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if `x` is not monotonically increasing or decreasing.

Definition at line 595 of file curvefit_interp.f90.

5.4.2.12 real(dp) function curvefit_interp::pi_raw_interp (class(polynomial_interp), intent(inout) *this*, integer(i32), intent(in) *jlo*, real(dp), intent(in) *pt*) `[private]`

Performs the actual interpolation.

Parameters

`in,out`	*this*	The polynomial_interp instance.
`in`	*jlo*	The array index below which `pt` is found in x.
`in`	*pt*	The independent variable value to interpolate.

Returns

The interpolated value.

Definition at line 649 of file curvefit_interp.f90.

5.4.2.13 real(dp) function curvefit_interp::si_diff1 (class(spline_interp), intent(inout) *this*, real(dp), intent(in) *pt*, class(errors), intent(inout), optional, target *err*) `[private]`

Interpolates to obtain the first derivative value at the specified independent variable.

Parameters

in,out	*this*	The interp_manager instance.
in	*pt*	The independent variable value to interpolate.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has yet been defined.

Returns

The interpolated value.

Definition at line 1135 of file curvefit_interp.f90.

5.4.2.14 real(dp) function, dimension(size(pts)) curvefit_interp::si_diff1_array (class(spline_interp), intent(inout) *this*, real(dp), dimension(:), intent(in) *pts*, class(errors), intent(inout), optional, target *err*) [private]

Interpolates to obtain the first derivative value at the specified independent variables.

Parameters

in,out	*this*	The interp_manager instance.
in	*pts*	An M-element array containing the independent variable values to interpolate.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has yet been defined.

Returns

An M-element array containing the interpolated values.

Definition at line 1181 of file curvefit_interp.f90.

5.4.2.15 real(dp) function curvefit_interp::si_diff2 (class(spline_interp), intent(inout) *this*, real(dp), intent(in) *pt*, class(errors), intent(inout), optional, target *err*) [private]

Interpolates to obtain the second derivative value at the specified independent variable.

Parameters

in,out	*this*	The interp_manager instance.
in	*pt*	The independent variable value to interpolate.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has yet been defined.

Returns

The interpolated value.

Definition at line 1228 of file curvefit_interp.f90.

5.4.2.16 real(dp) function, dimension(size(pts)) curvefit_interp::si_diff2_array (class(spline_interp), intent(inout) *this*, real(dp), dimension(:), intent(in) *pts*, class(errors), intent(inout), optional, target *err*) `[private]`

Interpolates to obtain the second derivative value at the specified independent variables.

Parameters

`in,out`	*this*	The interp_manager instance.
`in`	*pts*	An M-element array containing the independent variable values to interpolate.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has yet been defined.

Returns

An M-element array containing the interpolated values.

Definition at line 1266 of file curvefit_interp.f90.

5.4.2.17 subroutine curvefit_interp::si_init_1 (class(spline_interp), intent(inout) *this*, real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i32), intent(in), optional *order*, class(errors), intent(inout), optional, target *err*) `[private]`

Initializes the specified spline_interp instance. The end points are considered free such that the interpolant is quadratic over both the initial and final intervals.

Parameters

`in,out`	*this*	The spline_interp instance.
`in`	*x*	An N-element array containing the independent variable data. The data in this array must be either monotonically increasing or decreasing.
`in`	*y*	An N-element array containing the dependent variable data.
`in`	*order*	The order of the interpolating polynomial. This parameter is ignored as the spline is a cubic approximation.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are not the same size. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if `x` is not monotonically increasing or decreasing.

Definition at line 989 of file curvefit_interp.f90.

5.4.2.18 subroutine curvefit_interp::si_init_2 (class(spline_interp), intent(inout) *this*, real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, integer(i32), intent(in), optional *ibcbeg*, real(dp), intent(in), optional *ybcbeg*, integer(i32), intent(in), optional *ibcend*, real(dp), intent(in), optional *ybcend*, class(errors), intent(inout), optional, target *err*) `[private]`

Initializes the specified spline_interp instance.

Parameters

`in,out`	*this*	The spline_interp instance.
`in`	*x*	An N-element array containing the independent variable data. The data in this array must be either monotonically increasing or decreasing.
`in`	*y*	An N-element array containing the dependent variable data.
`in`	*ibcbeg*	An optional input that defines the nature of the boundary condition at the beginning of the spline. If no parameter, or an invalid parameter, is specified, the default natural condition (SPLINE_QUADRATIC_OVER_INTERVAL) is used. • SPLINE_QUADRATIC_OVER_INTERVAL: The spline is quadratic over its initial interval. No value is required for `ybcbeg`. • SPLINE_KNOWN_FIRST_DERIVATIVE: The spline's first derivative at its initial point is provided in `ybcbeg`. • SPLINE_KNOWN_SECOND_DERIVATIVE: The spline's second derivative at its initial point is provided in `ybcbeg`. • SPLINE_CONTINUOUS_THIRD_DERIVATIVE: The third derivative is continuous at x(2). No value is required for `ybcbeg`.
`in`	*ybcbeg*	If needed, the value of the initial point boundary condition. If needed, but not supplied, a default value of zero will be used.
`in`	*ibcend*	An optional input that defines the nature of the boundary condition at the end of the spline. If no parameter, or an invalid parameter, is specified, the default natural condition (SPLINE_QUADRATIC_OVER_INTERVAL) is used. • SPLINE_QUADRATIC_OVER_INTERVAL: The spline is quadratic over its final interval. No value is required for `ybcend`. • SPLINE_KNOWN_FIRST_DERIVATIVE: The spline's first derivative at its initial point is provided in `ybcend`. • SPLINE_KNOWN_SECOND_DERIVATIVE: The spline's second derivative at its initial point is provided in `ybcend`. • SPLINE_CONTINUOUS_THIRD_DERIVATIVE: The third derivative is continuous at x(n-1). No value is required for `ybcend`.
`in`	*ybcend*	If needed, the value of the final point boundary condition. If needed, but not supplied, a default value of zero will be used.

Parameters

out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are not the same size. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_INVALID_INPUT_ERROR: Occurs if `order` is less than 1. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if `x` is not monotonically increasing or decreasing.

Definition at line 1070 of file curvefit_interp.f90.

5.4.2.19 real(dp) function curvefit_interp::si_raw_interp (class(**spline_interp**), intent(inout) *this*, integer(i32), intent(in) *jlo*, real(dp), intent(in) *pt*) `[private]`

Performs the actual interpolation.

Parameters

in,out	*this*	The spline_interp instance.
in	*jlo*	The array index below which `pt` is found in x.
in	*pt*	The independent variable value to interpolate.

Returns

The interpolated value.

Definition at line 774 of file curvefit_interp.f90.

5.4.2.20 subroutine curvefit_interp::si_second_deriv (class(**spline_interp**), intent(inout) *this*, integer(i32), intent(in) *ibcbeg*, real(dp), intent(in) *ybcbeg*, integer(i32), intent(in) *ibcend*, real(dp), intent(in) *ybcend*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the second derivative terms for the cubic-spline model.

Parameters

in,out	*this*	The spline_interp_mgr instance.
in	*ibcbeg*	Defines the nature of the boundary condition at the beginning of the spline. • SPLINE_QUADRATIC_OVER_INTERVAL: The spline is quadratic over its initial interval. • SPLINE_KNOWN_FIRST_DERIVATIVE: The spline's first derivative at its initial point is provided in `ybcbeg`. • SPLINE_KNOWN_SECOND_DERIVATIVE: The spline's second derivative at its initial point is provided in `ybcbeg`. • SPLINE_CONTINUOUS_THIRD_DERIVATIVE: The third derivative is continuous at x(2).

Parameters

in	*ybcbeg*	If needed, the value of the initial point boundary condition.
in	*ibcend*	Defines the nature of the boundary condition at the end of the spline. • SPLINE_QUADRATIC_OVER_INTERVAL: The spline is quadratic over its final interval. • SPLINE_KNOWN_FIRST_DERIVATIVE: The spline's first derivative at its initial point is provided in `ybcend`. • SPLINE_KNOWN_SECOND_DERIVATIVE: The spline's second derivative at its initial point is provided in `ybcend`. • SPLINE_CONTINUOUS_THIRD_DERIVATIVE: The third derivative is continuous at x(n-1).
in	*ybcend*	If needed, the value of the final point boundary condition.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Remarks

This code is a slight modification of the SPLINE_CUBIC_SET routine from the `SPLINE` library.

Definition at line 842 of file curvefit_interp.f90.

5.5 curvefit_regression Module Reference

curvefit_regression

Data Types

- interface linear_least_squares

  Employs a least squares fit to determine the coefficient A in the linear system: Y = A ∗ X, where A can either be a scalar, or a matrix.
- type lowess_smoothing

  Defines a type for computing a smoothing of an X-Y data set using a robust locally weighted scatterplot smoothing (LOWESS) algorithm.
- interface moving_average

  Applies a moving average to smooth a data set.
- type nonlinear_regression

  A type for supporting nonlinear regression calculations.

Functions/Subroutines

- subroutine moving_average_1 (x, npts, err)

  Applies a moving average to smooth a data set.
- subroutine lowest (x, y, xs, ys, nleft, nright, w, userw, rw, ok)

  A support routine for the LOWESS library used to compute the smoothing of a desired value from a data set.
- subroutine lowess (x, y, f, nsteps, delta, ys, rw, res)

  Computes a smoothing of an X-Y data set using a robust locally weighted scatterplot smoothing (LOWESS) algorithm. Fitted values are computed at each of the supplied x values.
- subroutine ls_init (this, x, y, srt, err)

  Initializes the lowess_smoothing object.
- real(dp) function, dimension(:), allocatable ls_smooth (this, f, err)

  Performs the actual smoothing operation.
- pure integer(i32) function ls_get_num_pts (this)

  Gets the number of stored data points.
- pure real(dp) function ls_get_x (this, ind)

  Gets the x component of the requested data point.
- pure real(dp) function ls_get_y (this, ind)

  Gets the y component of the requested data point.
- subroutine ls_get_residual (this, x)

  Gets the residuals from each data point.
- subroutine nr_init (this, x, y, fcn, ncoeff, err)

  Initializes the nonlinear_regression object.
- subroutine nr_fcn (this, x, f)

  Computes the residual between the supplied data set, and the function value given a set of coefficients.
- pure logical function nr_is_fcn_defined (this)

  Determines if the function has been defined.
- pure integer(i32) function nr_get_eqn_count (this)

  Gets the number of equations required to solve the regression problem.
- pure integer(i32) function nr_get_var_count (this)

  Gets the number of variables (coefficients).
- subroutine nr_solve (this, c, res, ib, err)

  Computes the solution to the nonlinear regression problem using the Levenberg-Marquardt method.
- pure integer(i32) function nr_get_max_eval (this)

  Gets the maximum number of function evaluations allowed during a single solve.
- subroutine nr_set_max_eval (this, n)

  Sets the maximum number of function evaluations allowed during a single solve.
- pure real(dp) function nr_get_fcn_tol (this)

  Gets the convergence on function value tolerance.
- subroutine nr_set_fcn_tol (this, x)

  Sets the convergence on function value tolerance.
- pure real(dp) function nr_get_var_tol (this)

  Gets the convergence on change in variable tolerance.
- subroutine nr_set_var_tol (this, x)

  Sets the convergence on change in variable tolerance.
- pure real(dp) function nr_get_grad_tol (this)

  Gets the convergence on slope of the gradient vector tolerance.
- subroutine nr_set_grad_tol (this, x)

  Sets the convergence on slope of the gradient vector tolerance.
- pure logical function nr_get_print_status (this)

  Gets a logical value determining if iteration status should be printed.

- subroutine nr_set_print_status (this, x)

  Sets a logical value determining if iteration status should be printed.
- pure integer(i32) function nr_get_num_pts (this)

  Gets the number of stored data points.
- pure real(dp) function nr_get_x (this, ind)

  Gets the x component of the requested data point.
- pure real(dp) function nr_get_y (this, ind)

  Gets the y component of the requested data point.
- real(dp) function linear_least_squares_1var (x, y, err)

  Employs a least squares fit to determine the coefficient A in the linear system: $Y = A * X$.
- real(dp) function, dimension(size(y, 1), size(x, 1)) linear_least_squares_nvar (x, y, thrsh, err)

  Employs a least squares fit to determine the coefficient A in the linear system: $Y = A * X$.

5.5.1 Detailed Description

curvefit_regression

Purpose

To provide routines for perforing regression operations, and other data smoothing operations on sets of numerical data.

5.5.2 Function/Subroutine Documentation

5.5.2.1 real(dp) function curvefit_regression::linear_least_squares_1var (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(inout) *y*, class(errors), intent(inout), optional, target *err*) `[private]`

Employs a least squares fit to determine the coefficient A in the linear system: $Y = A * X$.

Parameters

`in`	*x*	An N-element array containing the independent variable data.
`in,out`	*y*	An N-element array containing the dependent variable data corresponding to `x`. On output, the contents of this array are overwritten as it is used for storage purposes by the algorithm.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are different sizes.

Returns

The scalar coefficient A.

Definition at line 1081 of file curvefit_regression.f90.

5.5.2.2 real(dp) function, dimension(size(y,1), size(x,1)) curvefit_regression::linear_least_squares_nvar (real(dp), dimension(:,:), intent(inout) *x*, real(dp), dimension(:,:), intent(in) *y*, real(dp), intent(in), optional *thrsh*, class(errors), intent(inout), optional, target *err*) `[private]`

Employs a least squares fit to determine the coefficient A in the linear system: Y = A ∗ X.

Parameters

`in,out`	*x*	An M-by-P matrix containing the P data points of the M independent variables.
`in`	*y*	An N-by-P matrix containing the P data points of the N dependent variables.
`in`	*thrsh*	An optional threshold value that defines a lower cutoff for singular values. Any singular values falling below this value will have their reciprocal replaced with zero.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if any of the matrix dimensions are not compatiable. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Returns

An N-by-M matrix relating Y to X such that: Y = A ∗ X.

Remarks

The algorithm attempts to compute the coefficient matrix A as follows. Y ∗ X∗∗T = A ∗ X ∗ X∗∗T Y ∗ X∗∗T ∗ INV(X ∗ X∗∗T) = A This does require that X ∗ X∗∗T does not result in a singular matrix. To handle the situation where X ∗ X∗∗T is singular, the Moore-Penrose pseudo-inverse, computed by means of singular value decomposition, is utilized to still arrive at a solution that, at minimum, has a minimum Euclidean norm of its residual. Let: PINV(X) = X∗∗T ∗ INV(X ∗ X∗∗T), Then: A = Y ∗ PINV(X)

Definition at line 1150 of file curvefit_regression.f90.

5.5.2.3 subroutine curvefit_regression::lowess (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, real(dp), intent(in) *f*, integer(i32), intent(in) *nsteps*, real(dp), intent(in) *delta*, real(dp), dimension(:), intent(out) *ys*, real(dp), dimension(:), intent(out) *rw*, real(dp), dimension(:), intent(out) *res*) `[private]`

Computes a smoothing of an X-Y data set using a robust locally weighted scatterplot smoothing (LOWESS) algorithm. Fitted values are computed at each of the supplied x values.

Parameters

`in`	*x*	An N-element containing the independent variable values of the data set. This array must be in a monotonically increasing order.
`in`	*y*	An N-element array of the dependent variables corresponding to `x`.
`in`	*f*	Specifies the amount of smoothing. More specifically, this value is the fraction of points used to compute each value. As this value increases, the output becomes smoother. Choosing a value in the range of 0.2 to 0.8 usually results in a good fit. As such, a reasonable starting point, in the absence of better information, is a value of 0.5.
`in`	*nsteps*	The number of iterations in the robust fit. If set to zero, a nonrobust fit is returned. Seeting this parameter equal to 2 should serve most purposes.

Parameters

in	*delta*	A nonnegative parameter which may be used to save computations. If N is less than 100, set delta equal to 0.0. If N is larger than 100, set delta = range(x) / k, where k determines the interpolation window used by the linear weighted regression computations.
out	*ys*	An N-element array that, on output, contains the fitted values.
out	*rw*	An N-element array that, on output, contains the robustness weights given to each data point.
out	*rs*	An N-element array that, on output, contains the residual `y - ys`.

Remarks

This routines is an implementation of the LOWESS routine from the LOWESS library. A link to this library, along with a basic description of the algorithm is available `here`. For a detailed understanding of the algorithm, see the `paper` by William Cleveland.

Definition at line 362 of file curvefit_regression.f90.

5.5.2.4 subroutine curvefit_regression::lowest (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, real(dp), intent(in) *xs*, real(dp), intent(out) *ys*, integer(i32), intent(in) *nleft*, integer(i32), intent(in) *nright*, real(dp), dimension(:), intent(out) *w*, logical, intent(in) *userw*, real(dp), dimension(:), intent(in) *rw*, logical, intent(out) *ok*) `[private]`

A support routine for the LOWESS library used to compute the smoothing of a desired value from a data set.

Parameters

in	*x*	An N-element containing the independent variable values of the data set. This array must be in a monotonically increasing order.
in	*y*	An N-element array of the dependent variables corresponding to `x`.
in	*xs*	The value of the independent variable at which the smoothing is computed.
out	*ys*	The fitted value.
in	*nleft*	The index of the first point which should be considered in computing the fit.
in	*nright*	The index of the last point which should be considered in computing the fit.
out	*w*	An N-element array that, on output, contains the weights for `y` in the expression for `ys`.
in	*userw*	If true, a robust fit is carried out using the weights in `rw`. If false, the values in `rw` are not used.
in	*rw*	An N-element array containing the robustness weights.
out	*ok*	Returns true if the calculations were performed; however, returns false if the weights are all zero-valued.

Remarks

This routines is an implementation of the LOWEST routine from the LOWESS library. A link to this library, along with a basic description of the algorithm is available `here`. For a detailed understanding of the algorithm, see the `paper` by William Cleveland.

Definition at line 250 of file curvefit_regression.f90.

5.5.2.5 pure integer(i32) function curvefit_regression::ls_get_num_pts (class(**lowess_smoothing**), intent(in) *this*) `[private]`

Gets the number of stored data points.

Parameters

in	*this*	The lowess_smoothing object.

Returns

The number of data points.

Definition at line 613 of file curvefit_regression.f90.

5.5.2.6 subroutine curvefit_regression::ls_get_residual (class(lowess_smoothing), intent(in) *this*, real(dp), dimension(:), intent(out) *x*) [private]

Gets the residuals from each data point.

Parameters

in	*this*	The lowess_smoothing object.
out	*x*	An N-element array where the residual data should be written.

Definition at line 665 of file curvefit_regression.f90.

5.5.2.7 pure real(dp) function curvefit_regression::ls_get_x (class(lowess_smoothing), intent(in) *this*, integer(i32), intent(in) *ind*) [private]

Gets the x component of the requested data point.

Parameters

in	*this*	The lowess_smoothing object.
in	*ind*	The one-based index of the data point to retrieve.

Returns

The x component of the requested data point.

Definition at line 630 of file curvefit_regression.f90.

5.5.2.8 pure real(dp) function curvefit_regression::ls_get_y (class(lowess_smoothing), intent(in) *this*, integer(i32), intent(in) *ind*) [private]

Gets the y component of the requested data point.

Parameters

in	*this*	The lowess_smoothing object.
in	*ind*	The one-based index of the data point to retrieve.

Returns

The y component of the requested data point.

Definition at line 648 of file curvefit_regression.f90.

5.5.2.9 subroutine curvefit_regression::ls_init (class(**lowess_smoothing**), intent(inout) *this*, real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, logical, intent(in), optional *srt*, class(errors), intent(inout), optional, target *err*) `[private]`

Initializes the lowess_smoothing object.

Parameters

`in,out`	*this*	The lowess_smoothing object.
`in`	*x*	An N-element containing the independent variable values of the data set. This array must be in a monotonically increasing order. The routine is capable of sorting the array into ascending order, dependent upon the value of `srt`. If sorting is performed, this routine will also shuffle `y` to match.
`in`	*y*	An N-element array of the dependent variables corresponding to `x`.
`in`	*srt*	An optional flag determining if `x` should be sorted. The default is to sort (true).
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are not the same size. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Definition at line 477 of file curvefit_regression.f90.

5.5.2.10 real(dp) function, dimension(:), allocatable curvefit_regression::ls_smooth (class(**lowess_smoothing**), intent(inout) *this*, real(dp), intent(in) *f*, class(errors), intent(inout), optional, target *err*) `[private]`

Performs the actual smoothing operation.

Parameters

`in,out`	*this*	The lowess_smoothing object.
`in`	*f*	Specifies the amount of smoothing. More specifically, this value is the fraction of points used to compute each value. As this value increases, the output becomes smoother. Choosing a value in the range of 0.2 to 0.8 usually results in a good fit. As such, a reasonable starting point, in the absence of better information, is a value of 0.5.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NO_DATA_DEFINED_ERROR: Occurs if no data has been defined. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Returns

The smoothed data points.

Definition at line 569 of file curvefit_regression.f90.

5.5.2.11 subroutine curvefit_regression::moving_average_1 (real(dp), dimension(:), intent(inout) *x*, integer(i32), intent(in) *npts*, class(errors), intent(inout), optional, target *err*) `[private]`

Applies a moving average to smooth a data set.

Parameters

`in,out`	*x*	On input, the signal to smooth. On output, the smoothed signal.
`in`	*npts*	The size of the averaging window. This value must be at least 2, but no more than the number of elements in `x`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `npts` is less than 2, or greater than the length of `x`. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Definition at line 164 of file curvefit_regression.f90.

5.5.2.12 subroutine curvefit_regression::nr_fcn (class(**nonlinear_regression**), intent(in) *this*, real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(out) *f*) `[private]`

Computes the residual between the supplied data set, and the function value given a set of coefficients.

Parameters

`in`	*this*	The nonlinear_regression object.
`in`	*x*	An N-element array containing the N coefficients.
`out`	*f*	An M-element array that, on output, contains the residual at each of the M data points.

Definition at line 766 of file curvefit_regression.f90.

5.5.2.13 pure integer(i32) function curvefit_regression::nr_get_eqn_count (class(**nonlinear_regression**), intent(in) *this*) `[private]`

Gets the number of equations required to solve the regression problem.

Parameters

`in`	*this*	The nonlinear_regression object.

Returns

The number of equations.

Definition at line 801 of file curvefit_regression.f90.

5.5.2.14 pure real(dp) function curvefit_regression::nr_get_fcn_tol (class(nonlinear_regression), intent(in) *this*) [private]

Gets the convergence on function value tolerance.

Parameters

in	*this*	The nonlinear_regression object.

Returns

The tolerance value.

Definition at line 925 of file curvefit_regression.f90.

5.5.2.15 pure real(dp) function curvefit_regression::nr_get_grad_tol (class(nonlinear_regression), intent(in) *this*) [private]

Gets the convergence on slope of the gradient vector tolerance.

Parameters

in	*this*	The nonlinear_regression object.

Returns

The tolerance value.

Definition at line 969 of file curvefit_regression.f90.

5.5.2.16 pure integer(i32) function curvefit_regression::nr_get_max_eval (class(nonlinear_regression), intent(in) *this*) [private]

Gets the maximum number of function evaluations allowed during a single solve.

Parameters

in	*this*	The nonlinear_regression object.

Returns

The maximum number of function evaluations.

Definition at line 902 of file curvefit_regression.f90.

5.5.2.17 pure integer(i32) function curvefit_regression::nr_get_num_pts (class(nonlinear_regression), intent(in) *this*) [private]

Gets the number of stored data points.

Parameters

in	*this*	The nonlinear_regression object.

Returns

The number of data points.

Definition at line 1016 of file curvefit_regression.f90.

5.5.2.18 pure logical function curvefit_regression::nr_get_print_status (class(nonlinear_regression), intent(in) *this*) [private]

Gets a logical value determining if iteration status should be printed.

Parameters

in	*this*	The nonlinear_regression object.

Returns

True if the iteration status should be printed; else, false.

Definition at line 992 of file curvefit_regression.f90.

5.5.2.19 pure integer(i32) function curvefit_regression::nr_get_var_count (class(nonlinear_regression), intent(in) *this*) [private]

Gets the number of variables (coefficients).

Parameters

in	*this*	The nonlinear_regression object.

Returns

The number of variables.

Definition at line 817 of file curvefit_regression.f90.

5.5.2.20 pure real(dp) function curvefit_regression::nr_get_var_tol (class(nonlinear_regression), intent(in) *this*) [private]

Gets the convergence on change in variable tolerance.

Parameters

`in`	*this*	The nonlinear_regression object.

Returns

The tolerance value.

Definition at line 947 of file curvefit_regression.f90.

5.5.2.21 pure real(dp) function curvefit_regression::nr_get_x (class(nonlinear_regression), intent(in) *this*, integer(i32), intent(in) *ind*) `[private]`

Gets the x component of the requested data point.

Parameters

`in`	*this*	The nonlinear_regression object.
`in`	*ind*	The one-based index of the data point to retrieve.

Returns

The x component of the requested data point.

Definition at line 1033 of file curvefit_regression.f90.

5.5.2.22 pure real(dp) function curvefit_regression::nr_get_y (class(nonlinear_regression), intent(in) *this*, integer(i32), intent(in) *ind*) `[private]`

Gets the y component of the requested data point.

Parameters

`in`	*this*	The nonlinear_regression object.
`in`	*ind*	The one-based index of the data point to retrieve.

Returns

The y component of the requested data point.

Definition at line 1051 of file curvefit_regression.f90.

5.5.2.23 subroutine curvefit_regression::nr_init (class(nonlinear_regression), intent(inout) *this*, real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, procedure(reg_fcn), intent(in), pointer *fcn*, integer(i32), intent(in) *ncoeff*, class(errors), intent(inout), optional, target *err*) `[private]`

Initializes the nonlinear_regression object.

Parameters

`in,out`	*this*	The nonlinear_regression object.
`in`	*x*	An N-element containing the independent variable values of the data set.
`in`	*y*	An N-element array of the dependent variables corresponding to `x`.
`in`	*fcn*	A pointer to the function whose coefficients are to be determined.
`in`	*ncoeff*	The number of coefficients in the function defined in `fcn`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are not the same size. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_INVALID_INPUT_ERROR: Occurs if `ncoeff` is less than or equal to zero.

Definition at line 702 of file curvefit_regression.f90.

5.5.2.24 pure logical function curvefit_regression::nr_is_fcn_defined (class(nonlinear_regression), intent(in) *this*) `[private]`

Determines if the function has been defined.

Parameters

`in`	*this*	The nonlinear_regression object.

Returns

Returns true if the function has been defined; else, false.

Definition at line 788 of file curvefit_regression.f90.

5.5.2.25 subroutine curvefit_regression::nr_set_fcn_tol (class(nonlinear_regression), intent(inout) *this,* real(dp), intent(in) *x*) `[private]`

Sets the convergence on function value tolerance.

Parameters

`in,out`	*this*	The nonlinear_regression object.
`in`	*x*	The tolerance value.

Definition at line 936 of file curvefit_regression.f90.

5.5.2.26 subroutine curvefit_regression::nr_set_grad_tol (class(nonlinear_regression), intent(inout) *this,* real(dp), intent(in) *x*) `[private]`

Sets the convergence on slope of the gradient vector tolerance.

Parameters

in	*this*	The nonlinear_regression object.

Returns

The tolerance value.

Definition at line 980 of file curvefit_regression.f90.

5.5.2.27 subroutine curvefit_regression::nr_set_max_eval (class(nonlinear_regression), intent(inout) *this*, integer(i32), intent(in) *n*) [private]

Sets the maximum number of function evaluations allowed during a single solve.

Parameters

in,out	*this*	The nonlinear_regression object.
in	*n*	The maximum number of function evaluations.

Definition at line 914 of file curvefit_regression.f90.

5.5.2.28 subroutine curvefit_regression::nr_set_print_status (class(nonlinear_regression), intent(inout) *this*, logical, intent(in) *x*) [private]

Sets a logical value determining if iteration status should be printed.

Parameters

in,out	*this*	The nonlinear_regression object.
in	*x*	True if the iteration status should be printed; else, false.

Definition at line 1004 of file curvefit_regression.f90.

5.5.2.29 subroutine curvefit_regression::nr_set_var_tol (class(nonlinear_regression), intent(inout) *this*, real(dp), intent(in) *x*) [private]

Sets the convergence on change in variable tolerance.

Parameters

in,out	*this*	The nonlinear_regression object.
in	*x*	The tolerance value.

Definition at line 958 of file curvefit_regression.f90.

5.5.2.30 subroutine curvefit_regression::nr_solve (class(nonlinear_regression), intent(inout) *this,* real(dp), dimension(:), intent(inout) *c,* real(dp), dimension(:), intent(out), optional, target *res,* type(iteration_behavior), intent(out), optional *ib,* class(errors), intent(inout), optional, target *err*) [private]

Computes the solution to the nonlinear regression problem using the Levenberg-Marquardt method.

Parameters

in	*this*	The nonlinear_regression object.
in,out	*c*	On input, an array containing initial estimates of the coefficients. On output, the comptued coefficient values.
out	*res*	An optional output array, whose size corresponds to the number of data points, that can be used to retrieve the residual error at each data point.
out	*ib*	An optional output, that if provided, allows the caller to obtain iteration performance statistics.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_OPERATION_ERROR: Occurs if no equations have been defined. • CF_INVALID_INPUT_ERROR: Occurs if the number of equations is less than than the number of variables. • CF_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized correctly. • CF_CONVERGENCE_ERROR: Occurs if the line search cannot converge within the allowed number of iterations. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available. • CF_TOLERANCE_TOO_SMALL_ERROR: Occurs if the requested tolerance is to small to be practical for the problem at hand.

Definition at line 851 of file curvefit_regression.f90.

5.6 curvefit_statistics Module Reference

curvefit_statistics

Data Types

- interface confidence_interval

    Computes the confidence interval based upon a standard normal distribution.
- interface covariance

    Computes the covariance matrix of two data sets.
- interface incomplete_beta

    Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)
- interface incomplete_gamma

    Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).
- interface incomplete_gamma_comp

    Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).

- interface mean

  Computes the mean of a data set.
- interface median

  Computes the median of a data set.
- interface standard_deviation

  Computes the corrected standard deviation of a data set.
- interface t_value

  Computes the t-value (t-score) given a percentage of the area under the standard normal distribution curve.
- interface variance

  Computes the sample variance of a data set.
- interface z_value

  Computes the z-value (z-score) given a percentage of the area under the standard normal distribution curve.

Functions/Subroutines

- pure real(dp) function mean_dbl (x)

  Computes the mean of a data set.
- real(dp) function median_dbl (x, srt)

  Computes the median of a data set.
- pure real(dp) function variance_dbl (x)

  Computes the sample variance of a data set.
- real(dp) function, dimension(2, 2) covariance_2sets (x, y, err)

  Computes the covariance matrix of two data sets.
- real(dp) function, dimension(size(x, 2), size(x, 2)) covariance_mtx (x, err)

  Computes the covariance matrix of N data sets of M observations.
- pure real(dp) function stdev_dbl (x)

  Computes the corrected standard deviation of a data set.
- real(dp) function conf_int (x, alpha, use_t, err)

  Computes the confidence interval based upon a standard normal distribution.
- real(dp) function std_norm_dist_z_score (alpha, err)

  Computes the z-value (z-score) given a percentage of the area under the standard normal distribution curve.
- real(dp) function t_dist_score (alpha, n, err)

  Computes the t-value (t-score) given a percentage of the area under the standard normal distribution curve.
- real(dp) function incomplete_gamma_scalar (a, x, err)

  Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).
- real(dp) function, dimension(size(x)) incomplete_gamma_array (a, x, err)

  Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).
- real(dp) function incomplete_gamma_comp_scalar (a, x, err)

  Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).
- real(dp) function, dimension(size(x)) incomplete_gamma_comp_array (a, x, err)

  Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).
- pure real(dp) function inc_gamma_series (a, x)

  Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x) is computed by its series representation.
- pure real(dp) function inc_gamma_cf (a, x)

  Computes the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x) is computed by Lentz's continued fraction approach.
- real(dp) function inc_beta_scalar (a, b, x, err)

  Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)
- real(dp) function, dimension(size(x)) inc_beta_array (a, b, x, err)

  Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)
- real(dp) function inc_beta_cf (a, b, x)

  Evaluates the incomplete beta function as a continued fraction.

5.6.1 Detailed Description

curvefit_statistics

Purpose

To provide a set of statistical routines for exploring curve fits of sets of numeric data.

5.6.2 Function/Subroutine Documentation

5.6.2.1 real(dp) function curvefit_statistics::conf_int (real(dp), dimension(:), intent(in) *x*, real(dp), intent(in) *alpha*, logical, intent(in), optional *use_t*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the confidence interval based upon a standard normal distribution.

Parameters

`in`	*x*	The data set.
`in`	*alpha*	The confidence level. This value must lie between zero and one such that: $0 < \text{alpha} < 1$.
`in`	*use↔_t*	Set to true to use the t-distribution in the event of an unknown true standard deviation; else, set to true to use a normal distribution. The default is false, such that a normal distribution is used by a default.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `alpha` is does not satisfy: $0 < \text{alpha} < 1$.

Returns

The confidence interval as the deviation from the mean.

Remarks

The confidence interval, assuming a standard normal distribution, is as follows: mu +/- z $*$ s / sqrt(n), where mu = the mean, and s = the standard deviation. This routine computes the z $*$ s / sqrt(n) portion leaving the computation of the mean to the user.

Definition at line 401 of file curvefit_statistics.f90.

5.6.2.2 real(dp) function, dimension(2,2) curvefit_statistics::covariance_2sets (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the covariance matrix of two data sets.

Parameters

`in`	*x*	An N-element array containing the first data set.
`in`	*y*	An N-element array containing the second data set.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are not the same size.

Returns

The 2-by-2 covariance matrix.

Definition at line 244 of file curvefit_statistics.f90.

5.6.2.3 real(dp) function, dimension(size(x, 2), size(x, 2)) curvefit_statistics::covariance_mtx (real(dp), dimension(:,:), intent(in) *x*, class(errors), intent(inout), optional, target *err*) [private]

Computes the covariance matrix of N data sets of M observations.

Parameters

in	*x*	The M-by-N matrix.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Returns

The N-by-N covariance matrix.

Definition at line 308 of file curvefit_statistics.f90.

5.6.2.4 real(dp) function, dimension(size(x)) curvefit_statistics::inc_beta_array (real(dp), intent(in) *a*, real(dp), intent(in) *b*, real(dp), dimension(:), intent(in) *x*, class(errors), intent(inout), optional, target *err*) [private]

Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)

Parameters

in	*a*	The parameter a.
in	*b*	The parameter b.
in	*x*	The parameter x. This parameter must lie in the interval: [0, 1].
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is not within its allowed range.

Definition at line 1054 of file curvefit_statistics.f90.

5.6.2.5 real(dp) function curvefit_statistics::inc_beta_cf (real(dp), intent(in) *a*, real(dp), intent(in) *b*, real(dp), intent(in) *x*) [private]

Evaluates the incomplete beta function as a continued fraction.

Parameters

in	*a*	The parameter a.
in	*b*	The parameter b.
in	*x*	The independent variable.

Returns

The result.

Definition at line 1120 of file curvefit_statistics.f90.

5.6.2.6 real(dp) function curvefit_statistics::inc_beta_scalar (real(dp), intent(in) *a*, real(dp), intent(in) *b*, real(dp), intent(in) *x*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)

Parameters

in	*a*	The parameter a.
in	*b*	The parameter b.
in	*x*	The parameter x. This parameter must lie in the interval: [0, 1].
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is not within its allowed range.

Definition at line 994 of file curvefit_statistics.f90.

5.6.2.7 pure real(dp) function curvefit_statistics::inc_gamma_cf (real(dp), intent(in) *a*, real(dp), intent(in) *x*) `[private]`

Computes the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x) is computed by Lentz's continued fraction approach.

Parameters

in	*a*	The parameter a.
in	*x*	The parameter x. This parameter must be greater than 0.

Returns

The incomplete gamma function.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gcf).

Definition at line 934 of file curvefit_statistics.f90.

5.6.2.8 pure real(dp) function curvefit_statistics::inc_gamma_series (real(dp), intent(in) *a*, real(dp), intent(in) *x*) [private]

Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x) is computed by its series representation.

Parameters

in	*a*	The parameter a.
in	*x*	The parameter x. This parameter must be greater than 0.

Returns

The incomplete gamma function.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gser).

Definition at line 886 of file curvefit_statistics.f90.

5.6.2.9 real(dp) function, dimension(size(x)) curvefit_statistics::incomplete_gamma_array (real(dp), intent(in) *a*, real(dp), dimension(:), intent(in) *x*, class(errors), intent(inout), optional, target *err*) [private]

Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).

Parameters

in	*a*	The coefficient. This parameter must be positive-valued.
in	*x*	An N-element array of independent variables. All values must be greater than or equal to zero.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is negative, or if `a` is not positive.
	return	The values of the function at `x`.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gammp).

Definition at line 679 of file curvefit_statistics.f90.

5.6.2.10 real(dp) function, dimension(size(x)) curvefit_statistics::incomplete_gamma_comp_array (real(dp), intent(in) *a*, real(dp), dimension(:), intent(in) *x*, class(errors), intent(inout), optional, target *err*) [private]

Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).

Parameters

in	*a*	The coefficient. This parameter must be positive-valued.
in	*x*	An N-element array of independent variables. All values must be greater than or equal to zero.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is negative, or if `a` is not positive.
	return	The values of the function at `x`.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gammq).

Definition at line 818 of file curvefit_statistics.f90.

5.6.2.11 real(dp) function curvefit_statistics::incomplete_gamma_comp_scalar (real(dp), intent(in) *a*, real(dp), intent(in) *x*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) * integrate(exp(-t) * t**(a - 1), t, 0, x).

Parameters

in	*a*	The coefficient. This parameter must be positive-valued.
in	*x*	The independent variable. This parameter must be greater than or equal to zero.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is negative, or if `a` is not positive.
	return	The value of the function at `x`.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gammq).

Definition at line 755 of file curvefit_statistics.f90.

5.6.2.12 real(dp) function curvefit_statistics::incomplete_gamma_scalar (real(dp), intent(in) *a*, real(dp), intent(in) *x*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) * integrate(exp(-t) * t**(a - 1), t, 0, x).

Parameters

`in`	*a*	The coefficient. This parameter must be positive-valued.
`in`	*x*	The independent variable. This parameter must be greater than or equal to zero.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is negative, or if `a` is not positive.
	return	The value of the function at `x`.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gammp).

Definition at line 617 of file curvefit_statistics.f90.

5.6.2.13 pure real(dp) function curvefit_statistics::mean_dbl (real(dp), dimension(:), intent(in) *x*) `[private]`

Computes the mean of a data set.

Parameters

`in`	*x*	The data set.

Returns

The mean of `x`.

Definition at line 117 of file curvefit_statistics.f90.

5.6.2.14 real(dp) function curvefit_statistics::median_dbl (real(dp), dimension(:), intent(inout) *x*, logical, intent(in), optional *srt*) `[private]`

Computes the median of a data set.

Parameters

`in,out`	*x*	The data set whose median is to be found. Ideally, the data set should be monotonically increasing; however, if it is not, it may be sorted by the routine, dependent upon the value of `srt`. On output, the array contents are unchanged; however, they may be sorted into ascending order (dependent upon the value of `srt`).
`in`	*srt*	An optional flag determining if `x` should be sorted. The default is to sort (true).

Returns

The median of `x`.

Definition at line 152 of file curvefit_statistics.f90.

5.6.2.15 real(dp) function curvefit_statistics::std_norm_dist_z_score (real(dp), intent(in) *alpha,* class(errors), intent(inout), optional, target *err*) [private]

Computes the z-value (z-score) given a percentage of the area under the standard normal distribution curve.

Parameters

in	*alpha*	The percentage of the area under the curve. This value must be between 0 and 1 such that: $0 < \text{alpha} < 1$.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `alpha` is does not satisfy: $0 < \text{alpha} < 1$.

Returns

The z-score or z-value by solving for z where: alpha = ERF(z / sqrt(2)), where ERF is the error function.

Definition at line 467 of file curvefit_statistics.f90.

5.6.2.16 pure real(dp) function curvefit_statistics::stdev_dbl (real(dp), dimension(:), intent(in) *x*) [private]

Computes the corrected standard deviation of a data set.

Parameters

in	*x*	The data set.

Returns

The standard deviation of `x`.

Definition at line 366 of file curvefit_statistics.f90.

5.6.2.17 real(dp) function curvefit_statistics::t_dist_score (real(dp), intent(in) *alpha,* integer(i32), intent(in) *n,* class(errors), intent(inout), optional, target *err*) [private]

Computes the t-value (t-score) given a percentage of the area under the standard normal distribution curve.

Parameters

in	*alpha*	The percentage of the area under the curve. This value must be between 0 and 1 such that: $0 < \text{alpha} < 1$.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `alpha` is does not satisfy: $0 < \text{alpha} < 1$.

Returns

The t-socre or t-value.

Definition at line 538 of file curvefit_statistics.f90.

5.6.2.18 pure real(dp) function curvefit_statistics::variance_dbl (real(dp), dimension(:), intent(in) *x*) `[private]`

Computes the sample variance of a data set.

Parameters

`in`	*x*	The data set.

Returns

The variance of `x`.

Remarks

To avoid overflow-type issues, Welford's algorithm is employed. A simple illustration of this algorithm can be found `here`.

Definition at line 202 of file curvefit_statistics.f90.

6 Data Type Documentation

6.1 curvefit_c_binding::c_linear_interp Type Reference

A C compatible type encapsulating a linear_interp object.

Public Attributes

- type(c_ptr) ptr

  A pointer to the linear_interp object.
- integer(i32) n

  The size of the linear_interp object, in bytes.

6.1.1 Detailed Description

A C compatible type encapsulating a linear_interp object.

Definition at line 45 of file curvefit_c_binding.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_c_binding.f90

6.2 curvefit_c_binding::c_lowess_smoothing Type Reference

A C compatible type encapsulating a lowess_smoothing object.

Public Attributes

- type(c_ptr) ptr

  A pointer to the lowess_smoothing object.
- integer(i32) n

  The size of the lowess_smoothing object, in bytes.

6.2.1 Detailed Description

A C compatible type encapsulating a lowess_smoothing object.

Definition at line 72 of file curvefit_c_binding.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_c_binding.f90

6.3 curvefit_c_binding::c_nonlinear_regression Type Reference

A C compatible type encapsulating a nonlinear_regression object.

Public Attributes

- type(c_ptr) ptr

  A pointer to the nonlinear_regression object.
- integer(i32) n

  The size of the nonlinear_regression object, in bytes.

6.3.1 Detailed Description

A C compatible type encapsulating a nonlinear_regression object.

Definition at line 81 of file curvefit_c_binding.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_c_binding.f90

6.4 curvefit_c_binding::c_polynomial_interp Type Reference

A C compatible type encapsulating a polynomial_interp object.

Public Attributes

- type(c_ptr) ptr

  A pointer to the polynomial_interp object.
- integer(i32) n

  The size of the polynomial_interp object, in bytes.

6.4.1 Detailed Description

A C compatible type encapsulating a polynomial_interp object.

Definition at line 54 of file curvefit_c_binding.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_c_binding.f90

6.5 curvefit_c_binding::c_spline_interp Type Reference

A C compatible type encapsulating a spline_interp object.

Public Attributes

- type(c_ptr) ptr

  A pointer to the spline_interp object.
- integer(i32) n

  The size of the spline_interp object, in bytes.

6.5.1 Detailed Description

A C compatible type encapsulating a spline_interp object.

Definition at line 63 of file curvefit_c_binding.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_c_binding.f90

6.6 curvefit_c_binding::cnonlin_reg_helper Type Reference

A type for helping to interface between a C function pointer, and the nonlinear_regression type.

Inheritance diagram for curvefit_c_binding::cnonlin_reg_helper:

6.7 curvefit_statistics::confidence_interval Interface Reference

Computes the confidence interval based upon a standard normal distribution.

Private Member Functions

- real(dp) function conf_int (x, alpha, use_t, err)

  Computes the confidence interval based upon a standard normal distribution.

6.7.1 Detailed Description

Computes the confidence interval based upon a standard normal distribution.

Definition at line 63 of file curvefit_statistics.f90.

6.7.2 Member Function/Subroutine Documentation

6.7.2.1 real(dp) function curvefit_statistics::confidence_interval::conf_int (real(dp), dimension(:), intent(in) *x*, real(dp), intent(in) *alpha*, logical, intent(in), optional *use_t*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the confidence interval based upon a standard normal distribution.

Parameters

`in`	*x*	The data set.
`in`	*alpha*	The confidence level. This value must lie between zero and one such that: $0 < \text{alpha} < 1$.
`in`	*use↵_t*	Set to true to use the t-distribution in the event of an unknown true standard deviation; else, set to true to use a normal distribution. The default is false, such that a normal distribution is used by a default.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `alpha` is does not satisfy: $0 < \text{alpha} < 1$.

Returns

The confidence interval as the deviation from the mean.

Remarks

The confidence interval, assuming a standard normal distribution, is as follows: mu +/- z $*$ s / sqrt(n), where mu = the mean, and s = the standard deviation. This routine computes the z $*$ s / sqrt(n) portion leaving the computation of the mean to the user.

Definition at line 401 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.8 curvefit_statistics::covariance Interface Reference

Computes the covariance matrix of two data sets.

Private Member Functions

- real(dp) function, dimension(2, 2) covariance_2sets (x, y, err)

  Computes the covariance matrix of two data sets.
- real(dp) function, dimension(size(x, 2), size(x, 2)) covariance_mtx (x, err)

  Computes the covariance matrix of N data sets of M observations.

6.8.1 Detailed Description

Computes the covariance matrix of two data sets.

Definition at line 49 of file curvefit_statistics.f90.

6.8.2 Member Function/Subroutine Documentation

6.8.2.1 real(dp) function, dimension(2,2) curvefit_statistics::covariance::covariance_2sets (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(in) *y*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the covariance matrix of two data sets.

Parameters

`in`	*x*	An N-element array containing the first data set.
`in`	*y*	An N-element array containing the second data set.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are not the same size.

Returns

The 2-by-2 covariance matrix.

Definition at line 244 of file curvefit_statistics.f90.

6.8.2.2 real(dp) function, dimension(size(x, 2), size(x, 2)) curvefit_statistics::covariance::covariance_mtx (real(dp), dimension(:,:), intent(in) *x*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the covariance matrix of N data sets of M observations.

Parameters

`in`	*x*	The M-by-N matrix.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Returns

The N-by-N covariance matrix.

Definition at line 308 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.9 curvefit_c_binding::creg_fcn Interface Reference

Describes a routine for finding the coefficients of a function of one variable.

Public Member Functions

- real(dp) function **creg_fcn** (x, n, c)

6.9.1 Detailed Description

Describes a routine for finding the coefficients of a function of one variable.

Parameters

`in`	*x*	The independent variable.
`in`	*n*	The number of coefficients in `c`.
`in`	*c*	An array of function coefficients.

Returns

The value of the function at `x`.

Definition at line 32 of file curvefit_c_binding.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_c_binding.f90

6.10 curvefit_calibration::crosstalk Interface Reference

Computes the crosstalk errors for a multiple degree-of-freedom data set.

Private Member Functions

- real(dp) function, dimension(size(xerr, 2), size(xerr, 2)) xtalk_1 (xerr, indices, err)

  Computes the crosstalk errors for a multiple degree-of-freedom data set.

6.10.1 Detailed Description

Computes the crosstalk errors for a multiple degree-of-freedom data set.

Definition at line 93 of file curvefit_calibration.f90.

6.10.2 Member Function/Subroutine Documentation

6.10.2.1 real(dp) function, dimension(size(xerr, 2), size(xerr, 2)) curvefit_calibration::crosstalk::xtalk_1 (real(dp), dimension(:,:), intent(in) *xerr,* integer(i32), dimension(:), intent(in) *indices,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the crosstalk errors for a multiple degree-of-freedom data set.

Parameters

`in`	*xerr*	An NPTS-by-NDOF matrix containing the measurement error values (computed such that XERR = X MEASURED - X APPLIED).
`in`	*indices*	A 2∗NDOF element array containing row indices defining the rows where each degree-of-freedom was applied in the data set `xerr`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `indices` is not 2∗NDOF in size. • CF_ARRAY_INDEX_ERROR: Occurs if any of the entries in `indices` are outside the row bounds of `xerr`.

Returns

A NDOF-by-NDOF matrix containing the crosstalk errors such that each loaded degree of freedom is represented by its own row, and each responding degree of freedom is represented by its own column.

Usage

The following program computes the crosstalk errors for a 2 DOF system. The applied data is as follows (there are 34 data points for each DOF).

```
0               0
3000            0
```

```
6000          0
7500          0
9000          0
12000         0
15000         0
7500          0
0             0
0             0
-3000         0
-6000         0
-7500         0
-9000         0
-12000        0
-15000        0
-7500         0
0             0
--------------------
0             0
0             67.79087067
0             135.5817413
0             203.3726196
0             271.1634827
0             338.9543762
0             203.3726196
0             0
0             0
0             -67.79087067
0             -135.5817413
0             -203.3726196
0             -271.1634827
0             -338.9543762
0             -203.3726196
0             0
```

The data output from the instrument under test is as follows.

```
0                  0
0.389050007        1.22E-03
0.778159976        2.59E-03
0.972689986        2.90E-03
1.167140007        3.14E-03
1.555999994        3.38E-03
1.944839954        3.56E-03
0.972599983        4.77E-03
-1E-05             -1.00E-05
0                  0
-0.388886005       2.10E-04
-0.777750015       5.10E-04
-0.972150028       6.90E-04
-1.166540027       8.80E-04
-1.555330038       1.30E-03
-1.944100022       1.78E-03
-0.971710026       5.80E-04
4.00E-05           3.00E-05
----------------------------
0                  0
-4.40E-04          0.271560013
-1.30E-03          0.543290019
-2.40E-03          0.815069973
-3.82E-03          1.086820006
-5.28E-03          1.358809948
-2.57E-03          0.815530002
1.50E-04           1.00E-05
0                  0
1.44E-03           -0.271450013
3.06E-03           -0.543120027
4.46E-03           -0.814930022
5.67E-03           -1.086799979
6.88E-03           -1.35879004
4.51E-03           -0.815479994
-2.00E-05          0
```

The code to compute the crosstalk errors given the above raw data is then as follows.

```
program main
```

```
    ! Parameters
    integer(i32), parameter :: npts = 34
    integer(i32), parameter :: ndof = 2

    ! Local Variables
    integer(i32) :: indices(2*ndof)
    real(dp), dimension(npts, ndof) :: xin, xout, xerr, xmeas
    real(dp), dimension(ndof, npts) :: xint, xmeast
    real(dp), dimension(ndof, ndof) :: c, ans, xt

    ! Initialization
    xin = reshape([0.0, 3000.0, 6000.0, 7500.0, 9000.0, 12000.0, &
        15000.0, 7500.0, 0.0, 0.0, -3000.0, -6000.0, -7500.0, -9000.0, &
        -12000.0, -15000.0, -7500.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, &
        0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, &
        0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, &
        0.0, 0.0, 0.0, 67.7908728, 135.5817456, 203.3726184, 271.1634912, &
        338.954364, 203.3726184, 0.0, 0.0, -67.7908728, -135.5817456, &
        -203.3726184, -271.1634912, -338.954364, -203.3726184, 0.0], &
        [npts, ndof])
    xout = reshape([0.0, 0.38905, 0.77816, 0.97269, 1.16714, 1.556, &
        1.94484, 0.9726, -1.0e-5, 0.0, -0.388886, -0.77775, -0.97215, &
        -1.16654, -1.55533, -1.9441, -0.97171, 4.0e-5, 0.0, -0.00044, &
        -0.0013, -0.0024, -0.00382, -0.00528, -0.00257, 0.00015, 0.0, &
        0.00144, 0.00306, 0.00446, 0.00567, 0.00688, 0.00451, -2.0e-5, &
        0.0, 0.00122, 0.00259, 0.0029, 0.00314, 0.00338, 0.00356, 0.00477,&
        -1.0e-5, 0.0, 0.00021, 0.00051, 0.00069, 0.00088, 0.0013, 0.00178,&
        0.00058, 3.0e-5, 0.0, 0.27156, 0.54329, 0.81507, 1.08682, 1.35881,&
        0.81553, 1.0e-5, 0.0, -0.27145, -0.54312, -0.81493, -1.0868, &
        -1.35879, -0.81548, 0.0], [npts, ndof])

    ! Compute the calibration gains
    xint = transpose(xin)
    xmeast = transpose(xout)
    c = linear_least_squares(xmeast, xint)
    xmeas = matmul(xout, transpose(c))
    xerr = xmeas - xin

    ! The indices are
    indices = [1, 17, 18, 34]

    ! Compute the crosstalk matrix
    xt = crosstalk(xerr, indices)
end program
```

The least squares fit generates the following matrix of calibration gains.

```
7713.710427        33.5206917
-0.214743728       249.4768498
```

The resulting measured values are then as follows.

```
0                  0
3001.05999         0.220815702
6002.58754         0.479040049
7503.146099        0.514603781
9003.085297        0.532721294
12002.64668        0.509090486
15002.05157        0.470495427
7502.514526        0.981144827
-0.077472309       -0.002492621
0                  0
-2999.74699        0.135900968
-5999.321307       0.294250127
-7498.860677       0.380902151
-8998.322469       0.470046784
-11997.32196       0.658317276
-14996.16495       0.861552092
-7495.47032        0.353365205
0.30955403         7.48E-03
----------------------------
0                  0
5.708846869        67.74803112
8.183633648        135.5385616
8.808803389        203.3416047
6.96458466         271.1372518
4.81985707         338.9927591
7.512893885        203.4564077
1.157391826        2.46E-03
0                  0
2.008550989        -67.72080332
```

```
5.398195074        -135.4965304
7.086130239        -203.3071324
7.306450061        -271.1326527
7.522743936        -338.9881361
7.453379661        -203.4443484
-0.154274205       4.29E-06
```

The crosstalk error matrix is then as follows.

```
0                  0.981144827
8.808803389        0
```

Definition at line 885 of file curvefit_calibration.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.11 curvefit_calibration::hysteresis Interface Reference

Computes the hysteresis in an ascending/descending data set.

Private Member Functions

- real(dp) function hysteresis_1 (xascend, ascend, xdescend, descend, err)

  Computes the hysteresis in an ascending/descending data set.
- real(dp) function hysteresis_2 (applied, measured, err)

  Computes the hysteresis in an ascending/descending data set.

6.11.1 Detailed Description

Computes the hysteresis in an ascending/descending data set.

Definition at line 72 of file curvefit_calibration.f90.

6.11.2 Member Function/Subroutine Documentation

6.11.2.1 real(dp) function curvefit_calibration::hysteresis::hysteresis_1 (real(dp), dimension(:), intent(in) *xascend,* real(dp), dimension(:), intent(in) *ascend,* real(dp), dimension(:), intent(in) *xdescend,* real(dp), dimension(:), intent(in) *descend,* class(errors), intent(inout), optional, target *err*) [private]

Computes the hysteresis in an ascending/descending data set.

Parameters

in	*xascend*	An N-element array containing the ascending calibration points. This array must be monotonically increasing or decreasing.
in	*ascend*	An N-element array containing the sensor output to the calibration points in `xascend`.
in	*xdescend*	An M-element array containing the descending calibration points. This array must be monotonically increasing or decreasing.
in	*descend*	An M-element array containing the sensor output to the calibration points in `xdescend`.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.

Returns

The hysteresis error.

Remarks

In order to account for slight variations between similar ascending and descending points, the algorithm used performs a linear interpolation between data points. The resulting interpolated value is then used to compute the reported hysteresis error.

Definition at line 353 of file curvefit_calibration.f90.

6.11.2.2 real(dp) function curvefit_calibration::hysteresis::hysteresis_2 (real(dp), dimension(:), intent(in) *applied,* real(dp), dimension(:), intent(in) *measured,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the hysteresis in an ascending/descending data set.

Parameters

`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_NONMONOTONIC_ARRAY_ERROR: Occurs if the calibration data is not monotonic in nature (either ascending or descending). • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The hysteresis error.

Remarks

In order to account for slight variations between similar ascending and descending points, the algorithm used performs a linear interpolation between data points. The resulting interpolated value is then used to compute the reported hysteresis error.

Definition at line 444 of file curvefit_calibration.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.12 curvefit_calibration::IDAMAX Interface Reference

Private Member Functions

- function **idamax** (n, dx, incx)

6.12.1 Detailed Description

Definition at line 45 of file curvefit_calibration.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.13 curvefit_statistics::incomplete_beta Interface Reference

Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)

Private Member Functions

- real(dp) function inc_beta_scalar (a, b, x, err)

  Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)
- real(dp) function, dimension(size(x)) inc_beta_array (a, b, x, err)

  Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)

6.13.1 Detailed Description

Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)

Definition at line 101 of file curvefit_statistics.f90.

6.13.2 Member Function/Subroutine Documentation

6.13.2.1 real(dp) function, dimension(size(x)) curvefit_statistics::incomplete_beta::inc_beta_array (real(dp), intent(in) *a*, real(dp), intent(in) *b*, real(dp), dimension(:), intent(in) *x*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the incomplete beta function: I(a,b) = 1 / B(a,b) ∗ integrate(t∗∗(a - 1) ∗ (1 - t)∗∗(b - 1), t, 0, x)

Parameters

`in`	*a*	The parameter a.
`in`	*b*	The parameter b.
`in`	*x*	The parameter x. This parameter must lie in the interval: [0, 1].
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is not within its allowed range.

Definition at line 1054 of file curvefit_statistics.f90.

6.13.2.2 real(dp) function curvefit_statistics::incomplete_beta::inc_beta_scalar (real(dp), intent(in) *a,* real(dp), intent(in) *b,* real(dp), intent(in) *x,* class(errors), intent(inout), optional, target *err*) [private]

Computes the incomplete beta function: I(a,b) = 1 / B(a,b) * integrate(t**(a - 1) * (1 - t)**(b - 1), t, 0, x)

Parameters

in	*a*	The parameter a.
in	*b*	The parameter b.
in	*x*	The parameter x. This parameter must lie in the interval: [0, 1].
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is not within its allowed range.

Definition at line 994 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.14 curvefit_statistics::incomplete_gamma Interface Reference

Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) * integrate(exp(-t) * t**(a - 1), t, 0, x).

Private Member Functions

- real(dp) function incomplete_gamma_scalar (a, x, err)

  *Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) * integrate(exp(-t) * t**(a - 1), t, 0, x).*
- real(dp) function, dimension(size(x)) incomplete_gamma_array (a, x, err)

  *Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) * integrate(exp(-t) * t**(a - 1), t, 0, x).*

6.14.1 Detailed Description

Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) * integrate(exp(-t) * t**(a - 1), t, 0, x).

Definition at line 84 of file curvefit_statistics.f90.

6.14.2 Member Function/Subroutine Documentation

6.14.2.1 real(dp) function, dimension(size(x)) curvefit_statistics::incomplete_gamma::incomplete_gamma_array (real(dp), intent(in) *a,* real(dp), dimension(:), intent(in) *x,* class(errors), intent(inout), optional, target *err*) [private]

Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) * integrate(exp(-t) * t**(a - 1), t, 0, x).

Parameters

`in`	*a*	The coefficient. This parameter must be positive-valued.
`in`	*x*	An N-element array of independent variables. All values must be greater than or equal to zero.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is negative, or if `a` is not positive.
	return	The values of the function at `x`.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gammp).

Definition at line 679 of file curvefit_statistics.f90.

6.14.2.2 real(dp) function curvefit_statistics::incomplete_gamma::incomplete_gamma_scalar (real(dp), intent(in) *a*, real(dp), intent(in) *x*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the incomplete gamma function: P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).

Parameters

`in`	*a*	The coefficient. This parameter must be positive-valued.
`in`	*x*	The independent variable. This parameter must be greater than or equal to zero.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is negative, or if `a` is not positive.
	return	The value of the function at `x`.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gammp).

Definition at line 617 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.15 curvefit_statistics::incomplete_gamma_comp Interface Reference

Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).

Private Member Functions

- real(dp) function incomplete_gamma_comp_scalar (a, x, err)

  Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).
- real(dp) function, dimension(size(x)) incomplete_gamma_comp_array (a, x, err)

  Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).

6.15.1 Detailed Description

Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).

Definition at line 93 of file curvefit_statistics.f90.

6.15.2 Member Function/Subroutine Documentation

6.15.2.1 real(dp) function, dimension(size(x)) curvefit_statistics::incomplete_gamma_comp::incomplete_gamma_comp_array (real(dp), intent(in) *a*, real(dp), dimension(:), intent(in) *x*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).

Parameters

`in`	*a*	The coefficient. This parameter must be positive-valued.
`in`	*x*	An N-element array of independent variables. All values must be greater than or equal to zero.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is negative, or if `a` is not positive.
	return	The values of the function at `x`.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gammq).

Definition at line 818 of file curvefit_statistics.f90.

6.15.2.2 real(dp) function curvefit_statistics::incomplete_gamma_comp::incomplete_gamma_comp_scalar (real(dp), intent(in) *a*, real(dp), intent(in) *x*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the complement of the incomplete gamma function: Q(a,x) = 1 - P(a,x), where P(a,x) = 1 / gamma(a) ∗ integrate(exp(-t) ∗ t∗∗(a - 1), t, 0, x).

Parameters

in	*a*	The coefficient. This parameter must be positive-valued.
in	*x*	The independent variable. This parameter must be greater than or equal to zero.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `x` is negative, or if `a` is not positive.
	return	The value of the function at `x`.

Remarks

This implementation is based upon the Numerical Recipes implementation found in section 6.2 of the text (routine: gammq).

Definition at line 755 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.16 curvefit_interp::interp_manager Type Reference

Describes an abstract base class allowing for interpolation of X-Y type data sets.

Inheritance diagram for curvefit_interp::interp_manager:

Public Member Functions

- procedure, public initialize => im_init

  Initializes the interp_manager instance.
- procedure, non_overridable, public locate => im_locate

  Attempts to locate the index in the array providing a lower bounds to the specified interpolation point.
- procedure, non_overridable, public hunt => im_hunt

  Attempts to locate the index in the array providing a lower bounds to the specified interpolation point.
- generic, public interpolate => im_perform, im_perform_array

  Interpolates to obtain the function value at the specified independent variable.
- procedure, public get_count => im_get_num_pts

  Gets the number of stored data points.
- procedure, public get_x => im_get_x

  Gets the x component of the requested data point.
- procedure, public get_y => im_get_y

  Gets the y component of the requested data point.

Private Member Functions

- procedure(interp_xy), deferred raw_interp

  Performs the actual interpolation.
- procedure, non_overridable **im_perform**
- procedure, non_overridable **im_perform_array**

Private Attributes

- integer(i32) **m_order**
- integer(i32) **m_savedindex**
- integer(i32) **m_indexcheck**
- logical **m_correlated**
- real(dp), dimension(:), allocatable **m_x**
- real(dp), dimension(:), allocatable **m_y**

6.16.1 Detailed Description

Describes an abstract base class allowing for interpolation of X-Y type data sets.

Notes

This interpolation object is conceptually based upon the interpolation scheme utilized by the Numerical Recipes in C++ text.

Definition at line 48 of file curvefit_interp.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_interp.f90

6.17 curvefit_interp::interp_xy Interface Reference

Defines the signature of a method used to interpolate a single value in an X-Y data set.

Private Member Functions

- real(dp) function **interp_xy** (this, jlo, pt)

6.17.1 Detailed Description

Defines the signature of a method used to interpolate a single value in an X-Y data set.

Parameters

`in,out`	*this*	The interp_manager based instance.
`in`	*jlo*	The array index below which `pt` is found in x.
`in`	*pt*	The independent variable value to interpolate.

Returns

The interpolated value.

Definition at line 148 of file curvefit_interp.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_interp.f90

6.18 curvefit_core::is_monotonic Interface Reference

Tests to see if an array is montonically increasing or decreasing.

Private Member Functions

- pure logical function is_monotonic_dbl (x)
  Tests to see if an array is montonically increasing or decreasing.
- pure logical function is_monotonic_i32 (x)
  Tests to see if an array is montonically increasing or decreasing.

6.18.1 Detailed Description

Tests to see if an array is montonically increasing or decreasing.

Definition at line 74 of file curvefit_core.f90.

6.18.2 Member Function/Subroutine Documentation

6.18.2.1 pure logical function curvefit_core::is_monotonic::is_monotonic_dbl (real(dp), dimension(:), intent(in) *x*) [private]

Tests to see if an array is montonically increasing or decreasing.

Parameters

in	*x*	The array to test.

Returns

Returns true if `x` is monotonic; else, false.

Definition at line 105 of file curvefit_core.f90.

6.18.2.2 pure logical function curvefit_core::is_monotonic::is_monotonic_i32 (integer(i32), dimension(:), intent(in) *x*) [private]

Tests to see if an array is montonically increasing or decreasing.

Parameters

in	*x*	The array to test.

Returns

Returns true if `x` is monotonic; else, false.

Definition at line 139 of file curvefit_core.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_core.f90

6.19 curvefit_interp::linear_interp Type Reference

Extends the interp_manager class allowing for linear, piecewise interpolation of a data set.

Inheritance diagram for curvefit_interp::linear_interp:

Collaboration diagram for curvefit_interp::linear_interp:

Private Member Functions

- procedure raw_interp => li_raw_interp

  Performs the actual interpolation.

Additional Inherited Members

6.19.1 Detailed Description

Extends the interp_manager class allowing for linear, piecewise interpolation of a data set.

Definition at line 84 of file curvefit_interp.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_interp.f90

6.20 curvefit_regression::linear_least_squares Interface Reference

Employs a least squares fit to determine the coefficient A in the linear system: Y = A $*$ X, where A can either be a scalar, or a matrix.

Private Member Functions

- real(dp) function linear_least_squares_1var (x, y, err)

  Employs a least squares fit to determine the coefficient A in the linear system: Y = A $*$ X.
- real(dp) function, dimension(size(y, 1), size(x, 1)) linear_least_squares_nvar (x, y, thrsh, err)

  Employs a least squares fit to determine the coefficient A in the linear system: Y = A $*$ X.

6.20.1 Detailed Description

Employs a least squares fit to determine the coefficient A in the linear system: Y = A $*$ X, where A can either be a scalar, or a matrix.

Definition at line 36 of file curvefit_regression.f90.

6.20.2 Member Function/Subroutine Documentation

6.20.2.1 real(dp) function curvefit_regression::linear_least_squares::linear_least_squares_1var (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(inout) *y*, class(errors), intent(inout), optional, target *err*) `[private]`

Employs a least squares fit to determine the coefficient A in the linear system: Y = A $*$ X.

Parameters

`in`	*x*	An N-element array containing the independent variable data.
`in,out`	*y*	An N-element array containing the dependent variable data corresponding to `x`. On output, the contents of this array are overwritten as it is used for storage purposes by the algorithm.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available. • CF_ARRAY_SIZE_ERROR: Occurs if `x` and `y` are different sizes.

Returns

The scalar coefficient A.

Definition at line 1081 of file curvefit_regression.f90.

6.20.2.2 real(dp) function, dimension(size(y,1), size(x,1)) curvefit_regression::linear_least_squares::linear_least_squares_nvar (real(dp), dimension(:,:), intent(inout) *x*, real(dp), dimension(:,:), intent(in) *y*, real(dp), intent(in), optional *thrsh*, class(errors), intent(inout), optional, target *err*) `[private]`

Employs a least squares fit to determine the coefficient A in the linear system: Y = A $*$ X.

Parameters

`in,out`	*x*	An M-by-P matrix containing the P data points of the M independent variables.

Parameters

in	*y*	An N-by-P matrix containing the P data points of the N dependent variables.
in	*thrsh*	An optional threshold value that defines a lower cutoff for singular values. Any singular values falling below this value will have their reciprocal replaced with zero.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if any of the matrix dimensions are not compatiable. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Returns

An N-by-M matrix relating Y to X such that: Y = A ∗ X.

Remarks

The algorithm attempts to compute the coefficient matrix A as follows. Y ∗ X∗∗T = A ∗ X ∗ X∗∗T Y ∗ X∗∗T ∗ INV(X ∗ X∗∗T) = A This does require that X ∗ X∗∗T does not result in a singular matrix. To handle the situation where X ∗ X∗∗T is singular, the Moore-Penrose pseudo-inverse, computed by means of singular value decomposition, is utilized to still arrive at a solution that, at minimum, has a minimum Euclidean norm of its residual. Let: PINV(X) = X∗∗T ∗ INV(X ∗ X∗∗T), Then: A = Y ∗ PINV(X)

Definition at line 1150 of file curvefit_regression.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_regression.f90

6.21 curvefit_regression::lowess_smoothing Type Reference

Defines a type for computing a smoothing of an X-Y data set using a robust locally weighted scatterplot smoothing (LOWESS) algorithm.

Public Member Functions

- procedure, public initialize => ls_init

    Initializes the lowess_smoothing object.
- procedure, public smooth => ls_smooth

    Performs the actual smoothing operation.
- procedure, public get_count => ls_get_num_pts

    Gets the number of stored data points.
- procedure, public get_x => ls_get_x

    Gets the x component of the requested data point.
- procedure, public get_y => ls_get_y

    Gets the y component of the requested data point.
- procedure, public get_residuals => ls_get_residual

    Gets the residuals from each data point.

Private Attributes

- real(dp), dimension(:), allocatable m_x

  N-element array of x data points - sorted into ascending order.
- real(dp), dimension(:), allocatable m_y

  N-element array of y data points.
- real(dp), dimension(:), allocatable m_weights

  N-element array containing the robustness weights for each data point.
- real(dp), dimension(:), allocatable m_residuals

  N-element array containing the residuals (Y - YS)
- real(dp) m_delta

  Scaling parameter used to define the nature of the linear interpolations used by the algorithm.
- logical m_init = .false.

  Tracks whether or not ls_init has been called.

6.21.1 Detailed Description

Defines a type for computing a smoothing of an X-Y data set using a robust locally weighted scatterplot smoothing (LOWESS) algorithm.

Definition at line 46 of file curvefit_regression.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_regression.f90

6.22 curvefit_statistics::mean Interface Reference

Computes the mean of a data set.

Private Member Functions

- pure real(dp) function mean_dbl (x)

  Computes the mean of a data set.

6.22.1 Detailed Description

Computes the mean of a data set.

Definition at line 31 of file curvefit_statistics.f90.

6.22.2 Member Function/Subroutine Documentation

6.22.2.1 pure real(dp) function curvefit_statistics::mean::mean_dbl (real(dp), dimension(:), intent(in) *x*) [private]

Computes the mean of a data set.

Parameters

in	*x*	The data set.

Returns

The mean of `x`.

Definition at line 117 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.23 curvefit_statistics::median Interface Reference

Computes the median of a data set.

Private Member Functions

- real(dp) function median_dbl (x, srt)
  Computes the median of a data set.

6.23.1 Detailed Description

Computes the median of a data set.

Definition at line 37 of file curvefit_statistics.f90.

6.23.2 Member Function/Subroutine Documentation

6.23.2.1 real(dp) function curvefit_statistics::median::median_dbl (real(dp), dimension(:), intent(inout) *x*, logical, intent(in), optional *srt*) `[private]`

Computes the median of a data set.

Parameters

in,out	*x*	The data set whose median is to be found. Ideally, the data set should be monotonically increasing; however, if it is not, it may be sorted by the routine, dependent upon the value of `srt`. On output, the array contents are unchanged; however, they may be sorted into ascending order (dependent upon the value of `srt`).
in	*srt*	An optional flag determining if `x` should be sorted. The default is to sort (true).

Returns

The median of `x`.

Definition at line 152 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.24 curvefit_regression::moving_average Interface Reference

Applies a moving average to smooth a data set.

Private Member Functions

- subroutine moving_average_1 (x, npts, err)
  Applies a moving average to smooth a data set.

6.24.1 Detailed Description

Applies a moving average to smooth a data set.

Definition at line 29 of file curvefit_regression.f90.

6.24.2 Member Function/Subroutine Documentation

6.24.2.1 subroutine curvefit_regression::moving_average::moving_average_1 (real(dp), dimension(:), intent(inout) *x*, integer(i32), intent(in) *npts*, class(errors), intent(inout), optional, target *err*) `[private]`

Applies a moving average to smooth a data set.

Parameters

`in,out`	*x*	On input, the signal to smooth. On output, the smoothed signal.
`in`	*npts*	The size of the averaging window. This value must be at least 2, but no more than the number of elements in `x`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `npts` is less than 2, or greater than the length of `x`. • CF_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available.

Definition at line 164 of file curvefit_regression.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_regression.f90

6.25 curvefit_regression::nonlinear_regression Type Reference

A type for supporting nonlinear regression calculations.

Inheritance diagram for curvefit_regression::nonlinear_regression:

Collaboration diagram for curvefit_regression::nonlinear_regression:

Public Member Functions

- procedure, public initialize => nr_init

    Initializes the nonlinear_regression object.
- procedure, public fcn => nr_fcn

    Computes the residual between the supplied data set, and the function value given a set of coefficients.
- procedure, public is_fcn_defined => nr_is_fcn_defined

    Determines if the function has been defined.
- procedure, public get_equation_count => nr_get_eqn_count

    Gets the number of equations required to solve the regression problem.
- procedure, public get_variable_count => nr_get_var_count

    Gets the number of variables (coefficients).
- procedure, public solve => nr_solve

    Computes the solution to the nonlinear regression problem using the Levenberg-Marquardt method.
- procedure, public get_max_fcn_evals => nr_get_max_eval

    Gets the maximum number of function evaluations allowed during a single solve.
- procedure, public set_max_fcn_evals => nr_set_max_eval

    Sets the maximum number of function evaluations allowed during a single solve.
- procedure, public get_fcn_tolerance => nr_get_fcn_tol

    Gets the convergence on function value tolerance.
- procedure, public set_fcn_tolerance => nr_set_fcn_tol

    Sets the convergence on function value tolerance.
- procedure, public get_var_tolerance => nr_get_var_tol

    Gets the convergence on change in variable tolerance.
- procedure, public set_var_tolerance => nr_set_var_tol

    Sets the convergence on change in variable tolerance.
- procedure, public get_gradient_tolerance => nr_get_grad_tol

    Gets the convergence on slope of the gradient vector tolerance.
- procedure, public set_gradient_tolerance => nr_set_grad_tol

    Sets the convergence on slope of the gradient vector tolerance.
- procedure, public get_print_status => nr_get_print_status

    Gets a logical value determining if iteration status should be printed.
- procedure, public set_print_status => nr_set_print_status

    Sets a logical value determining if iteration status should be printed.
- procedure, public get_count => nr_get_num_pts

    Gets the number of stored data points.
- procedure, public get_x => nr_get_x

    Gets the x component of the requested data point.
- procedure, public get_y => nr_get_y

    Gets the y component of the requested data point.

Private Attributes

- procedure(reg_fcn), pointer, nopass m_rfcn => null()
  A pointer to the routine containing the function of interest.
- real(dp), dimension(:), allocatable m_x
  The x data points.
- real(dp), dimension(:), allocatable m_y
  The y data points.
- integer(i32) m_ncoeff = 0
  The number of coefficients in the function of interest.
- logical m_init = .false.
  Tracks whether or not nr_init has been called.
- type(least_squares_solver) m_solver
  The Levenberg-Marquardt solver.

6.25.1 Detailed Description

A type for supporting nonlinear regression calculations.

Definition at line 79 of file curvefit_regression.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_regression.f90

6.26 curvefit_calibration::nonlinearity Interface Reference

Computes the best-fit nonlinearity of a data set.

Private Member Functions

- real(dp) function bf_nonlin (applied, measured, err)
  Computes the best-fit nonlinearity of a data set.

6.26.1 Detailed Description

Computes the best-fit nonlinearity of a data set.

Definition at line 60 of file curvefit_calibration.f90.

6.26.2 Member Function/Subroutine Documentation

6.26.2.1 real(dp) function curvefit_calibration::nonlinearity::bf_nonlin (real(dp), dimension(:), intent(in) *applied,* real(dp), dimension(:), intent(in) *measured,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the best-fit nonlinearity of a data set.

Parameters

in	*applied*	An N-element array containing the values applied to the measurement instrument.
in	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The nonlinearity error.

Definition at line 211 of file curvefit_calibration.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.27 curvefit_interp::polynomial_interp Type Reference

Extends the interp_manager class allowing for polynomial interpolation of a data set.

Inheritance diagram for curvefit_interp::polynomial_interp:

Collaboration diagram for curvefit_interp::polynomial_interp:

Public Member Functions

- procedure, public initialize => pi_init

  Initializes the polynomial_interp instance.

Private Member Functions

- procedure raw_interp => pi_raw_interp

  Performs the actual interpolation.

Private Attributes

- real(dp), dimension(:), allocatable **m_c**
- real(dp), dimension(:), allocatable **m_d**
- real(dp) **m_dy**

6.27.1 Detailed Description

Extends the interp_manager class allowing for polynomial interpolation of a data set.

Definition at line 93 of file curvefit_interp.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_interp.f90

6.28 curvefit_core::reg_fcn Interface Reference

Describes a routine for finding the coefficients of a function of one variable.

Private Member Functions

- real(real64) function **reg_fcn** (x, c)

6.28.1 Detailed Description

Describes a routine for finding the coefficients of a function of one variable.

Parameters

in	*x*	The independent variable.
in	*c*	An array of function coefficients.

Returns

The value of the function at `x`.

Definition at line 88 of file curvefit_core.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_core.f90

6.29 curvefit_calibration::repeatability Interface Reference

Computes the repeatability of a sequence of tests.

Private Member Functions

- real(dp) function repeat_1 (applied, measured, err)
  Computes the repeatability of a sequence of tests.

6.29.1 Detailed Description

Computes the repeatability of a sequence of tests.

Definition at line 86 of file curvefit_calibration.f90.

6.29.2 Member Function/Subroutine Documentation

6.29.2.1 real(dp) function curvefit_calibration::repeatability::repeat_1 (real(dp), dimension(:,:), intent(in) *applied*, real(dp), dimension(:,:), intent(in) *measured*, class(errors), intent(inout), optional, target *err*) `[private]`

Computes the repeatability of a sequence of tests.

Parameters

`in`	*applied*	An NPTS-by-NTEST matrix containing at least 2 columns (tests) of NPTS values applied to the measurement instrument.
`in`	*measured*	An NPTS-by-NTEST matrix containing the corresponding calibrated output from the instrument.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The largest magnitude deviation from the initial test.

Remarks

Repeatability is considered as the largest magnitude deviation of subsequent tests from the initial test. Noting that it is very likely that consecutive test points will vary slightly, test 2 through test N are linearly interpolated such that their test points line up with those from test 1.

Definition at line 608 of file curvefit_calibration.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.30 curvefit_calibration::return_to_zero Interface Reference

Computes the return to zero error in an ascending/descending data set.

Private Member Functions

- real(dp) function rtz_1 (applied, measured, tol, err)

  Computes the return to zero error in an ascending/descending data set.

6.30.1 Detailed Description

Computes the return to zero error in an ascending/descending data set.

Definition at line 80 of file curvefit_calibration.f90.

6.30.2 Member Function/Subroutine Documentation

6.30.2.1 real(dp) function curvefit_calibration::return_to_zero::rtz_1 (real(dp), dimension(:), intent(in) *applied,* real(dp), dimension(:), intent(in) *measured,* real(dp), intent(in), optional *tol,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the return to zero error in an ascending/descending data set.

Parameters

`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.
`in`	*tol*	An optional input that specifies the tolerance used in finding the matching data points. If no value is specified, the default value of the square root of machine precision times the largest magnitude value in `xcal` is used.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The return to zero error.

Definition at line 522 of file curvefit_calibration.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.31 curvefit_calibration::seb Interface Reference

Computes the static error band of a data set.

Private Member Functions

- type(seb_results) function seb_1 (applied, output, fullscale, err)

  Computes the static error band of a data set.

6.31.1 Detailed Description

Computes the static error band of a data set.

Definition at line 54 of file curvefit_calibration.f90.

6.31.2 Member Function/Subroutine Documentation

6.31.2.1 type(seb_results) function curvefit_calibration::seb::seb_1 (real(dp), dimension(:), intent(in) *applied,* real(dp), dimension(:), intent(in) *output,* real(dp), intent(in) *fullscale,* class(errors), intent(out), optional, target *err*) `[private]`

Computes the static error band of a data set.

Parameters

`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*output*	An N-element array containing the values output by the instrument as a result of the values given in `applied`.
`in`	*fullscale*	The full scale measurement value for the instrument. The units must be consistent with those of `applied`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `output` are not the same size. • CF_INVALID_INPUT_ERROR: Occurs if `fullscale` is sufficiently close to zero to be considered zero. Sufficiently close in this instance is considered to be the square root of machine precision.

Returns

The static error band information.

Definition at line 126 of file curvefit_calibration.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.32 curvefit_calibration::seb_results Type Reference

Defines a container for static error band related information.

Private Attributes

- real(dp) seb

  The static error band.
- real(dp) output

  The static error band output, at full scale load.
- real(dp) slope

  The slope of the static error band fit.

6.32.1 Detailed Description

Defines a container for static error band related information.

Definition at line 32 of file curvefit_calibration.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.33 curvefit_interp::spline_interp Type Reference

Extends the interp_manager class allowing for cubic spline interpolation of a data set.

Inheritance diagram for curvefit_interp::spline_interp:

Collaboration diagram for curvefit_interp::spline_interp:

Public Member Functions

- procedure, public initialize => si_init_1

  Initializes the spline_interp instance.
- procedure, public initialize_spline => si_init_2

  Initializes the spline_interp instance while allowing definition of boundary conditions.
- generic, public first_derivative => si_diff1, si_diff1_array

  Interpolates to obtain the first derivative value at the specified independent variable.
- generic, public second_derivative => si_diff2, si_diff2_array

  Interpolates to obtain the second derivative value at the specified independent variable.

Private Member Functions

- procedure raw_interp => si_raw_interp

  Performs the actual interpolation.
- procedure compute_diff2 => si_second_deriv

  Computes the second derivative terms for the cubic-spline model.
- procedure **si_diff1**
- procedure **si_diff1_array**
- procedure **si_diff2**
- procedure **si_diff2_array**

Private Attributes

- real(dp), dimension(:), allocatable **m_ypp**

6.33.1 Detailed Description

Extends the interp_manager class allowing for cubic spline interpolation of a data set.

Definition at line 108 of file curvefit_interp.f90.

The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_interp.f90

6.34 curvefit_calibration::split_ascend_descend Interface Reference

Splits a data set into ascending and descending components.

Private Member Functions

- subroutine split_ascend_descend_1 (x, ascend, descend, nascend, ndescend, err)

  Splits a data set into ascending and descending components.

6.34.1 Detailed Description

Splits a data set into ascending and descending components.

Definition at line 99 of file curvefit_calibration.f90.

6.34.2 Member Function/Subroutine Documentation

6.34.2.1 subroutine curvefit_calibration::split_ascend_descend::split_ascend_descend_1 (real(dp), dimension(:), intent(in) *x*, real(dp), dimension(:), intent(out) *ascend*, real(dp), dimension(:), intent(out) *descend*, integer(i32), intent(out) *nascend*, integer(i32), intent(out) *ndescend*, class(errors), intent(inout), optional, target *err*) [private]

Splits a data set into ascending and descending components.

Parameters

in	*x*	An N-element array containing the data set to split.
out	*ascend*	An array where the ascending points will be written. Ensure this array is appropriately sized to accept all the ascending points (it can be oversized).
out	*descend*	An array where the descending points will be written. Ensure this array is appropriately sized to accept all the descending points (it can be oversized).
out	*nascend*	The actual number of values written into `ascend`.
out	*ndescend*	The actual number of values written into `descend`.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if either `ascend` or `descend` is too small to actually accept all of the necessary data.

Remarks

The routine operates by finding the first occurrence where the data set is no longer monotonic, and then copies everything prior to that value, along with the the inflection value, into the output ascending data array. The routine then searches for either a change in direction, or a value that matches the first value in the ascending data set within some tolerance to determine the bounds on the descending data set. Once the bounds are determined, the descending data set is copied from the original array and placed in the output descending data array. This then means that any remaining data in the original data set that lies after either of the aforementioned sets is ignored.

Example

```
Given the following array X,
 X:
  0.0000000000000000
  0.38905000686645508
  0.77815997600555420
  0.97268998622894287
  1.1671400070190430
  1.5559999942779541
  1.9448399543762207
  0.97259998321533203
  -9.9999997473787516E-006

This routine splits the array into the following ascending and
descending arrays.

ASCENDING:
  0.0000000000000000
  0.38905000686645508
  0.77815997600555420
  0.97268998622894287
  1.1671400070190430
  1.5559999942779541
  1.9448399543762207

DESCENDING:
  1.9448399543762207
  0.97259998321533203
  -9.9999997473787516E-006
```

Definition at line 1011 of file curvefit_calibration.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.35 curvefit_statistics::standard_deviation Interface Reference

Computes the corrected standard deviation of a data set.

Private Member Functions

- pure real(dp) function stdev_dbl (x)

  Computes the corrected standard deviation of a data set.

6.35.1 Detailed Description

Computes the corrected standard deviation of a data set.

Definition at line 56 of file curvefit_statistics.f90.

6.35.2 Member Function/Subroutine Documentation

6.35.2.1 pure real(dp) function curvefit_statistics::standard_deviation::stdev_dbl (real(dp), dimension(:), intent(in) *x*) `[private]`

Computes the corrected standard deviation of a data set.

Parameters

`in`	*x*	The data set.

Returns

The standard deviation of `x`.

Definition at line 366 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.36 curvefit_statistics::t_value Interface Reference

Computes the t-value (t-score) given a percentage of the area under the standard normal distribution curve.

Private Member Functions

- real(dp) function t_dist_score (alpha, n, err)

  Computes the t-value (t-score) given a percentage of the area under the standard normal distribution curve.

6.36.1 Detailed Description

Computes the t-value (t-score) given a percentage of the area under the standard normal distribution curve.

Definition at line 77 of file curvefit_statistics.f90.

6.36.2 Member Function/Subroutine Documentation

6.36.2.1 real(dp) function curvefit_statistics::t_value::t_dist_score (real(dp), intent(in) *alpha,* integer(i32), intent(in) *n,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the t-value (t-score) given a percentage of the area under the standard normal distribution curve.

Parameters

`in`	*alpha*	The percentage of the area under the curve. This value must be between 0 and 1 such that: 0 < alpha < 1.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `alpha` is does not satisfy: 0 < alpha < 1.

Returns

The t-socre or t-value.

Definition at line 538 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.37 curvefit_calibration::terminal_nonlinearity Interface Reference

Computes the terminal nonlinearity of a data set.

Private Member Functions

- real(dp) function term_nonlin (applied, measured, err)
  Computes the terminal nonlinearity of a data set.

6.37.1 Detailed Description

Computes the terminal nonlinearity of a data set.

Definition at line 66 of file curvefit_calibration.f90.

6.37.2 Member Function/Subroutine Documentation

6.37.2.1 real(dp) function curvefit_calibration::terminal_nonlinearity::term_nonlin (real(dp), dimension(:), intent(in) *applied,* real(dp), dimension(:), intent(in) *measured,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the terminal nonlinearity of a data set.

Parameters

`in`	*applied*	An N-element array containing the values applied to the measurement instrument.
`in`	*measured*	An N-element array containing the calibrated output of the instrument as a result of the values given in `applied`.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_ARRAY_SIZE_ERROR: Occurs if `applied` and `measured` are not the same size.

Returns

The terminal nonlinearity error.

Definition at line 267 of file curvefit_calibration.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_calibration.f90

6.38 curvefit_statistics::variance Interface Reference

Computes the sample variance of a data set.

Private Member Functions

- pure real(dp) function variance_dbl (x)

  Computes the sample variance of a data set.

6.38.1 Detailed Description

Computes the sample variance of a data set.

Definition at line 43 of file curvefit_statistics.f90.

6.38.2 Member Function/Subroutine Documentation

6.38.2.1 pure real(dp) function curvefit_statistics::variance::variance_dbl (real(dp), dimension(:), intent(in) *x*) [private]

Computes the sample variance of a data set.

Parameters

in	*x*	The data set.

Returns

The variance of `x`.

Remarks

To avoid overflow-type issues, Welford's algorithm is employed. A simple illustration of this algorithm can be found `here`.

Definition at line 202 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

6.39 curvefit_statistics::z_value Interface Reference

Computes the z-value (z-score) given a percentage of the area under the standard normal distribution curve.

Private Member Functions

- real(dp) function std_norm_dist_z_score (alpha, err)

  Computes the z-value (z-score) given a percentage of the area under the standard normal distribution curve.

6.39.1 Detailed Description

Computes the z-value (z-score) given a percentage of the area under the standard normal distribution curve.

Definition at line 70 of file curvefit_statistics.f90.

6.39.2 Member Function/Subroutine Documentation

6.39.2.1 real(dp) function curvefit_statistics::z_value::std_norm_dist_z_score (real(dp), intent(in) *alpha,* class(errors), intent(inout), optional, target *err*) `[private]`

Computes the z-value (z-score) given a percentage of the area under the standard normal distribution curve.

Parameters

`in`	*alpha*	The percentage of the area under the curve. This value must be between 0 and 1 such that: 0 $<$ alpha $<$ 1.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • CF_INVALID_INPUT_ERROR: Occurs if `alpha` is does not satisfy: 0 $<$ alpha $<$ 1.

Returns

The z-score or z-value by solving for z where: alpha = ERF(z / sqrt(2)), where ERF is the error function.

Definition at line 467 of file curvefit_statistics.f90.

The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/curvefit/src/curvefit_statistics.f90

Index