SLIM Curve

1.0

Generated by Doxygen 1.8.6

Mon Mar 17 2014 11:56:14

Contents

Chapter 1

SLIM-curve package for exponential curve fitting of spectral lifetime data.

SLIM Curve is a curve fitting library used for Fluorescent Lifetime Imaging or FLIM and Spectral Lifetime Imaging or SLIM. It is based on code developed by Paul Barber and the Advanced Technology Group at the Gray Institute for Radiation Oncology & Biology, University of Oxford, and used for FLIM functionality in his TRI2 (Time Resolved Imaging) software. It is also used in the LOCI SLIM Plugin project.

For exponential lifetime fitting there are two core algorithms within SLIM Curve: The first is a triple integral method that does a very fast estimate of a single exponential lifetime component. The second is a Levenberg-Marquardt algorithm or LMA that uses an iterative, least-squares-minimization approach to generate a fit. This works with single, double and triple exponential models, as well as stretched exponential. There is also code to perform 'global' analysis over a number of signals symultaneously (e.g. over an image), where the lifetimes can be considered constant across the data set, but the amplitudes are allowed to vary for each signal. There is also a completely generic global analysis function. A third algorithm is available to perform phasor analysis.

In addition there is a non-negative linear least squares algorithm that is useful for spectral unmixing in SLIM.

The code is written in C89 compatible C and is thread safe for fitting multiple pixels concurrently. Several files are provided as wrappers to call this library from Java code: EcfWrapper.c and .h provide a subset of function calls used by SLIM Plugin, these may be invoked directly from Java using JNA. In addition there is a Java CurveFitter project that provides a wrapper to the SLIM Curve code. This invokes the C code using JNI, with loci_curvefitter_SLIMCurveFitter.c and .h.

For further details, see: http://loci.wisc.edu/software/slim-curve

If you are familiar with the program TRI2, that uses SLIM Curve, this screenshot may help you to understand the meaning of the parameters.

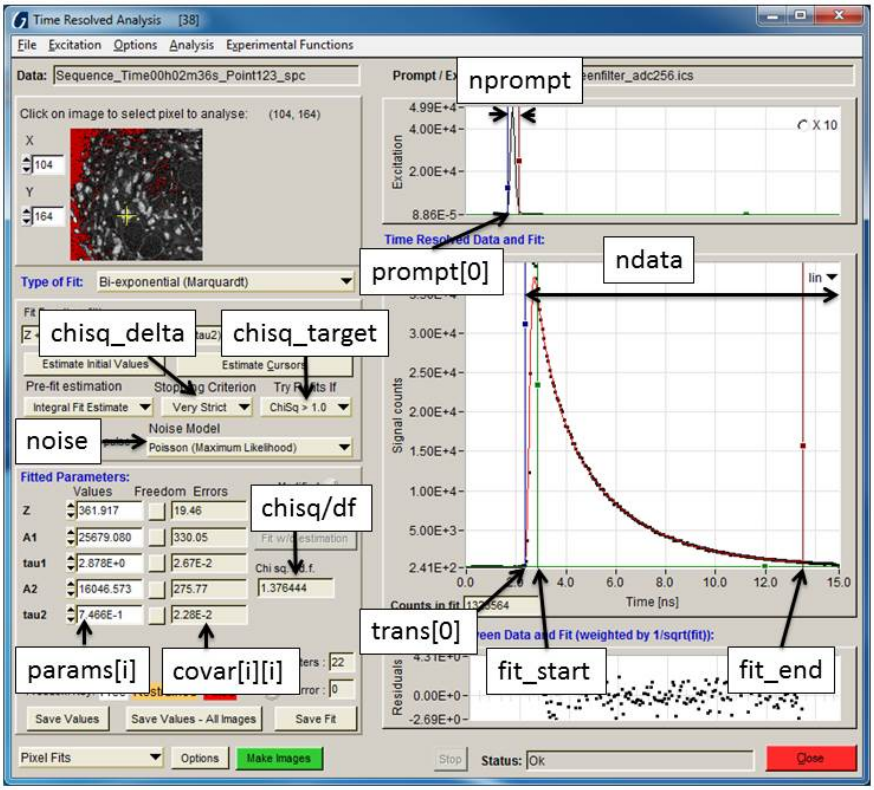

Figure 1.1: How some SLIM-Curve paramters are used in TRI2.

Library Contents:

Directory	Contents
src	source files
src/main/c	The source files for the SLIM Curve library
src/slim-curve-cmd/c	The source files for the stand alone executable wrapper for the library
test_files	dat and ini settings file for testing
src/main/c/doc	API documentation (Doxygen output)

To Build the Stand Alone Program using CMake and gcc under Linux:

Create a build folder, and cd to it

```
mkdir build
cd build
```

Run CMake

```
cmake ../CMakeLists.txt
```

Run make

```
make
```

To Run the Stand Alone Executable

Copy the executable to the test_files folder for convenience

```
cp slim-curve-cmd ../test_files
```

Run the program with the test files

```
cd ../test_files
./slim-curve-cmd test.ini transient.dat
```

Author

Paul Barber

Copyright

Chapter 2

File Index

2.1 File List

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

Chapter 3

File Documentation

3.1 Ecf.h File Reference

SLIM Curve - Exponential Curve Fitting Header.

Enumerations

- enum noise_type {
  NOISE_CONST, **NOISE_GIVEN**, **NOISE_POISSON_DATA**, **NOISE_POISSON_FIT**,
  NOISE_GAUSSIAN_FIT, **NOISE_MLE** }

  Noise Type.
- enum fit_type { **FIT_GLOBAL_MULTIEXP**, **FIT_GLOBAL_STRETCHEDEXP** }

  Fit Type for optimised exponential global analysis.
- enum restrain_type { **ECF_RESTRAIN_DEFAULT**, **ECF_RESTRAIN_USER** }

  Restrain Type.

Functions

- int GCI_triple_integral_fitting_engine (float xincr, float y[], int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], float *Z, float *A, float *tau, float *fitted, float *residuals, float *chisq, float chisq_target)

  The main entry point for Triple Integral or Rapid Lifetime Determination (RLD).
- int GCI_marquardt_fitting_engine (float xincr, float *trans, int ndata, int fit_start, int fit_end, float prompt[], int nprompt, noise_type noise, float sig[], float param[], int paramfree[], int nparam, restrain_type restrain, void(*fitfunc)(float, float[], float *, float[], int), float *fitted, float *residuals, float *chisq, float **covar, float **alpha, float **erraxes, float chisq_target, float chisq_delta, int chisq_percent)

  The main entry point for LM and MLE fitting.
- int **GCI_triple_integral** (float xincr, float y[], int fit_start, int fit_end, noise_type noise, float sig[], float *Z, float *A, float *tau, float *fitted, float *residuals, float *chisq, int division)
- int **GCI_triple_integral_instr** (float xincr, float y[], int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], float *Z, float *A, float *tau, float *fitted, float *residuals, float *chisq, int division)
- int **GCI_marquardt** (float x[], float y[], int ndata, noise_type noise, float sig[], float param[], int paramfree[], int nparam, restrain_type restrain, void(*fitfunc)(float, float[], float *, float[], int), float *fitted, float *residuals, float **covar, float **alpha, float *chisq, float chisq_delta, float chisq_percent, float **erraxes)
- int **GCI_marquardt_instr** (float xincr, float y[], int ndata, int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], float param[], int paramfree[], int nparam, restrain_type restrain, void(*fitfunc)(float, float[], float *, float[], int), float *fitted, float *residuals, float **covar, float **alpha, float *chisq, float chisq_delta, float chisq_percent, float **erraxes)
- void **GCI_marquardt_cleanup** (void)

- int GCI_marquardt_global_exps_instr (float xincr, float **trans, int ndata, int ntrans, int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], int ftype, float **param, int paramfree[], int nparam, restrain_type restrain, float chisq_delta, float **fitted, float **residuals, float chisq_trans[], float *chisq_global, int *df, int drop_bad_transients)

  Global analysis analysis function for exponential functions.
- int GCI_marquardt_global_generic_instr (float xincr, float **trans, int ndata, int ntrans, int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], float **param, int paramfree[], int nparam, int gparam[], restrain_type restrain, float chisq_delta, void(*fitfunc)(float, float[], float *, float[], int), float **fitted, float **residuals, float chisq_trans[], float *chisq_global, int *df)

  Global analysis analysis function for generic functions.
- int GCI_SPA_1D_marquardt (float x[], float y[], int ndata, noise_type noise, float sig[], float param[], int paramfree[], int nparam, restrain_type restrain, float chisq_delta, void(*fitfunc)(float, float[], float *, float[], int), int spa_param, int spa_nvalues, float spa_low, float spa_high, float chisq[], void(*progressfunc)(float))

  Support plane analysis.
- int GCI_SPA_2D_marquardt (float x[], float y[], int ndata, noise_type noise, float sig[], float param[], int paramfree[], int nparam, restrain_type restrain, float chisq_delta, void(*fitfunc)(float, float[], float *, float[], int), int spa_param1, int spa_nvalues1, float spa_low1, float spa_high1, int spa_param2, int spa_nvalues2, float spa_low2, float spa_high2, float **chisq, void(*progressfunc)(float))

  Support plane analysis for 2 parameters.
- int GCI_SPA_1D_marquardt_instr (float xincr, float y[], int ndata, int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], float param[], int paramfree[], int nparam, restrain_type restrain, float chisq_delta, void(*fitfunc)(float, float[], float *, float[], int), int spa_param, int spa_nvalues, float spa_low, float spa_high, float chisq[], float chisq_target, void(*progressfunc)(float))

  Support plane analysis with instrument response (prompt or IRF).
- int GCI_SPA_2D_marquardt_instr (float xincr, float y[], int ndata, int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], float param[], int paramfree[], int nparam, restrain_type restrain, float chisq_delta, void(*fitfunc)(float, float[], float *, float[], int), int spa_param1, int spa_nvalues1, float spa_low1, float spa_high1, int spa_param2, int spa_nvalues2, float spa_low2, float spa_high2, float **chisq, float chisq_target, void(*progressfunc)(float))

  Support plane analysis for 2 parameters with instrument response (prompt or IRF).
- int GCI_SPA_1D_marquardt_global_exps_instr (float xincr, float **trans, int ndata, int ntrans, int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], int ftype, float **param, int paramfree[], int nparam, restrain_type restrain, float chisq_delta, int drop_bad_transients, int spa_param, int spa_nvalues, float spa_low, float spa_high, float chisq_global[], int df[], void(*progressfunc)(float))

  Support plane analysis for exponential global analysis with instrument response (prompt or IRF).
- int GCI_SPA_2D_marquardt_global_exps_instr (float xincr, float **trans, int ndata, int ntrans, int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], int ftype, float **param, int paramfree[], int nparam, restrain_type restrain, float chisq_delta, int drop_bad_transients, int spa_param1, int spa_nvalues1, float spa_low1, float spa_high1, int spa_param2, int spa_nvalues2, float spa_low2, float spa_high2, float **chisq_global, int **df, void(*progressfunc)(float))

  Support plane analysis for exponential global with 2 parameters with instrument response (prompt or IRF).
- int GCI_SPA_1D_marquardt_global_generic_instr (float xincr, float **trans, int ndata, int ntrans, int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], float **param, int paramfree[], int nparam, int gparam[], restrain_type restrain, float chisq_delta, void(*fitfunc)(float, float[], float *, float[], int), int spa_param, int spa_nvalues, float spa_low, float spa_high, float chisq_global[], int df[], void(*progressfunc)(float))

  Support plane analysis for generic global analysis with instrument response (prompt or IRF).
- int GCI_SPA_2D_marquardt_global_generic_instr (float xincr, float **trans, int ndata, int ntrans, int fit_start, int fit_end, float instr[], int ninstr, noise_type noise, float sig[], float **param, int paramfree[], int nparam, int gparam[], restrain_type restrain, float chisq_delta, void(*fitfunc)(float, float[], float *, float[], int), int spa_param1, int spa_nvalues1, float spa_low1, float spa_high1, int spa_param2, int spa_nvalues2, float spa_low2, float spa_high2, float **chisq_global, int **df, void(*progressfunc)(float))

  Support plane analysis for generic global with 2 parameters with instrument response (prompt or IRF).
- int GCI_set_restrain_limits (int nparam, int restrain[], float minval[], float maxval[])

  Setting restraints.
- void GCI_multiexp_lambda (float x, float param[], float *y, float dy_dparam[], int nparam)

multi-exp predefined fitting model based on decay rates (lambdas).

- void GCI_multiexp_tau (float x, float param[], float *y, float dy_dparam[], int nparam)

  multi-exp predefined fitting model based on lifetimes (taus).
- void GCI_stretchedexp (float x, float param[], float *y, float dy_dparam[], int nparam)

  stretched-exp predefined fitting model.
- float ** GCI_ecf_matrix (long nrows, long ncols)

  Allocate a 2d matrix for ecf fitting.
- void GCI_ecf_free_matrix (float **m)

  Free a 2d matrix.
- void ECF_ExportParams_start (char path[])

  Start exporting fit details to a file for each fit.
- void ECF_ExportParams_stop (void)

  Stop exporting fit details to a file for each fit.

3.1.1 Detailed Description

SLIM Curve - Exponential Curve Fitting Header.

Definition in file Ecf.h.

3.1.2 Enumeration Type Documentation

3.1.2.1 enum fit_type

Fit Type for optimised exponential global analysis.

Chooses between a multi-exponential fit (FIT_GLOBAL_MULTIEXP) and the special 'stretched' exponential (FIT_GLOBAL_STRETCHEDEXP).

Definition at line 51 of file Ecf.h.

3.1.2.2 enum noise_type

Noise Type.

'NOISE_CONST' - every data point is assumed to have the same supplied variance.

'NOISE_GIVEN' - every data point can have an individual variance, given via a data array.

'NOISE_GAUSSIAN_FIT' - Variance for Gaussian distribution is used at all data points Variants based on the data or the fit.

'NOISE_POISSON_DATA and NOISE_POISSON_FIT' - Variance for Gaussian distribution is used with a lower limit of 15, this being the point where the Gaussian approximation begins to break down with Poissonian data. Variants based on the data or the fit.

'NOISE_MLE' - Maximum Likelihood Estimation through the use of the Poisson equation (Laurence and Chromy 2010).

Definition at line 45 of file Ecf.h.

3.1.2.3 enum restrain_type

Restrain Type.

Chooses the restrain type, either the default with very wide sensible limits (ECF_RESTRAIN_DEFAULT) or 'user' defined limits (ECF_RESTRAIN_USER). See GCI_set_restrain_limits.

Definition at line 56 of file Ecf.h.

3.1.3 Function Documentation

3.1.3.1 void ECF_ExportParams_start (char *path[]*)

Start exporting fit details to a file for each fit.

Parameters

in	*path*	The path of the file to use to save fit details.

3.1.3.2 void ECF_ExportParams_stop (void)

Stop exporting fit details to a file for each fit.

3.1.3.3 void GCI_ecf_free_matrix (float ** *m*)

Free a 2d matrix.

3.1.3.4 float** GCI_ecf_matrix (long *nrows,* long *ncols*)

Allocate a 2d matrix for ecf fitting.

3.1.3.5 int GCI_marquardt_fitting_engine (float *xincr,* float * *trans,* int *ndata,* int *fit_start,* int *fit_end,* float *prompt[],* int *nprompt,* noise_type *noise,* float *sig[],* float *param[],* int *paramfree[],* int *nparam,* restrain_type *restrain,* void(*)(float, float[], float *, float[], int) *fitfunc,* float * *fitted,* float * *residuals,* float * *chisq,* float ** *covar,* float ** *alpha,* float ** *erraxes,* float *chisq_target,* float *chisq_delta,* int *chisq_percent*)

The main entry point for LM and MLE fitting.

Uses GCI_marquardt_instr() to fit repeatedly until chisq_target is met or a maximum number of iterations are used. This can be used to fit any function through the 'fitfunc', but is primarily used for single, multi and stretched exponential fits. Predefined fitting models GCI_multiexp_tau and GCI_stretchedexp are provided elsewhere in the library.

Parameters

in	*xincr*	The time increment in between the values in the y array.
in	*trans*	The transient (time resolved) signal to be analysed, the 'data'.
in	*ndata*	The number of data points.
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit. Some data before this start index is required for convolution with the prompt.
in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*prompt*	The instrument response (IRF) or prompt signal to be used (optional, can pass NULL).
in	*nprompt*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in,out	*param*	An array of parameters, the order of which must match fitfunc. Provide parameter estimates, these are overridden with the fitted values.

in	*paramfree*	An array indicating which parameters are free (1), fixed (0)
in	*nparam*	The number of parameters.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*fitfunc*	Encodes the function to fit to the data, e.g. GCI_multiexp_tau.
out	*fitted*	An array containing values fitted to the data, the 'fit'. Fit points are coincident in time with the data points.
out	*residuals*	An array containing the difference between the data and the fit.
out	*chisq*	The resulting raw chi squared value of the fit. To get the reduced chisq, divide by the degrees of freedom (fit_start - fit_end - nparam)
out	*covar*	The covariance matrix. Allocate with a square matrix with GCI_ecf_matrix (nparam, nparam).
out	*alpha*	The alpha matrix. Allocate with a square matrix with GCI_ecf_matrix (nparam, nparam).
out	*erraxes*	The dimensions of the confidence ellipsoid of the chisq. See chisq_percent below. Allocate with a square matrix with GCI_ecf_matrix (nparam, nparam).
in	*chisq_target*	A raw chi squared value to aim for. If this value is reached fitting will stop. If you want to aim for a reduced chisq (say 1.1 or 1.0) you must multiply by the degree of freedom. (TRI2: "Try refits")
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*chisq_percent*	Defines the confidence interval when calculating the error axes, e.g. 95 %.

Returns

An error code, 0 = success.

3.1.3.6 int GCI_marquardt_global_exps_instr (float *xincr*, float ** *trans*, int *ndata*, int *ntrans*, int *fit_start*, int *fit_end*, float *instr[]*, int *ninstr*, noise_type *noise*, float *sig[]*, int *ftype*, float ** *param*, int *paramfree[]*, int *nparam*, restrain_type *restrain*, float *chisq_delta*, float ** *fitted*, float ** *residuals*, float *chisq_trans[]*, float * *chisq_global*, int * *df*, int *drop_bad_transients*)

Global analysis analysis function for exponential functions.

The main entry point for Global analysis with exponential functions. This assumes global exponential params (lifetimes) and local amplitudes.

Parameters

in	*xincr*	The time increment inbetween the values in the y array.
in	*trans*	A pointer to a matrix of transient (time resolved) signals to be analysed, the 'data'. Allocate with GCI_ecf_matrix.
in	*ndata*	The number of data points in each transient.
in	*ntrans*	The number of transient signals (pixels in a time resolved image).
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit. Some data before this start index is required for convolution with the prompt.
in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*instr*	The instrument reponse (IRF) or prompt signal to be used (optional, can pass NULL).
in	*ninstr*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.

in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in	*ftype*	The type of function to fit can be multi-exponential (FIT_GLOBAL_MULTIEXP) or stretched exponential (FIT_GLOBAL_STRETCHEDEXP). For other fitting functions use the global generic function.
in,out	*param*	A pointer to an array of parameter arrays, the order of which must match fitfunc.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters per fit.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
out	*fitted*	A pointer to a matrix containing values fitted to the data, the 'fit'. Fit points are coincident in time with the data points. Only the first row is used. Allocate with GCI_ecf_matrix with nrows=1.
out	*residuals*	A matrix containing the difference between the data and the fit. Only the first row is used. Allocate with GCI_ecf_matrix with nrows=1.
out	*chisq_trans*	An array of the resulting raw chi squared values of the fits. To get the reduced chisq, divide by the degrees of freedom (fit_start - fit_end - nparam).
out	*chisq_global*	The resulting raw chi squared value of the total global fit. To get the reduced chisq, divide by the degrees of freedom df.
out	*df*	The degrees of freedom.
in	*drop_bad_transients*	Remove individual transients from the fit that have a -ve return value from GCI_marquardt_global_exps_do_fit_single (default = 1)

Returns

An error code, 0 = success.

See Also

GCI_marquardt_global_generic_instr

3.1.3.7 int GCI_marquardt_global_generic_instr (float *xincr*, float ** *trans*, int *ndata*, int *ntrans*, int *fit_start*, int *fit_end*, float *instr[]*, int *ninstr*, noise_type *noise*, float *sig[]*, float ** *param*, int *paramfree[]*, int *nparam*, int *gparam[]*, restrain_type *restrain*, float *chisq_delta*, void(*)(float, float[], float *, float[], int) *fitfunc*, float ** *fitted*, float ** *residuals*, float *chisq_trans[]*, float * *chisq_global*, int * *df*)

Global analysis analysis function for generic functions.

The main entry point for Global analysis with generic functions.

Parameters

in	*xincr*	The time increment inbetween the values in the y array.
in	*trans*	A pointer to a matrix of transient (time resolved) signals to be analysed, the 'data'. Allocate with GCI_ecf_matrix.
in	*ndata*	The number of data points in each transient.
in	*ntrans*	The number of transient signals (pixels in a time resolved image).
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit. Some data before this start index is required for convolution with the prompt.

in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*instr*	The instrument reponse (IRF) or prompt signal to be used (optional, can pass NULL).
in	*ninstr*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in,out	*param*	A pointer to an array of parameter arrays, the order of which must match fitfunc. Allocate with GCI_ecf_matrix (ntrans, nparam). The global params will be the same for every transient.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters per fit.
in	*gparam*	An array marking which parameters are to be globally determined.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*fitfunc*	Encodes the function to fit to the data, e.g. GCI_multiexp_tau.
out	*fitted*	A pointer to a matrix containing values fitted to the data, the 'fit'. Fit points are coincident in time with the data points. Only the first row is used. Allocate with GCI_ecf_matrix with nrows=1.
out	*residuals*	A matrix containing the difference between the data and the fit. Only the first row is used. Allocate with GCI_ecf_matrix with nrows=1.
out	*chisq_trans*	An array of the resulting raw chi squared values of the fits. To get the reduced chisq, divide by the degrees of freedom (fit_start - fit_end - nparam).
out	*chisq_global*	The resulting raw chi squared value of the total global fit. To get the reduced chisq, divide by the degrees of freedom df.
out	*df*	The degrees of freedom.

Returns

An error code, 0 = success.

See Also

GCI_marquardt_global_exps_instr

3.1.3.8 void GCI_multiexp_lambda (float *x*, float *param[]*, float * *y*, float *dy_dparam[]*, int *nparam*)

multi-exp predefined fitting model based on decay rates (lambdas).

Parameters

in	*x*	The x value to evaluate the function at.
in	*param[]*	Array containing the parameters to use in the evaluation.
out	*y*	The value of the function at the x value.
out	*dy_dparam*	Array of dy/dparam values at this point
in	*nparam*	The number of parameters (may not be used if model has a fixed parameter number)

3.1.3.9 void GCI_multiexp_tau (float *x*, float *param[]*, float * *y*, float *dy_dparam[]*, int *nparam*)

multi-exp predefined fitting model based on lifetimes (taus).

Parameters

in	*x*	The x value to evaluate the function at.
in	*param[]*	Array containing the parameters to use in the evaluation.
out	*y*	The value of the function at the x value.
out	*dy_dparam*	Array of dy/dparam values at this point
in	*nparam*	The number of parameters (may not be used if model has a fixed parameter number)

3.1.3.10 int GCI_set_restrain_limits (int *nparam,* int *restrain[],* float *minval[],* float *maxval[]*)

Setting restraints.

Parameters

in	*nparam*	The number of parameters.
in	*restrain*	Array indicating which parameters to restrain (set non-zero to restrain).
in	*minval*	Array of minimum values for each parameter (only those for the restrained parameters set by the restrain array are used).
in	*maxval*	Array of maximum values for each parameter (only those for the restrained parameters set by the restrain array are used).

Returns

An error code, 0 = success.

3.1.3.11 int GCI_SPA_1D_marquardt (float *x[],* float *y[],* int *ndata,* noise_type *noise,* float *sig[],* float *param[],* int *paramfree[],* int *nparam,* restrain_type *restrain,* float *chisq_delta,* void(∗)(float, float[], float ∗, float[], int) *fitfunc,* int *spa_param,* int *spa_nvalues,* float *spa_low,* float *spa_high,* float *chisq[],* void(∗)(float) *progressfunc*)

Support plane analysis.

This fucntion will perform a number of fits with one fixed parameter that varies between the spa_low and spa_high values. The array of chisq values is returned that forms the 'support plane'.

Parameters

in	*x*	The x or time values at which the y data points are provided.
in	*y*	The transient (time resolved) signal to be analysed, the 'data'.
in	*ndata*	The number of data points.
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in,out	*param*	An array of parameters, the order of which must match fitfunc. Provide parameter estimates, these are overridden with the fitted values.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.

in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*fitfunc*	Encodes the function to fit to the data, e.g. GCI_multiexp_tau.
in	*spa_param*	The index to the param we are analysing with spa.
in	*spa_nvalues*	The number of values of the parameter we are to calculate the chisq value for.
in	*spa_low*	The lowest parameter value to use.
in	*spa_high*	The highest parameter value to use.
out	*chisq*	An array of resulting raw chi squared values. To get the reduced chisq, divide by the degrees of freedom (fit_start - fit_end - nparam)
in	*progressfunc*	A pointer to a function that may provide some feedback to the user on progress. A number between 0 and 1 is sent to this function after each iteration. (optional: may by NULL)

Returns

An error code, 0 = success.

See Also

GCI_SPA_2D_marquardt
GCI_SPA_1D_marquardt_instr
GCI_SPA_2D_marquardt_instr

3.1.3.12 int GCI_SPA_1D_marquardt_global_exps_instr (float *xincr*, float ** *trans*, int *ndata*, int *ntrans*, int *fit_start*, int *fit_end*, float *instr[]*, int *ninstr*, noise_type *noise*, float *sig[]*, int *ftype*, float ** *param*, int *paramfree[]*, int *nparam*, restrain_type *restrain*, float *chisq_delta*, int *drop_bad_transients*, int *spa_param*, int *spa_nvalues*, float *spa_low*, float *spa_high*, float *chisq_global[]*, int *df[]*, void(*)(float) *progressfunc*)

Support plane analysis for exponential global analysis with instrument response (prompt or IRF).

This fucntion will perform a number of fits with one fixed parameter that varies between the spa_low and spa_high values. The array of chisq values is returned that forms the 'support plane'.

Parameters

in	*xincr*	The time increment in between the values in the y array.
in	*trans*	A pointer to a matrix of transient (time resolved) signals to be analysed, the 'data'. Allocate with GCI_ecf_matrix.
in	*ndata*	The number of data points.
in	*ntrans*	The number of transient signals (pixels in a time resolved image).
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit. Some data before this start index is required for convolution with the prompt.
in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*instr*	The instrument response (IRF) or prompt signal to be used (optional, can pass NULL).
in	*ninstr*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in	*ftype*	The type of function to fit can be multi-exponential (FIT_GLOBAL_MULTIEXP) or stretched exponential (FIT_GLOBAL_STRETCHEDEXP). For other fitting functions use the global generic function.

in,out	*param*	A pointer to an array of parameter arrays, the order of which must match fitfunc.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*drop_bad_transients*	Remove individual transients from the fit that have a -ve return value from GCI_marquardt_global_exps_do_fit_single (default = 1)
in	*spa_param*	The index to the param we are analysing with spa.
in	*spa_nvalues*	The number of values of the parameter we are to calculate the chisq value for.
in	*spa_low*	The lowest parameter value to use.
in	*spa_high*	The highest parameter value to use.
out	*chisq_global*	The resulting array of raw chi squared values of the total global fit. To get the reduced chisq, divide by the degrees of freedom df.
out	*df*	The degrees of freedom.
in	*progressfunc*	A pointer to a function that may provide some feedback to the user on progress. A number between 0 and 1 is sent to this function after each iteration. (optional: may by NULL)

Returns

An error code, 0 = success.

See Also

GCI_SPA_2D_marquardt_global_exps_instr
GCI_SPA_1D_marquardt_global_generic_instr
GCI_SPA_2D_marquardt_global_generic_instr

3.1.3.13 int GCI_SPA_1D_marquardt_global_generic_instr (float *xincr*, float ** *trans*, int *ndata*, int *ntrans*, int *fit_start*, int *fit_end*, float *instr[]*, int *ninstr*, noise_type *noise*, float *sig[]*, float ** *param*, int *paramfree[]*, int *nparam*, int *gparam[]*, restrain_type *restrain*, float *chisq_delta*, void(*)(float, float[], float *, float[], int) *fitfunc*, int *spa_param*, int *spa_nvalues*, float *spa_low*, float *spa_high*, float *chisq_global[]*, int *df[]*, void(*)(float) *progressfunc*)

Support plane analysis for generic global analysis with instrument response (prompt or IRF).

This fucntion will perform a number of fits with one fixed parameter that varies between the spa_low and spa_high values. The array of chisq values is returned that forms the 'support plane'.

Parameters

in	*xincr*	The time increment in between the values in the y array.
in	*trans*	A pointer to a matrix of transient (time resolved) signals to be analysed, the 'data'. Allocate with GCI_ecf_matrix.
in	*ndata*	The number of data points.
in	*ntrans*	The number of transient signals (pixels in a time resolved image).
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit. Some data before this start index is required for convolution with the prompt.
in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*instr*	The instrument response (IRF) or prompt signal to be used (optional, can pass NULL).

in	*ninstr*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in,out	*param*	A pointer to an array of parameter arrays, the order of which must match fitfunc.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters.
in	*gparam*	An array marking which parameters are to be globally determined.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*fitfunc*	Encodes the function to fit to the data, e.g. GCI_multiexp_tau.
in	*spa_param*	The index to the param we are analysing with spa.
in	*spa_nvalues*	The number of values of the parameter we are to calculate the chisq value for.
in	*spa_low*	The lowest parameter value to use.
in	*spa_high*	The highest parameter value to use.
out	*chisq_global*	The resulting array of raw chi squared values of the total global fit. To get the reduced chisq, divide by the degrees of freedom df.
out	*df*	The degrees of freedom.
in	*progressfunc*	A pointer to a function that may provide some feedback to the user on progress. A number between 0 and 1 is sent to this function after each iteration. (optional: may by NULL)

Returns

An error code, 0 = success.

See Also

GCI_SPA_1D_marquardt_global_exps_instr
GCI_SPA_2D_marquardt_global_exps_instr
GCI_SPA_2D_marquardt_global_generic_instr

3.1.3.14 int GCI_SPA_1D_marquardt_instr (float *xincr*, float *y[]*, int *ndata*, int *fit_start*, int *fit_end*, float *instr[]*, int *ninstr*, noise_type *noise*, float *sig[]*, float *param[]*, int *paramfree[]*, int *nparam*, restrain_type *restrain*, float *chisq_delta*, void(∗)(float, float[], float ∗, float[], int) *fitfunc*, int *spa_param*, int *spa_nvalues*, float *spa_low*, float *spa_high*, float *chisq[]*, float *chisq_target*, void(∗)(float) *progressfunc*)

Support plane analysis with instrument response (prompt or IRF).

This fucntion will perform a number of fits with one fixed parameter that varies between the spa_low and spa_high values. The array of chisq values is returned that forms the 'support plane'.

Parameters

in	*xincr*	The time increment in between the values in the y array.
in	*y*	The transient (time resolved) signal to be analysed, the 'data'.
in	*ndata*	The number of data points.
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit. Some data before this start index is required for convolution with the prompt.

in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*instr*	The instrument response (IRF) or prompt signal to be used (optional, can pass NULL).
in	*ninstr*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in,out	*param*	An array of parameters, the order of which must match fitfunc. Provide parameter estimates, these are overridden with the fitted values.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*fitfunc*	Encodes the function to fit to the data, e.g. GCI_multiexp_tau.
in	*spa_param*	The index to the param we are analysing with spa.
in	*spa_nvalues*	The number of values of the parameter we are to calculate the chisq value for.
in	*spa_low*	The lowest parameter value to use.
in	*spa_high*	The highest parameter value to use.
out	*chisq*	An array of resulting raw chi squared values. To get the reduced chisq, divide by the degrees of freedom (fit_start - fit_end - nparam)
in	*chisq_target*	A raw chi squared value to aim for. If this value is reached fitting will stop. If you want to aim for a reduced chisq (say 1.1 or 1.0) you must multiply by the degree of freedom. (TRI2: "Try refits")
in	*progressfunc*	A pointer to a function that may provide some feedback to the user on progress. A number between 0 and 1 is sent to this function after each iteration. (optional: may by NULL)

Returns

An error code, 0 = success.

See Also

GCI_SPA_1D_marquardt
GCI_SPA_2D_marquardt
GCI_SPA_2D_marquardt_instr

3.1.3.15 int GCI_SPA_2D_marquardt (float *x[]*, float *y[]*, int *ndata*, noise_type *noise*, float *sig[]*, float *param[]*, int *paramfree[]*, int *nparam*, restrain_type *restrain*, float *chisq_delta*, void(∗)(float, float[], float ∗, float[], int) *fitfunc*, int *spa_param1*, int *spa_nvalues1*, float *spa_low1*, float *spa_high1*, int *spa_param2*, int *spa_nvalues2*, float *spa_low2*, float *spa_high2*, float ∗∗ *chisq*, void(∗)(float) *progressfunc*)

Support plane analysis for 2 parameters.

This fucntion will perform a number of fits with two fixed parameters that vary between the spa_low and spa_high values. The array of chisq values is returned that forms the 'support plane'.

Parameters

in	*x*	The x or time values at which the y data points are provided.

in	*y*	The transient (time resolved) signal to be analysed, the 'data'.
in	*ndata*	The number of data points.
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in,out	*param*	An array of parameters, the order of which must match fitfunc. Provide parameter estimates, these are overridden with the fitted values.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*fitfunc*	Encodes the function to fit to the data, e.g. GCI_multiexp_tau.
in	*spa_param1*	The index to first param we are analysing with spa.
in	*spa_nvalues1*	The number of values of first parameter we are to calculate the chisq value for.
in	*spa_low1*	The lowest first parameter value to use.
in	*spa_high1*	The highest first parameter value to use.
in	*spa_param2*	The index to second param we are analysing with spa.
in	*spa_nvalues2*	The number of values of second parameter we are to calculate the chisq value for.
in	*spa_low2*	The lowest second parameter value to use.
in	*spa_high2*	The highest second parameter value to use.
out	*chisq*	An matrix of resulting raw chi squared values. To get the reduced chisq, divide by the degrees of freedom (fit_start - fit_end - nparam), Allocate with chisq = (float $**$)malloc
in	*progressfunc*	A pointer to a function that may provide some feedback to the user on progress. A number between 0 and 1 is sent to this function after each iteration. (optional: may by NULL)

Returns

An error code, 0 = success.

See Also

GCI_SPA_1D_marquardt
GCI_SPA_1D_marquardt_instr
GCI_SPA_2D_marquardt_instr

3.1.3.16 int GCI_SPA_2D_marquardt_global_exps_instr (float *xincr*, float $**$ *trans*, int *ndata*, int *ntrans*, int *fit_start*, int *fit_end*, float *instr[]*, int *ninstr*, noise_type *noise*, float *sig[]*, int *ftype*, float $**$ *param*, int *paramfree[]*, int *nparam*, restrain_type *restrain*, float *chisq_delta*, int *drop_bad_transients*, int *spa_param1*, int *spa_nvalues1*, float *spa_low1*, float *spa_high1*, int *spa_param2*, int *spa_nvalues2*, float *spa_low2*, float *spa_high2*, float $**$ *chisq_global*, int $**$ *df*, void($*$)(float) *progressfunc*)

Support plane analysis for exponential global with 2 parameters with instrument response (prompt or IRF).

This fucntion will perform a number of fits with two fixed parameters that vary between the spa_low and spa_high values. The array of chisq values is returned that forms the 'support plane'.

Parameters

in	*xincr*	The time increment in between the values in the y array.
in	*trans*	A pointer to a matrix of transient (time resolved) signals to be analysed, the 'data'. Allocate with GCI_ecf_matrix.
in	*ndata*	The number of data points.
in	*ntrans*	The number of transient signals (pixels in a time resolved image).
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit. Some data before this start index is required for convolution with the prompt.
in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*instr*	The instrument response (IRF) or prompt signal to be used (optional, can pass NULL).
in	*ninstr*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in	*ftype*	The type of function to fit can be multi-exponential (FIT_GLOBAL_MULTIEXP) or stretched exponential (FIT_GLOBAL_STRETCHEDEXP). For other fitting functions use the global generic function.
in,out	*param*	A pointer to an array of parameter arrays, the order of which must match fitfunc.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*drop_bad_transients*	Remove individual transients from the fit that have a -ve return value from GCI_marquardt_global_exps_do_fit_single (default = 1)
in	*spa_param1*	The index to first param we are analysing with spa.
in	*spa_nvalues1*	The number of values of first parameter we are to calculate the chisq value for.
in	*spa_low1*	The lowest first parameter value to use.
in	*spa_high1*	The highest first parameter value to use.
in	*spa_param2*	The index to second param we are analysing with spa.
in	*spa_nvalues2*	The number of values of second parameter we are to calculate the chisq value for.
in	*spa_low2*	The lowest second parameter value to use.
in	*spa_high2*	The highest second parameter value to use.
out	*chisq_global*	The resulting array of raw chi squared values of the total global fit. To get the reduced chisq, divide by the degrees of freedom df.
out	*df*	The degrees of freedom.
in	*progressfunc*	A pointer to a function that may provide some feedback to the user on progress. A number between 0 and 1 is sent to this function after each iteration. (optional: may by NULL)

Returns

An error code, 0 = success.

See Also

GCI_SPA_1D_marquardt_global_exps_instr
GCI_SPA_1D_marquardt_global_generic_instr
GCI_SPA_2D_marquardt_global_generic_instr

3.1.3.17 int GCI_SPA_2D_marquardt_global_generic_instr (float *xincr,* float ** *trans,* int *ndata,* int *ntrans,* int *fit_start,* int *fit_end,* float *instr[],* int *ninstr,* noise_type *noise,* float *sig[],* float ** *param,* int *paramfree[],* int *nparam,* int *gparam[],* restrain_type *restrain,* float *chisq_delta,* void(*)(float, float[], float *, float[], int) *fitfunc,* int *spa_param1,* int *spa_nvalues1,* float *spa_low1,* float *spa_high1,* int *spa_param2,* int *spa_nvalues2,* float *spa_low2,* float *spa_high2,* float ** *chisq_global,* int ** *df,* void(*)(float) *progressfunc*)

Support plane analysis for generic global with 2 parameters with instrument response (prompt or IRF).

This fucntion will perform a number of fits with two fixed parameters that vary between the spa_low and spa_high values. The array of chisq values is returned that forms the 'support plane'.

Parameters

in	*xincr*	The time increment in between the values in the y array.
in	*trans*	A pointer to a matrix of transient (time resolved) signals to be analysed, the 'data'. Allocate with GCI_ecf_matrix.
in	*ndata*	The number of data points.
in	*ntrans*	The number of transient signals (pixels in a time resolved image).
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit. Some data before this start index is required for convolution with the prompt.
in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*instr*	The instrument response (IRF) or prompt signal to be used (optional, can pass NULL).
in	*ninstr*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in,out	*param*	A pointer to an array of parameter arrays, the order of which must match fitfunc.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters.
in	*gparam*	An array marking which parameters are to be globally determined.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*fitfunc*	Encodes the function to fit to the data, e.g. GCI_multiexp_tau.
in	*spa_param1*	The index to first param we are analysing with spa.
in	*spa_nvalues1*	The number of values of first parameter we are to calculate the chisq value for.
in	*spa_low1*	The lowest first parameter value to use.
in	*spa_high1*	The highest first parameter value to use.
in	*spa_param2*	The index to second param we are analysing with spa.
in	*spa_nvalues2*	The number of values of second parameter we are to calculate the chisq value for.
in	*spa_low2*	The lowest second parameter value to use.
in	*spa_high2*	The highest second parameter value to use.
out	*chisq_global*	The resulting array of raw chi squared values of the total global fit. To get the reduced chisq, divide by the degrees of freedom df.
out	*df*	The degrees of freedom.
in	*progressfunc*	A pointer to a function that may provide some feedback to the user on progress. A number between 0 and 1 is sent to this function after each iteration. (optional: may by NULL)

Returns

An error code, 0 = success.

See Also

GCI_SPA_1D_marquardt_global_exps_instr
GCI_SPA_2D_marquardt_global_exps_instr
GCI_SPA_1D_marquardt_global_generic_instr

3.1.3.18 int GCI_SPA_2D_marquardt_instr (float *xincr*, float *y[]*, int *ndata*, int *fit_start*, int *fit_end*, float *instr[]*, int *ninstr*, noise_type *noise*, float *sig[]*, float *param[]*, int *paramfree[]*, int *nparam*, restrain_type *restrain*, float *chisq_delta*, void(∗)(float, float[], float ∗, float[], int) *fitfunc*, int *spa_param1*, int *spa_nvalues1*, float *spa_low1*, float *spa_high1*, int *spa_param2*, int *spa_nvalues2*, float *spa_low2*, float *spa_high2*, float ∗∗ *chisq*, float *chisq_target*, void(∗)(float) *progressfunc*)

Support plane analysis for 2 parameters with instrument response (prompt or IRF).

This fucntion will perform a number of fits with two fixed parameters that vary between the spa_low and spa_high values. The array of chisq values is returned that forms the 'support plane'.

Parameters

in	*xincr*	The time increment in between the values in the y array.
in	*y*	The transient (time resolved) signal to be analysed, the 'data'.
in	*ndata*	The number of data points.
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit. Some data before this start index is required for convolution with the prompt.
in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*instr*	The instrument response (IRF) or prompt signal to be used (optional, can pass NULL).
in	*ninstr*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
in,out	*param*	An array of parameters, the order of which must match fitfunc. Provide parameter estimates, these are overridden with the fitted values.
in	*paramfree*	An array indicating which parameters are free (1), fixed (0).
in	*nparam*	The number of parameters.
in	*restrain*	Parameter restrain_type. Normally use ECF_RESTRAIN_DEFAULT. Use ECF_RESTRAIN_USER if restraining parameters has been setup via GCI_set_restrain_limits.
in	*chisq_delta*	An individual fit will continue if the chisq value changes by more then this amount. Try 1E-5. (TRI2: "Stopping Criterion")
in	*fitfunc*	Encodes the function to fit to the data, e.g. GCI_multiexp_tau.
in	*spa_param1*	The index to first param we are analysing with spa.
in	*spa_nvalues1*	The number of values of first parameter we are to calculate the chisq value for.
in	*spa_low1*	The lowest first parameter value to use.
in	*spa_high1*	The highest first parameter value to use.
in	*spa_param2*	The index to second param we are analysing with spa.
in	*spa_nvalues2*	The number of values of second parameter we are to calculate the chisq value for.
in	*spa_low2*	The lowest second parameter value to use.
in	*spa_high2*	The highest second parameter value to use.
out	*chisq*	An matrix of resulting raw chi squared values. To get the reduced chisq, divide by the degrees of freedom (fit_start - fit_end - nparam), Allocate with `chisq = (float **)malloc`
in	*chisq_target*	A raw chi squared value to aim for. If this value is reached fitting will stop. If you want to aim for a reduced chisq (say 1.1 or 1.0) you must multiply by the degree of freedom. (TRI2: "Try refits")
in	*progressfunc*	A pointer to a function that may provide some feedback to the user on progress. A number between 0 and 1 is sent to this function after each iteration. (optional: may by NULL)

Returns

An error code, 0 = success.

See Also

GCI_SPA_1D_marquardt
GCI_SPA_2D_marquardt
GCI_SPA_1D_marquardt_instr

3.1.3.19 void GCI_stretchedexp (float *x*, float *param[]*, float * *y*, float *dy_dparam[]*, int *nparam*)

stretched-exp predefined fitting model.

Parameters

in	*x*	The x value to evaluate the function at.
in	*param[]*	Array containing the parameters to use in the evaluation.
out	*y*	The value of the function at the x value.
out	*dy_dparam*	Array of dy/dparam values at this point
in	*nparam*	The number of parameters (may not be used if model has a fixed parameter number)

3.1.3.20 int GCI_triple_integral_fitting_engine (float *xincr*, float *y[]*, int *fit_start*, int *fit_end*, float *instr[]*, int *ninstr*, noise_type *noise*, float *sig[]*, float * *Z*, float * *A*, float * *tau*, float * *fitted*, float * *residuals*, float * *chisq*, float *chisq_target*)

The main entry point for Triple Integral or Rapid Lifetime Determination (RLD).

Uses GCI_triple_integral_*() to fit repeatedly with different integration periods until chisq_target is met or a maximum number of iterations are used.

Parameters

in	*xincr*	The time increment inbetween the values in the y array.
in	*y*	The transient (time resolved) signal to be analysed, the 'data'.
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit.
in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
in	*instr*	The instrument reponse (IRF) or prompt signal to be used (optional, can pass NULL).
in	*ninstr*	The number of data points in the prompt (ignored if prompt = NULL).
in	*noise*	The noise_type to be used.
in	*sig*	The standard deviation at each data point in y if noise_type NOISE_GIVEN is used (optional, can pass NULL).
out	*Z*	The returned background value from the fit.
out	*A*	The returned amplitude value from the fit.
out	*tau*	The returned lifetime value from the fit.
out	*fitted*	An array containing values fitted to the data, the 'fit'. Fit points are coincident in time with the data points.
out	*residuals*	An array containing the difference between the data and the fit.
out	*chisq*	The resulting raw chi squared value of the fit. To get the reduced chisq, divide by the degrees of freedom (fit_start - fit_end - nparam)

in	*chisq_target*	A raw chi squared value to aim for. If this value is reached fitting will stop. If you want to aim for a reduced chisq (say 1.1 or 1.0) you must multiply by the degree of freedom. (TRI2: "Try refits")

Returns

An error code, 0 = success.

3.2 GCI_Lsqnonneg.h File Reference

SLIM Curve - Non-negative Least Squares Header.

Functions

- int GCI_lsqnonneg (double ∗∗A, double ∗b, double ∗x, int m, int n, int preserve, double ∗rnorm, double ∗lambda)

  This function solves the non-negative least squares problem.

3.2.1 Detailed Description

SLIM Curve - Non-negative Least Squares Header.

Definition in file GCI_Lsqnonneg.h.

3.2.2 Function Documentation

3.2.2.1 int GCI_lsqnonneg (double ∗∗ *A*, double ∗ *b*, double ∗ *x*, int *m*, int *n*, int *preserve*, double ∗ *rnorm*, double ∗ *lambda*)

This function solves the non-negative least squares problem.

Minimise |Ax-b| subject to x >= 0 (where |v| is the 2-norm of the vector v). !!! NB: A and B will both be modified unless preserve is non-zero (see below).

Parameters

in	*A*	An m x n matrix, in the form double A[n][m], so the columns of A are A[0], A[1], ..., A[n-1]
in	*b*	The m-vector
out	*x*	The solution
in	*m*	The size of the 'm' dimensions of A
in	*n*	The size of the 'n' dimensions of A
in	*preserve*	Copy A and b before solving the problem so they are not modified
out	*rnorm*	The value of |Ax-b| with the determined x if the function was successful or if the iteration count was exceeded. This can be NULL.
out	*lambda*	An n-vector which will contain the dual vector on completion (that is, the Lagrange multipliers). This can be NULL.

Returns

The return value will be 0 on success, and negative if a problem occurred:

- -1: m > MAX_EQNS or m <= 0
- -2: n > MAX_VARS or n <= 0
- -3: iteration count exceeded: more than 3∗n iterations performed
- -4: memory allocation problems

3.3 GCI_Phasor.h File Reference

SLIM Curve - Phasor analysis Header.

Functions

- int GCI_Phasor (float xincr, float y[], int fit_start, int fit_end, float *Z, float *u, float *v, float *taup, float *taum, float *tau, float *fitted, float *residuals, float *chisq)

  Take transient data and perform phasor analysis, returning u, v, and estimated lifetime.
- double GCI_Phasor_getPeriod ()

  Get the phasor period that was calculated and used in the last call to GCI_Phasor.

3.3.1 Detailed Description

SLIM Curve - Phasor analysis Header. Classic Phasor or Polar approach to FLIM. See Clayton 2004 or Leray 2008.

Definition in file GCI_Phasor.h.

3.3.2 Function Documentation

3.3.2.1 int GCI_Phasor (float *xincr*, float *y[]*, int *fit_start*, int *fit_end*, float * *Z*, float * *u*, float * *v*, float * *taup*, float * *taum*, float * *tau*, float * *fitted*, float * *residuals*, float * *chisq*)

Take transient data and perform phasor analysis, returning u, v, and estimated lifetime.

u = integral(data(t) * cos(wt)) dt / integral(data(t)) dt v = integral(data(t) * sin(wt)) dt / integral(data(t)) dt

tau phi = taup = 1/w * (v/u) tau mod = taum = 1/w * sqrt(1/(u^2 + v^2) - 1) tau average = tau = (taup + taum) / 2

Parameters

in	*xincr*	The time increment inbetween the values in the y array.
in	*y*	The transient (time resolved) signal to be analysed, the 'data'.
in	*fit_start*	The index into the y array marking the start to the data to be used in the fit.
in	*fit_end*	The index into the y array marking the end of the data to be used in the fit.
out	*Z*	must have been estimated previously so that it can be subtracted from the data here.
out	*u*	The 'horizontal' phasor coordinate.
out	*v*	The 'vertical' phasor coordinate.
out	*taup*	The lifetime calculated from the phase change.
out	*taum*	The lifetime calculated from the amplitude change (the demodulation).
out	*tau*	The average of the other taus.
out	*fitted*	An array containing values fitted to the data, the 'fit'. Fit points are coincident in time with the data points.
out	*residuals*	An array containing the difference between the data and the fit.
out	*chisq*	The resulting reduced chi squared value of the fit.

Returns

An error code, 0 = success.

3.3.2.2 double GCI_Phasor_getPeriod ( )

Get the phasor period that was calculated and used in the last call to GCI_Phasor.

Returns

The period in the time units of xinc.

Index