recoos 2.2

$Date: 2002/11/05 18:31:18 $
Muon and neutrino reconstruction
for underwater Cerenkov telescopes

Ole Streicher, Christopher Wiebusch

This software wants to be distributed freely, but Ole Streicher and Christopher Wiebusch hold the copyright. Do not remove copyright notes from the files. We think the GNU Public Licence is a good basic for scientific projects, so we will put this software under the GPL.

This manual incarnates in a `postscript`, `HTML` (WWW) and in an `Emacs-info` version.

In case of bugs, questions, remarks, suggestions or flowers, please send them per e-mail to:
`siegmund@ifh.de (Siegmund mailing list)`
`ole@ifh.de (Ole Streicher)`
`wiebusch@ifh.de (Christopher Wiebusch)`

$Id: recoos.texi,v 1.39 2002/11/05 18:31:18 cph Exp $

1 How to use this manual

First a warning: This manual will not teach you the physics in the algorithms described here. People not having a recoos guru at their disposition are encouraged to have a look at the following literature.

Muon reconstruction with AMANDA
: C. Wiebusch, Proceedings of Workshop on Simulation and Analysis Methods for Large Neutrino Telescopes, DESY ZEUTHEN, Germany July 6-9, 1998 edited by C. Spiering DESY Zeuthen, Zeuthen, Germany

Direct-Walk II
: P. Steffen, AMANDA Internal Report 20020201

Simplex V Powell's Minimization of Pandel Timing Likelihood
: `http://amanda.physics.wisc.edu/steele/academic/research/simplex_v_powell/simplex_v_powell.html`, D. Steele

`rdmc` **Manual**
: available under `siegmund/rdmc/rdmc.ps` after `make recoos_doc`.

For recoos, the impatient reader will want to read the examples, See Chapter 6 [Examples], page 32. Also, since programmers seem to tend only updating the online help, check also `recoos -hh |& less` from time to time.

2 What is recoos?

recoos is a program package used for reconstruction and quality analysis. The package consist of three programs:

`recoos` See Chapter 3 [recoos], page 3
: The reconstruction program implementing Chi-square and likelihood muon track reconstructions for AMANDA and BAIKAL.

`munt` See Chapter 4 [munt], page 24
: A program booking a `CWN` N-tuple with the `muff` criteria for analysis with `PAW`.

`muff` See Chapter 5 [muff], page 31
: A filter program implementing various quality criteria for the selection of events.

The programs are written on basis of the `rdmc`-IO library and are intended to work as filters in the data stream. They use the `rdmc` memory data structures and may use all data formats provided by the `rdmc` library. Their functionality is controlled with command line options.

3 recoos

3.1 Usage

The program is invoked from the command line via

```
recoos [arguments] [file]
```

If a filename is omitted, `recoos` reads from the standard input. Following options may be given:

`-n <xxx>` Read only <xxx> events

`-s <xxx>` Skip to the first <xxx> events

`-o file` Output file. If no file is given, `recoos` writes the data stream to standard output.

`-h[h]` This help [and more help]

`-v[v]` Verbose mode [more verbose output]. The verbose output is written to standard error. Additional `-v` increase the output.

`-r method` Reconstruction method. This flag selects the kind of reconstruction, e.g. simple muon reconstruction. See Section 3.1.2 [Reconstruction methods], page 4.

`-i method` First guess method. This flag selects the algorithm used for the starting values of the reconstruction, e.g. a line-fit. See Section 3.1.3 [First Guess methods], page 7.

`-m method` Minimization method. This flag selects the package used for function minimisation, e.g. `Minuit`. See Section 3.1.1 [Minimisation methods], page 3.

`-z type` Minimization function. This flag selects the type of minimisation function, e.g. a Pandel function for Amanda. See Section 3.1.4 [Minimisation functions], page 11.

`-l type` Parametrization type. This flag selects which parametrization (x,y,z,zenith, azimuth or other) is used. See Section 3.1.5 [Parametrisations], page 13.

`-x par1,par2...`
Use other than the default free parameter set, f.e. to include energy reconstruction. See Section 3.1.5 [Parametrisations], page 13.

`-X opt` Extended reco options. This flag is used to set various additional options, e.g. the rejection of specific hits during the reconstruction See Section 3.1.6 [Extended reconstruction options], page 14

`-p para` This flag allows to adjust additional parameters, which are passed to the reconstruction, e.g. the trigger. See Section 3.1.7 [Reconstruction parameters], page 15

`-y` This flag calls in the usual hit cleaning options. Please confer `rdmc` manual for a listing of available options.

You may get a detailed list of available parameters for each option if you specify an `"h"` as option parameter.

3.1.1 Minimisation methods (-m)

Recoos gives you the possibility to choose the minimisation procedure. Currently, three different methods are implemented:

Minuit (-m m).
This is taken from the CERN libraries. It is hard wired to use MIGRAD minimisation with maximal 2000 loops. It allows to calculate some extended values which may be useful for filterung good events (like the error ellipse), but this feature is a bit buggy and may crash if you don't use the standard parametrisation. The Minuit strategy may be changed using the additional option `-p m=<strategy>` with `<strategy>` being 0 (bad and fast, default), 1 or 2 (best and slowest).

Powells (-m p).
This is the "Direction Set Method" from the "Numerical Recipes" book, using maximal 2000 iterations and a fractional tolerance of 1e-9. See "Numeric Recipes, 2nd edition", pg.406ff for further reference.

Simplex (-m x)
The use of simplexes is also described well in the "Numerical Recipes" book. The simplex minimiser does not depend on knowledge of gradients and so is well-suited to the kinky AMANDA probability densities. It is also the current speed champion of all the minimisers. Please note that if you use simplex, you are encouraged to explicitly set the initial step sizes as the default powell step sizes are not of the correct scale.

Simulated annealing (-m s).
Also stolen from the "Numeric Recipes". Parameters are: Temperature 1.0, decrease 0.05, fractional tolerance 1e-8, 50 iterations. See pg.443ff of the above mentioned book for further reference.

3.1.2 Reconstruction methods (-r)

Here you may choose which "high-level" method you want to use for reconstruction. This describes how everything is glued together:

Muon Reconstruction (-r m).
Default straightforward reconstruction: After calculating a first guess (from the `-i` option) start a minimisation of the sum of all minimisation functions (`-z` option).

Neutrino/fake search (-r f).
Like the previous, but if an up going track is found or the reconstruction resulted in a bad minimised track (minimal function value above a certain limit), the reconstruction is repeated with maximal 25 down going tracks (with an exponential cos(theta) distribution). The returned track is the first good minimised down going track, or the best fit (by the minimal function value) at all.

Only first guess (-r n).
Just make a first guess according to the `-i` option and return the result. No minimisation.

Paraboloid fit (-r p).
This fit needs a track around which the angular error is to be determined. Around this direction a number (default is 3) of ellipses is placed (in zenith/(track) azimuth space, the coordinate perpendicular to zenith is called "track azimuth"). On each ellipse 8 points are chosen and the minimizer is run to obtain the (-log) likelihood. The variables xyzza should be chosen. zenith and azimuth should be fixed. x,y,z should be free, if one wants to include the correlations (recommended). The dimensions of the outmost ellipse can be chosen (default is a circle with 10 degrees diameter). Additionally the likelihood values and angle coordinates can be dumped to stdout in order to visualize the paraboloid fits (-D option - recommended to be used in conjunction with -n xxx to limit the output and with -o file option to separate f2k stuff from data).

New options are

```
-rp to choose the paraboloid fit
-p c=xxx for the number of rings
-p d=x:y for the diameters of the outmost ring
-p D=[01] for output dump
```

An example:

```
recoos -v \ (verbose output)
-mp \ (powell minimization ok, -mx simplex not yet tested)
-rp \ ("p"araboloid fit)
-p c=3 \ (number of "c"ircles/ellipses)
-p d=10:10 \ (zenith:azimuth diameters)
-p D=0 \ (D=0 no stdout dump (can be omitted) D=1 does dump)
-X g=f -X s=c \
-z a_upandel \ (only tested llh, but algorithm is independent)
-l xyzza \ (only tested parameterization)
-x x,y,z \ (strongly recommended to fix zenith, azimuth)
-X o=2 \ (use same physics as in original fit!)
-p L=0. -p n=1000:5000 -p j=15 -p a=96. -p t=1:5 \
-y U=2 -if -p f=2 \ (which fit to base the error upon)
your-input-file \ (input)
-o your-ouput-file (output)
```

the fit fills the FIT line with the track from the minimum of the paraboloid. The more important error information is given in the corresponding user line.

The name of USER line is "prbl_xx", where xx is the two-digit number of the fit the error of which is to be found. There are 21 entries in this line, all of which begin with a "p" and the two digit number xx. They are limited in length to 8 characters to comply with munt standards. Hence they are some- what unreadable:

```
"pxxsellh" // likelihood-value of basic fit
"pxxfllh" // likelihood value at paraboloid minimum
"pxxfze" // theta value at paraboloid minimum
"pxxfaz" // phi value at paraboloid minimum
"pxxfzeof" // if basic fit is rotated to equator and called (0,0), this
is theta offset of paraboloid min.
"pxxfazof" // if basic fit is rotated to equator and called (0,0), this
is phi offset of paraboloid min.
"pxxfcu11" // 1-1 component of paraboloid curvature matrix
"pxxfcu12" // 1-2 component of paraboloid curvature matrix
"pxxfcu22" // 2-2 component of paraboloid curvature matrix
"pxxsigze" // expected error in zenith
"pxxsigaz" // expected error in track azimuth
"pxxcovaz" // expected covariance between zenith an track azimuth
"pxxerr1" // after rotation of confidence ellipse: bigger axis (that
corresponds to variance not std. dev)
"pxxerr2" // after rotation of confidence ellipse: smaller axis (that
corresponds to variance not std. dev)
```

"pxxangle" // rotation angle applied to transfom to uncorrelated coordinates

"pxxchi2" // squared differences of paraboloid llh and rdmc llh over all points

"pxxdetM" // determinant of 2d error matrix

"pxxtzeof" // true zenith rotated

"pxxtazof" // true azumith rotated

"pxxtze" // true zenith not rotated

"pxxtaz" // true azumith not rotated

Remarks

1. The simplex minimizer had some problems, if there were fixed parameters. This problem only arises if there is a fixed parameter between two free ones. With the -lxyzza option and -x x,y,z that shouldn't pose a problem though.

True grid fit (-r t).

This fit distributes a given number (num: -p i=num) of points on the unit sphere. These points determine the angle in a track hypothesis. The x,y,z coordinates are minimized by the minimizer. Variables are filled as by the random iterative (standard) gridsearch algorithm - with the second best likelihood in the other hemisphere as described there.

New options are

-rt to choose the truegrid fit

-p i=xxx for the number of points

-p D=[01] for output dump

An example:

```
recoos -v \
-mp \
-rt \ ("t"ruegrid reconstruction)
-p i=100 \ (with 100 points on the unit sphere)
-p D=0 \ (as with paraboloid reco - no dump)
-X g=f -X s=c \
-z a_upandel -l xyzza \
-x x,y,z \ (zenith and azimuth are fixed by the grid
-X o=2 \
-p L=0. -p n=1000:5000 -p j=15 -p a=96. -p t=1:5 \
-y U=2 \ (some hit cleaning should be used)
-if -p f=2 \ (this information is only used in a technical way)
your-input-file \
"|" \ pipe things into a single minimization step
recoos -v \
-mp -rm \
-X g=f -X s=c \
-z a_upandel -l xyzza \
-x x,y,z,zenith,azimuth \
-X o=2 \
```

```
-p L=0. -p n=1000:5000 -p j=15 -p a=96. -p t=1:5 \
-y U=2 -if -p f=3 \
-o your-output-file
```

Remarks:

1. It is recommend to run a single iteration of the full minimization process after the truegrid search. That way the angle can get optimized as well.

2. The program should run ok with both the powell and the simplex minimizers. See remark above.

3. With i=100 the program runs about 20%-40% slower than an 16fold iterative search.

Multiple reconstruction (`-r g`).
Iterative fit. Basically like `-r m`, but takes several approaches to avoid local minima in the likelihood landscape. See See Section 3.1.7 [Reconstruction parameters], page 15 how to use this.

3.1.3 First Guess methods (`-i`)

The reconstruction needs some starting point to make a good minimisation of the likelihood. This starting point can be selected with the `-i` option. Implemented first guesses are available for

3.1.3.1 Guesses for track like events

Direct Walk (`-i s`).
Peter Steffen's combinatoric track finder. See literature of this document for a description.

`res->val[JK_CUTFLAG]`
: Number of track hits in event time window

`res->val[JK_CHI2]`
: Number of hits in event time window

`res->val[JK_RCHI2]`
: N_hit, the hits assigned to the track

`res->val[JK_SIGTH]`
: Track length

`res->val[JK_PROB]`
: Spread of hits along the track

`res->val[JK_COVMIN]`
: Average OM distance from track

`res->val[JK_COVMAX]`
: Number of track candidates

Line Fit (`-i m`).
Also called "Stenger" (by its first author). This is the default. It just makes a linear fit between the space and time coordinates of the event. Note that this is sometimes erroneous called "plane wave fit", but the plane wave fit is different from that (see below). Minimises

$$\chi^2 = \sum_{i=1}^{N} a_i^w \left(\vec{x}_i - \vec{X}_0 - \vec{v} \cdot t_i \right)^2$$

where N is the number of used modules, a_i the hit amplitude, w the weighting exponent (`-p w=`w). The starting point of the track is $\vec{X}_0$, the resulting direction $\vec{n} = (\cos\phi \sin\Theta, \sin\phi \sin\Theta, \cos\Theta)$, and the fitted particle speed is given by $\vec{v} = v \cdot \vec{n}$. This first guess gives the following results:

```
rec->x = xmean - vx * t_mean;
rec->y = ymean - vy * t_mean;
rec->z = zmean - vz * t_mean;
rec->px = vx / vabs;
rec->py = vy / vabs;
rec->pz = vz / vabs;
res->val[JK_CUTFLAG]=1;
res->val[JK_RCHI2]=chi2/(ampsum -2.0);
res->val[JK_SIGTH]=vz;
res->val[JK_PROB]=vabs;
res->val[JK_COVMIN]=ampsum/((float)event->nhits);
res->val[JK_COVMAX]=maxamp;
```

3.1.3.2 Guesses for shower like events

COG (`-i c`).
Centre of gravity of hits (Amp weighted). Azimuth is set to `M_PI/2`, cosine of the zenith to -1.0. Does not touch `JK_*`.

```
rec->x = cog[0];
rec->y = cog[1];
rec->z = cog[2];
rec->phi = M_PI/2; /* dummy */
rec->costh = -1.0;
```

Tensor of Inertion of hits (`-i i`).
Calculates the eigenvalues of the tensor of inertia, which is defined as follows:

$$I^{kl} = \sum_i a_i^w \cdot (\delta^{kl} \vec{r}_i^2 - r_i^k \cdot r_i^l)$$

where the $\vec{r}_i$ are the location of the hit modules. The three eigenvalues $I_1 < I_2 < I_3$ can be interpreted as cigar or pancake configuration of hits. If I_1/I_3 is small, the event is rather track like, if I_1/I_3 is around 1, the event is rather spherical and thus shower like.

```
rec->x = rc[0] (=cogx);
rec->y = rc[1] (=cogy);
rec->z = rc[2] (=cogz);
rec->e = 8000.;
rec->px = vpoint*v[min][1] (x-direction of the main axis);
rec->py = vpoint*v[min][2] (y-direction of the main axis);
rec->pz = vpoint*v[min][3] (z-direction of the main axis);
chi2=dmin/(d[1]+d[2]+d[3]);
res->val[JK_RCHI2]=chi2;
```

```
res->val[JK_PROB]=dmin (=I1);
res->val[JK_SIGTH]=vdir;
res->val[JK_COVMIN]=I2;
res->val[JK_COVMAX]=I3;
```

Time flow in the event (`-i t`).

Calculates the following:

$$N = \sum_i a_i^w$$

$$t_0 = 1/N \sum_i a_i^w t_i$$

$$\vec{r}_0 = 1/N \sum_i a_i^w \vec{x}_i$$

$$\vec{v} = 1/N \sum_i a_i^w (\vec{r}_i - \vec{r}_0)/(t_i - t_0)$$

```
rec->x = rc[0] (=cog x);
rec->y = rc[1] (=cog y);
rec->z = rc[2] (=cog z);
rec->t = t0 (=cog t);
rec->px = v[0];
rec->py = v[1];
rec->pz = v[2];
res->val[JK_CUTFLAG]=1
res->val[JK_RCHI2]=chi2/ampsum;
res->val[JK_PROB]=vabs;
res->val[JK_COVMIN]=ampsum/((float)ev->nhits);
res->val[JK_COVMAX]=maxamp;
```

Plane Wave (`-i q`).

The "real" plane wave fit. Since this gives only the track direction, additionally the Centre of Gravity of all hits is calculated and used as track vertex. Minimises

$$\chi^2 = \sum_{i=1}^{N} a_i^w \left(\vec{x}_i \cdot \vec{n} - v \cdot (t_i - t_0)\right)^2$$

where N is the number of used modules, a_i the hit amplitude, w the weighting exponent (`-p w=`w). The resulting direction is $\vec{n} = (\cos\phi \sin\Theta, \sin\phi \sin\Theta, \cos\Theta)$, and the fitted speed is given by v.

Dipole moment (`-i d`).

Calculates dipole moment of hits. A. Biron writes: "The idea is to suppress fakes from two coincident muons. Suppose you have first a down-going muon in the bottom of the detector. Then the x_i-x_i-1 are all down going. Later you have a downgoing muon in the top of the detector. Then you have one up going term x_i-x_i-1 connecting the hit clusters of the two muons. All remaining x_i-x_i-1 are then again down going from the second muon. In total you get a strong indication for down going." Performs time ordering of the hits.

$$g_i = (1/2 \cdot (a_i + a_{i-1}))^w; \qquad i = 1 \ldots \texttt{nhits}$$

$$N = \sum_i g_i$$

$$d\vec{x}_i = \vec{x}_i - \vec{x}_{i-1} * g_i$$

$$\vec{v} = \sum d\vec{x}/(d\vec{x})^2$$

$$\vec{r} = \sum 1/2 \cdot (\vec{x}_i + \vec{x}_{i-1}) \cdot g_i$$

```
res->val[JK_RCHI2]=chi2/(a_sum - 2.0);
res->val[JK_RCHI2]=0.;
res->val[JK_SIGTH] = rec->pz * v;
res->val[JK_PROB]=v;
res->val[JK_CUTFLAG]=1;
res->val[JK_COVMIN]=a_sum/((float)ev->nhits);
res->val[JK_COVMAX]=maxamp;
```

Cascade (`-i k`).
COG based cascade first guess. As of Mon Jun 10 18:03:40 CEST 2002, repository version 1.44, this does not fill more than COG into the result.

3.1.3.3 Various other guesses

All three first guesses (`-i a`).
Guesses (`i, m, q`=tensor of inertia, line fit, plane wave) are performed (in this order).

Reco track and energy (`-i e`).
Guesses reco track and energy using the following: This is empirical guess for energy; log10(en) = p0+p1*y^p2, where en is in GeV and y=nhits/llen*sumadc/nch. p0...p2 are magic numbers. Leaves `JK_*` alone.

Vertical down through COG of hits (`-i v`).
Just a vertical down going track.

```
res->val[JK_CUTFLAG]=0;
```

`res->val[JK_RCHI2]`
Reduced Cherenkov chi square

Random vertical (`-i w`).
The zenith angle is calculated from a random exponential distribution of cos(theta), while the azimuth is isotropic.

```
res->val[JK_CUTFLAG]=0;
```

`res->val[JK_RCHI2]`
Reduced Cherenkov chi square

Random isotrop (`-i r`).
A totally random track trough the canter of gravity of the hits.

```
res->val[JK_CUTFLAG]=0;
```

`res->val[JK_RCHI2]`
Reduced Cherenkov chi square

Generated track (`-i g`).
Use the leading muon of the track which generates this event as first guess. Of course, this works well only for Monte Carlo tracks. Leaves `JK_*` alone.

Reconstructed track (`-i f`).
Use one of the previous reconstructed tracks as first guess. The default is the first one found in the file. This may be changed with the option `-p f=<fit nr>`. This allows you to copy tracks in your f2k. Leaves `JK_*` alone.

Most of the first guess methods allow to specify an amplitude weight for the calculation. By default, this is set to zero (no amplitude weight). To change it, use the `-p w=<weight>` with a real number as weight.

3.1.4 Minimisation functions (-z)

You need to specify the likelihood function to use for the minimisation. This parameter is required whenever you want to have a minimisation (t.m. if the reconstruction doesn't stop after the first guess). The function is given by the `-z function` option. To specify more then one function you may either use subsequent `-z` options, or within one option concatenated by a `"+"` sign. A non-default weight can be specified in the form `"function:weight"`, or with `"weight*function"`. Currently implemented functions can be grouped into track fitting ()

3.1.4.1 Track fitting methods

`a_upandel`
`a_upandel_mpe`
`a_upandel_psa`
Ultimate Pandel (track) for 1pe, mpe, psa) (NEW)

`a_phpnh_table`
Hit/no hit using photon tables. Use `-X t=filename`. Exactly the same as a_phpnh except that the phit/nohit probabilities are read directly from the PTD tables, rather than from the pandel parametrisation. Minimise with simplex (`-m x`) with the following step sizes: `-x x:step=40,y:step=40,z:step=40,zenith:step=0.1,azimuth:step=0.2` (DS)

`a_timing_table`
Timing reconstruction using the photon tables. Use `-X t=filename`. This function is a product of single-hit likelihoods just the same as the "standard" a_upandl single-pe muon track likelihood function, except that the probabilities are read directly from the tables, thus eliminating the need to refer to a pandel parametrization. Minimize with simplex (`-m x`) with the following stepsizes: `-x x:step=40,y:step=40,z:step=40,zenith:step=0.1,azimuth:step=0.2` (DS)

`a_zwght` Amanda zenith weight for Bayesian reconstruction. Ad-hoc shape of prior.

`a_izwght` Same as `a_zwght`, but up and down are flipped.

`a_zwght2` Amanda zenith weight for Bayesian reconstruction. Prior modeled after the angular spectrum of muons predicted by Monte Carlo. Which primaries? Which tracking? Which ice?

`a_zwght3` As `a_zwght2`, but depth dependent muon flux is turned into angular spectrum. Use like this: `-p Z=0.:1. -z a_upandel+a_zenith_range+a_zwght3`.

`a_zwght3_invert`
Same as `a_zwght3` but up and down are flipped.

`a_fpt_phpnh`
New Pandel (track phit/nohit) function.

`a_pt_pnh` Pandel (track pnohit) funtion

`a_pt_ph` Pandel (track phit) function

`a_pt_amp` Ultimate Pandel (phit+pnohit+adc+tot) (NOT YET???)

`a_pt_tot` Ultimate Pandel tot info only (NOT YET???)

`a_pt_adc` Ultimate Pandel adc info only (NOT YET???)

3.1.4.2 Point Source (a.k.a. Shower) Methods

According to M. Kowalski, only `a_pp_upandel` and `a_upandel_ps_mpe` are used at the moment.

`a_pp_upandel`
: Ultimate Pandel for point sources. Standard for 1 pe cascades.

`a_upandel_ps_mpe`
: Ultimate Pandel (point source, mpe). Standard for mpe cascades.

`a_pp_phpnh_f`
: Cascade phit pnohit function, isotropic and fast

`a_pp_phpnh_ig`
: Ignacio's parametrization for phpnh-cascades. Non-isotropic version of `a_pp_phpnh_f`.

`a_pp_poisson`
: Point source npe-Poisson like function. Unclear.

`a_pp_psa` ???

`a_pp_upandel_psa`
: Poisson saturated Pandel function. Difference w.r.t `a_pp_poisson`?

3.1.4.3 Baikal Methods

`b_phit`
`b_pnohit`
`b_amp` Baikal P(hit) of hit channels, P(nohit) of channels not hit and P(amplitude) of hit channels for the Baikal detector. This is based on a table of amplitude distributions for muon energies between 33GeV and 150TeV. This will be described in the PhD thesis of Ole Streicher. Note that when you use these functions the first time, `recoos` needs some time (4 minutes on my K6-200) to calculate the whole table. This table (size approx. 1.5MByte) ist then stored in `/tmp/reco_lik_table.dat` to speedup the next start of `recoos`.

`b_camp` Baikal amplitude including correlations.

3.1.4.4 Personal Methods

`chi2_planewave`
: R. Porrata's plane wave-idea

`a_leu_1` M. Leuthold's first likelihood (environmentally correct) (whatever this means).

`a_pnhit` M. Leuthold's no hit prob function

`a_pamp` M. Leuthold's no amp prob function

`a_pedamp` P. Miocinovic's amplitude probability

`a_ffun` A. Bouchta's Amanda-Function (experimental)

3.1.4.5 Various Methods

`chi2_time`
: Normal Cerenkov (track) chi2. It just assumes gaussian errors for arrival times.

`chi2_scatt`
: Scatter Cherenkov (track) chi2 (amanda)

`a_zenith_range`
: Amanda restricted zenith range ???

`a_spaseadc`
: SPASE based adc function

`m_costh` Assume an exponential muon angle spectrum for reconstruction.

`m_ener` Assume a power law for energy reconstruction. This adds the probability to get a certain energy to the likelihood to get rid of the fact that a high energy muon is much less probable than a low energy. The default spectral index is 2.7, and can be changed with the `-p e=<index>` option.

`user` A user defined function, defined in `reco_user.c`.

3.1.4.6 Obsolete and Old Methods

`a_upandel_zw`
: Ultimate Pandel zenith weighted (OBSOLETE)

`a_ppandel`
: Old Patched Pandel time-likelihood (OLD default amanda)

`a_phit` Old Amanda P-hit (track).

`a_pnohit` Old Amanda P-nohit (track)

`a_phitpnohit`
: Old Amanda P-hit + P-nohit (track)

`a_pp_phpnh`
: (Then) new Pandel (points phit/nohit) function (M. Kowalski's first). Replaced by `a_pp_phpnh_f`.

`a_pp_phpnh_old`
: Old Pandel (points phit/nohit) function. Online help says New Pandel (points phit/nohit) function.

The following examples show how to specify `chi2_time` with the weight of 0.5 and `b_pnohit` with the default weight (1). Note the quotes (") to avoid interpretation of the * sign in the shell.

```
recoos -z chi2_time:0.5 -z b_pnohit
recoos -z "0.5*chi2_time" -z b_pnohit
recoos -z "0.5*chi2_time+b_pnohit"
recoos -z chi2_time:0.5+b_pnohit:1
```

3.1.5 Parametrisations (`-l`)

The option `-l` allows you to choose a certain parametrization method. It specifies which parameters are used by the minimization procedure. The following parametrizations are implemented:

`xyzza` Vertex (x, y, z), zenith angle, azimuth angle.

`xyazt` x and y of the track for z=0, azimuth, zenith and time relative to a coordinate system with the first guess as z axis.

`energy` Energy of the track.

`logE` Log10(energy) of the track.

`length` Track length.

`xyzt` Position: (x, y, z) and time. Used with cascade fit.

`za` Direction: zenith, azimuth.

You may choose several parametrizations by using subsequent `-l` options, or by concatenating them with a `","` as separator: `-l xyzza,logE`. Be carefully that you dont choose the same parameter twice (like `-l xyzza,xyazt` or `-l energy,logE`) since this may give unpredicatble results.

Normally, all defined parameters are free. You may define which parameters are fixed and which are free with the `-x` option with a comma separated list as parameters. For example, `-l xyzza,logE -x x,y,z,logE` can be used to make an energy reconstruction with the track direction (zenith, azimuth) fixed.

Recoos now also supports an enhanced syntax for setting reconstruction free parameters: the `-x` switch arguments now use the canonical form:

```
-x par [: [ min | max | step ] = # ]
```

where `par` is one of the defined free parameters, and the optional trailing `min`, `max`, or `step` specify lower and upper bounds on the parameter, or parameter stepsizes, respectively. Thus, for example, if you wanted to `recoos` with `x` bounded by `+/-` 100 m and have an initial step of 25 m, you'd say:

```
-x x:min=-100:max=100:step=25
```

3.1.6 Extended reconstruction options (-X)

The extended options will allow the user to specify the offset of the fitted track, specify some ice optics parameters and the like.

`-X g=<par>`

Specify centring of time residuals for first guess.

`<par>=c` Mean of all time residuals (default)

`<par>=m` Median of all time time residuals

`<par>=p` Most probable of all time residuals

`<par>=f` First of all time residuals

`<par>=n` No time correction

This option is disabled if the first guess is the previous reco track or the generated track.

`-X o=<par>`

Which optics to be used for Pandel reconstructions

`<par>=0` no hole ice irregularities

`<par>=1` hole ice lambda=100cm

`<par>=2` hole ice lambda=50cm (default)

`<par>=3` hole ice lambda=30cm

`<par>=4` hole ice lambda=10cm

`-X t=<filename>`
Specifies the name of a photon table (PTD style) for use with `-z a_timing_table`. **Filename must be shorter than 30 characters.**

`-X s=<par>`
Transform the track origin closest to a point ...

`<par>=o` ... closest to origing (default),

`<par>=c` ... closest to COG of hits,

`<par>=z` ... where time=0.

`<par>=n` Do not transform.

3.1.7 Reconstruction parameters (-p)

This is the section where you will find everything which was not covered before. Topics include triggering, more optics, noise of PMTs and so on.

3.1.7.1 Trigger

`-p t=min_string:min_channel`
sets a new trigger (**after cleaning**), default 1:5

3.1.7.2 First Guesses

`-p f=<par>`
Take fit number `<par>` as first guess, default last fit. **Only effective for `-if`.**

`-p g=<par>`
Take track number `<par>` as first guess, default `RDMC_NA`. **Only effective for `-ie`.**

3.1.7.3 PMT properties

`-p n=rate:window`
Assume a certain noise rate (Hz) within a time window (ns). Adds -2ln(rate*window) to the likelihood function for each OM.

`-p j=jitter`
Time jitter of PMT (ns). 5 ns in Baikal, 8 ns in AMANDA.

3.1.7.4 Energy dependencies

`-p S=<file>`
Set the file name for the energy dependent amplitudes (Baikal) (default baikal_amp_file.dat.bz2)

`-p A=<file>`
Set the file name for the energy dependent amplitudes (Amanda) (default amanda_amp_file.dat.gz)

`-p T=<file>`
Set the file name for the energy dependent discriminator thresholds (Amanda) (default amanda_th_file.dat.gz)

3.1.7.5 Iterative fit

`-p M=<num>`
: Max number of fits in case of Iterative or Fake search (`-rg -rf`) sets the number of fits which are attempted (default 25)

`-p L=<val>`
: Cut Likelihood/chi2 in case of Iterative or Fake search (`-rg -rf`) search is stopped if a chi2 or likelihoood better this value is found (default 1.5)

3.1.7.6 Particle Id and Spectral Index

`-p p=particle-id`
: The Fit recceives this partice/track id (default mu+). See "phonebook" for list of IDs.

`-p e=index`
: set the spectral for energy reconstruction (only for `-z m_ener`) (default 2.7)

3.1.7.7 Zenith weighting

`-p Z=min_cos(zenith):max_cos(zenith)`
: restricts the allowed zenith range between min and max of the cos(zenith) angle (Presently only used by zenith dependent Likelihood-Penalty functions) (default: -1:1)

`-p B=<val>`
: pass a bias factor for zenith weighting (experts only) (default: 1)

3.1.7.8 Other

Here's everything which did not fit in the above.

`-p w=<p1>` (default: 0) power of the amplitude weight in `-im` and `-iq`

`-p r=<val>`
: (default: 0) Radius of cylinder around the track to use as active volume. 0 = no limit, for negative values radius determined from hit patern

`-p m=<i>` (default 0) change the minuit strategy (SET STR) (only for `-m m`) i.e. how many times the likelihood function is called 0: bad,fast 1: better,slower 2: best,slowest

`-p F=<fit tol>`
: (default: 0.0001) Fit precision

`-p H=<val>`
: change the cos(zenith) of the horizon (default 0)

`-p C=<high>[:<low>[:<radius>[:<buffer>]]]`
: (default: 230:-250:65:70) definition of cylinder defining Amanda and additional size of buffer needed for Ped's energy reco

`-p P=<file>`
: (default `amanda_pe_file.dat.gz`) Set the file name for the <pe> table photonics? driver file (Amanda)

`-p E=<ene>`
: (default: last fit energy) force a first guess energy (such as 1E6) in MeV

`-p s=[1|2]`
: (default 1) Baikal summator mode: 1: one OM on summator 2: both OMs on summator

`-p N=[0|1]`
: (default: 0) use true NPE values provided by Amasim (for testing)

3.2 Internal structure of recoos

This section given an introduction into the internal structures of recoos. You may ignore this as long as you dont want to extend recoos by adding new code.

Recoos was developed to provide a maximal flexibility to the developer. This was done in separating all important tasks into own structures and functions which are quite independent. They are described in the next subsections.

The reconstruction program is built on top of rdmc and heavily uses the feautures of this library. So, in the following, we assume that you are already familar with rdmc. If not, it is strongly recommended to read the RDMC manual.

3.2.1 Where is what in recoos

`main.c`
: The mother of the program. Contains just the frame for everything. Doesn't know anything about physics.

`reco_init.c`
`reco_init.h`
: Initialize something in the beginning of the reconstruction.

`reco_par.c`
`reco_par.h`
`reco_opt.c`
`reco_opt.h`
`reco_control.c`
`reco_control.h`
: General structures for the use in the reconstruction (see below).

`reco_fg.c`
`inertion.c`
`reco_fg.h`
: Several first guess methods (`-i` option).

`tcorr.c`
`tcorr.h`
: Functions to calculate the time correction (`-X g` option).

`reco.c`
`reco.h`
: Minimization routines (`-m` option).

`powell.c`
`sa_reconstruction.c`
`sa_reconstruction.h`
`minr.h`
: General minimization routines, called in `reco.c`

`reco_hl.c`
`reco_hl.h`
: High level reconstruction routines (`-r option`)

`parametrize.c`
`parametrize.h`
: Parametrizations for the minimization routines (`-l` option)

`loglikfuncs.c`
`loglikfuncs.h`
`reco_amanda.c`
`reco_baikal.c`
`reco_cherenkov.c`
: Likelihood functions (`-z` option)

`reco_user.c`
: User defined likelihood function (enabled with `-z user`).

`adam_table.c`
`adam_table.h`
`pandel_amanda.c`
`pandel_amanda.h`
: Physics routines used for AMANDA reconstruction

`baikal.c`
`baikal.h`
`physics_baikal.c`
`reco_amp_table.c`
`reco_amp_table.h`
: Physics routines used by Baikal reconstruction

`physics.h`
`misc.c`
`misc.h`
`definitions.h`
: Several stuff that doesnt fit elsewhere

3.2.2 Structures used in recoos

The `rdmc` library provides the data structures `mtrack` for the reconstruction track parameters and `jk` for the reconstruction results. These data structures are dynamically allocated for each reconstruction. The representation in the data format corresponds to one `F`-line for each fit, in which the data of both structures are written. These values are filled into a N-tuple by the munt See Chapter 4 [munt], page 24 program, cuts on these parameters can be applied by the filter program muff See Chapter 5 [muff], page 31. The structures are defined in `rdmc.h`:

```
typedef struct {
  int id;                                  /* particle id */
  float e;                               /* kinetic energy */
  float x,y,z;         /* parameter of a track point [m] */
  float t;                  /* time [nsec] of  point xyz */
  float costh;        /* theta cosinus of the muon track */
  float phi;                        /* phi value in radians */
  float px, py, pz;                  /* direction cosinuus */
  float length;                     /* length of the track */
  int nmuon;      /* number of muons producing this track */
} mtrack;
typedef struct {
  int   id;
  float chi2, rchi2;
  float prob;
  float sigth;
  float covmin,covmax;
  int cutflag;
```

```
} jk;
```

It has to be noted that the naming of the `jk` structure elements is historically and should not be regarded as a definition but rather as 6 float and 2 integer placeholders where a reconstruction program may store information (without the need to create rdmc-like `user`-structures. In the following we give the specific definitions concerning the `recoos` program.

The values stored are different for Minuit (`-mm`) and powells (`-mp`) reconstruction (Powells stores only `jk.rchi2` and `jk.cutflag`). If no reconstruction, but only a first guess was caried out (`-rn`), the values stored are different for each used algorithm.

`mtrack.id`
: Particle id, recoos stores 5 (MUON_PLUS).

`mtrack.e` 0.0, since no energy reconstruction is implemented yet,

`mtrack.x y z t`
: Coordinates and time of a point on the track.

`mtrack.costh phi`
: Direction where the track comes from

`mtrack.px py pz`
: Direction cosine of the track

`mtrack.length nmuon`
: stores `length=BIG`=1e6 m (defined in `rdmc.h` and `nmuon=1`

`jk.id` An identifier of the reconstructon program/algorithm should be stored here. Its value is presently undefined. `recoos` stores: 3000 + 100*min_method + 10*abs(reco_method) + first_method, which is a number between 3000 and 4000. The exact definition is likely to be changed in the future.

`jk.chi2` This value is not stored in the current data-format (!), therefore it is not used by most `recoos` options and filled with the return values of the reconstruction functions, which is often the chi2but also often mostly the same value as `jk.rchi2`

`jk.rchi2` This is the reduced chi2 or -log(Likelihood) calculated for the track parameters. Usually it contains the reduced value of the minimisation function at the minimum (e.g. likelihood or line-fit) calculated for the specific model. In case no model was applied (e.g. vertical track) the reduced chi2 is calculated for a pure Cherenkov model. For muon reconstruction the reduced values are 0. if less than 5 channels are hit or the value at the minimum (e.g. chi2 or log(likelihood)) divided by (Nhits -4). In case of plane wave or line fit models it is divided by (Nhits -2). The first guess calculating the tensor of inertion returns the length of the main axis divided by the sum of the other two axis.

`jk.prob` The line-fit and plane-wave and time-flow first guesses fill the calculated speed parameter of the wave. The first guess calculating the tensor of inertia returns the length of the main axis, L1. The grid search stores the best likelihood found above the horizon.

`jk.sigth` The line-fit, plane-wave and time-flow first guesses fill the calculated speed of the wave into z-direction. The first guess calculating the tensor of inertion returns the direction of time flow projected on the main axis.

`jk.covmin`
: All amplitude weighted first guesses fill the average amplitude weight in the event. The tensor of inertia fit will store the intermediate eigenvalue, L2.

`jk.covmax`
: All amplitude weighted first guesses fill the largest amplitude weight in the event. The tensor of inertia fit will store the largest eigenvalue, L3.

`jk.cutflag`
: This is usually a negative number in case of errrors, 0 if no real reconstruction was done, but a fixed value was filled. For powels minimisation the number of iterations is given (-1 in case of too many steps (e.g.2000)). In case of ..minuit..

3.2.3 First guess functions

They provide a first guess which will be used by the minimization, but can also used independently.

As an example we look here into the line fit which is some kind of standard. It is defined in `reco.h` as

```
double reco_astg(const array *ar,  mevt *ev,
        mtrack *rec, mevt_special_t  *resu);
```

All it does is to calculate the track parameters from the hit times (in `*ev`) with the use of the OM positions (they are in `*ar`) and fill this into the track `rec`. The code for `reco_astg()` can be found in `reco_fg.c`.

To write your own first guess:

1. Write a function similar to `reco_astg()`, put it into `reco_fg.c`
2. Include the header into `reco.h`
3. In the beginning of reco_fg.c, you find the table `defined_reco_methods[]`. Add an entry for your function before the line `{ NULL, NULL, NULL}`. The fields to fill are

   `char *id` A short name, used as command line flag (`-i` option)

   `char *description`
   : A one line description of the method

   `reco_fg_fun_t *func`
   : Your function name

   For `reco_stg()`, this line is

   ```
   {"m", "Line Fit [Stenger] (Amp weighted)", reco_astg},
   ```

3.2.4 Time correction functions

The first guess usually doesn't provide a good time information for the track. To fit a time which is somehow usable for the minimization, this function is called after the first guess. Standard here is to set the time that the time chi2 is minimal. This is done in the function

```
double tmean(const array *ar, const mtrack *tr, const mevt *ev);
```

which resides in `tcorr.c` (prototypes in `tcorr.h`). To add your own function, you should write a similar one (which returns the time correction), put it into `tcorr.c` and add it to the table at the top of this file. The entries here are

`char *id` A unique id, for the command line of recoos (`-X g` option)

`char *description`
: A one-line description

`tcr_fun_t *func`
: Your function name

F.e. `{"c", "mean of all time-residuals", tmean},` for the time correction with mean values.

3.2.5 Likelihood functions

The likelihood functions are the "soul" of the reconstruction. They contain actually all the physics. You may add several of them at the command line. The minimization tries to find a track that these functions get a minimal value. To write your own, you may firstly use the "user defined function". Just look into `user.c` and fill the empty space by physics. If you are lucky with your function and it should stay permanently, you may write your own. The prototype is

```
double chi2(const array *ar, const mtrack *tr,  mevt *ev);
```

Add it to `reco_amanda.c` or `reco_baikal.c`, add the prototype to `loglikfuncs.c` and add them to the table `defined_reco_functions` just at the beginning of `loglikfuncs.c`. The fields to fill are

`char *id` A short name, for calling them via `-z` option

`char *description`
: A one line description

`reco_fun_t *func`
: The function name

`double weight`
: The default weight

An example is

```
{"chi2_time", "Normal Cherenkov (track) chi2", chi2 , 1.0},
```

for the normal chisquare function.

3.2.6 Minimization procedures

These functions define the algorithm used for minimization (like Powells, Minuit, or Simulated Annealing). Because these are normally larger functions (and they may be just copied from other sources), they usuallu reside in their own files. The file `reco.c` contains just a stub to call them with the right parameters and initializations, as well as a table of all defined minimization procedures. The function called by recoos has the prototype

```
double myminimizer(const array *geom, mevt *event,
                   mtrack *rec, mevt_special_t *res, minuit_t *fun);
```

It should call the parametrization and deparametrization functions as well as doing the correct initialization. The values needed for that come from the static structure `reco_control`. For details, see the source code - it is still not in a really smart style (But, on the other hand: Adding new minimization procedures is not an every-day job).

The minimization function should minimize the function `fun()`, which has a Minuit like type:

```
void fun(int *npar, double *grad, double *fval,
        double *xval, int *iflag, int (*futil)());
```

The parameter fields `grad`, `iflag` and `futil()` are just dummies and should not be used. `npar` must contain the number of parameters, `xval` the vector of parameters (fixed and free). `*fval` returns the function values to be minimized.

After a successfull call, the minimizer should fill the reconstruction results to `res`. The return value of the minimizer procedure is the minimized function value, or -1 in case of an error.

However, after suceeding, add your function to the table `defined_minimizers[]` which has the following fields:

`char *id` A short name, for calling them via `-m` option

`char *description`
: A one line description

`reco_method_fun_t *func`
: The function name

Example line for the Powell's minimizer is

```
{"p", "Powell's", reco_powell},
```

3.2.7 Parametrization functions

These functions handle the conversion from physical track parameters (in `mtrack`) to a vector of parameters and back. The minimization procedures (powells, minuit etc) will change only this vector to minimize. The "parametrization function" (which fills the vector from the track) is called once, to fill the first guess into the parameter vector.

The "deparametrization function" fills the values back to the track. It is used during the minimization procedure (before calling the likelihood functions, since they work with tracks), and after the minimization to get the resulting track.

If you write your own function, be sure that the two functions are really complementary. If you need additional storage place for temporary values, you may use global variables for that (although it is a bad idea here - it will give problems when we extend recoos for parallel processing).

The functions have the prototypes

```
static void mypamrametrize(mtrack *tr, double *xval);
static void mydeparametrize(mtrack *tr, double *xval);
```

The natural place to store them is the file `parametrize.c`. For your parametrization, you should also fill a structure `reco_param_t` and add the name of this structure to the table `defined_parametrizations[]` in the beginning of the file. The entries in your `reco_param_t` are:

`char *id` A short name, for calling them via `-x` option

`char *description`
: A one line description

`int npars` Total number of parameters for this parametrization

`para_settings_t par[MAX_RECO_NPARS]`
: Default parameter settings (see below). Should be filled for every used parameter.

`reco_para_fun_t *fromtrack`
: The function name for the function which fills the values from the track into the parameter vector

`reco_depara_fun_t *totrack`
: The function name for the function which fills the values from the parameter vector into the track

For the parameter settings (`par` field), there exists another structure. Note that not all values are used by each minimization procedure.

`char *name`
: The name of this parameter

`int free` This is 1 if the parameter is free and 0 if it is fixed by default.

`double step`
: The initial step value.

`double lbound`
`double ubound`
: lower and upper boundaries

You may change which parameters are fixed in run time with the `-x` option which takes the names of all free parameters.

An example for the track length parametrization is

```
static void para_length(const mtrack *tr, double *xval);
static void depara_length(mtrack *tr, const double *xval);
reco_param_t parametrization_xyzzae = {
  "length", /* id */
  "Track length", /* description */
  1,        /* number of parameters */
  { {"length",       1, 1.0, 0.0, 0.0} }, /* parameter list */
  para_length,     /* parametrization function */
  depara_length,   /* deparametrization function */
};

static void para_length(const mtrack *tr, double *xval)
{
  xval[0] = tr->length;
}

static void depara_length(mtrack *tr, const double *xval)
{
  tr->length = xval[0];
}
```

3.2.8 Changing default values

4 munt

4.1 Usage

```
Usage: munt [arguments] file
   Book a CWN N-tuple (id 100) with reco cuts
 arguments:
  -v        Verbose mode
  -h        This help
  -H        Long help
  -X        Book the reco results (see long help)
  -Y        Book the reco observables (see long help)
  -Z        Book the ndirect block (see long help)
  -S        Book the smoothness block (see long help)
  -T        Book the hit-coincidence (see long help)
  -U        Book the USER-BLOCK (see long help)
  -V        Book the generator block (see long help)
  -W        Book the topological block (see long help)
  -I        Book the UCI variables (temporary only)
  -C        Book astronom. coordinates (see long help)
  -B        Book the Baikal variables (temporary only)
  -J        Book the OM information block
  -K        Book the hit information block
              Calling -K multiple times (i.e. -K -K)
              is needed to get all variables
  -o name   Output file
  -O name   HBOOK output file (default: munt.hbook)
  -f        Force creation of the hbook file
  -F        Force output of events (default=no)
  -n        Process event even with no hits
  -L        Allow large Hbook files (e.g.Recl=4096)
  -p <what> Init the physics parameter for <what>
    what=b : Baikal
    what=a : Amanda (default)
  -y opt    Hit reduction/ event cleaning  options
  file      Input file (default: stdin)
```

4.2 The Munt n-tuple

An N-Tuple with id 100 is booked, which may contain the following blocks:

4.2.1 Global info - MNT_HEAD

IRUN	Run number
IEVENT	Event number
NCH	Number of hit channels
NSTR	Number hit strings
NHITS	Number of hits

4.2.2 Generated tracks - MNT_GEN

Booked with the -V option.

NTRACK	Number of generated tracks
TRUEZEN	True zenith angle
TRUEAZI	True azimuth angle
TRUEDIST	True (closest) distance to detector (0,0,0)
TRUEX	True x of closest to detector (0,0,0)
TRUEY	True y of closest to detector (0,0,0)
TRUEZ	True z of closest to detector (0,0,0)
TRUEEN	True energy

4.2.3 Reco observables for generated tracks - MNT_OGEN

Booked with the -V -Y options.

TPHA	Average Hit probability (per hit channel)
TPNHA	Average NO-Hit probability (per NOT hit channels)
TPHNHA	Average Hit + NO-Hit probability (per all channels)
TPHE	Hit probability (hit channels) (event)
TPNHE	NO-Hit probability (NOT hit channels) (event)
TCHI2CER	Chi^2/nhits value according to cherenkov model
TCHI2SCA	Chi^2/nhits value according to scattering model

4.2.4 Ndirect for generated tracks - MNT_OGEN

Booked with the -V -Z options.

TGDIR	Summed Ndirect Gaussian value
TRUEEARLY	Number of hits to early (residual <= -5 ns)
TRUEDIRA	Number of direct hits (-5 < residual +20 ns)
TRUEDIRB	Number of direct hits (-5 < residual +35 ns)
TRUEDIRC	Number of direct hits (-5 < residual +75 ns)
TRUELATE	Number of late hits (75< residual < 10 mu sec)
TRUEDSTRA	Number of strings with direct hits (-5 < residual +20 ns)
TRUEDSTRB	Number of strings with direct hits (-5 < residual +35 ns)
TDCOGX	X-Centre of Gravity of direct hits (-5< residual <25ns)
TDCOGY	Y-Centre of Gravity of direct hits (-5< residual <25ns)
TDIRCOGZ	Z-Centre of Gravity of direct hits (-5< residual <25ns)

`TDIRSDIST`
: Smallest perp distance of direct hits (-5< residual <25ns)

`TDIRMDIST`
: Smallest perp distance of direct hits (-5< residual <25ns)

`TDLDIST` Highest perp distance of direct hits (-5< residual <25ns)

`TDRDIST` Smallest perp distance of late hits (75ns< residual)

`TDZLEN` Average perp distance of direct hits (-5< residual <25ns)

4.2.5 Reconstruction results - `MNT_REC`

Booked with the `-X` option.

`NFIT` Number of reconstructions

`ZENITH(NFIT)`
: Reco Zenith

`AZIMUTH(NFIT)`
: Reco Azimuth

`DIST(NFIT)`
: Reco (closest) distance to detector (0,0,0)

`DISX(NFIT)`
: Reco x of closest distance to (0,0,0)

`DISY(NFIT)`
: Reco y of closest distance to (0,0,0)

`DISZ(NFIT)`
: Reco z of closest distance to (0,0,0)

`ENERGY(NFIT)`
: Reco energy

`DELANG(NFIT)`
: Reco pointing error (solid angle)

`DELVERT(NFIT)`
: Reco distance error to (0,0,0)

`FITID(NFIT)`
: Identifier for Reco program (see rdmc->jk)

`JKFLAG(NFIT)`
: Fit result status of Reco-program (see rdmc->jk and recoos)

`JKRCHI(NFIT)`
: Fit result minimum value (see rdmc->jk and recoos)

`JKPROB(NFIT)`
: Fit result parameter Reco-program (see rdmc->jk and recoos)

`JKSIGTH(NFIT)`
: Fit result error estimate Reco-program (see rdmc->jk and recoos)

`JKCMIN(NFIT)`
: Fit result error estimate Reco-program (see rdmc->jk and recoos)

`JKCMAX(NFIT)`
: Fit result error estimate Reco-program (see rdmc->jk and recoos)

4.2.6 Reconstruction observables MNT_OREC

Booked with the -X -Y options.

NOFIT Number of reconstructions

CAUSAN(NOFIT)
Most negative time residual

CAUSAP(NOFIT)
most postive time residual

MINDRES(NOFIT)
Most negative: dr * cos(track, dr)/c -dt (whole detector)

MAXDRES(NOFIT)
Most positive: dr * cos(track, dr)/c -dt (whole detector)

CMINDRES(NOFIT)
Most negative: dr * cos(track, dr)/c -dt (neighbour OM)

CMAXDRES(NOFIT)
Most positive: dr * cos(track, dr)/c -dt (neighbour OM)

MAXD1PE(NOFIT)
Largest distance to 1PE

MAXD2PE(NOFIT)
Largest distance to 2PE

MAXD3PE(NOFIT)
Largest distance to 3PE

MAXD4PE(NOFIT)
Largest distance to >=4PE

MIND1PE(NOFIT)
Smallest distance to an OM with a hit

MIND0PE(NOFIT)
Smallest distance to an OM with No hit

PHA(NOFIT)
Average Hit probability (per hit channel)

PNHA(NOFIT)
Average NO-Hit probability (per NOT hit channels)

PHNHA(NOFIT)
Average Hit + NO-Hit probability (per all channels)

PHE(NOFIT)
Hit probability (hit channels) (event)

PNHE(NOFIT)
NO-Hit probability (NOT hit channels) (event)

CHI2CER(NOFIT)
Chi^2/nhits value according to cherenkov model

CHI2SCA(NOFIT)
Chi^2/nhits value according to scattering model

4.2.7 Reconstruction ndirect results - MNT_DREC

Booked with the -X -Z options.

NDFIT Number of reconstructions

GDIR(NDFIT)
Summed Ndirect Gaussian value

NEARLY(NDFIT)
Number of hits to early (residual <= -5 ns)

NDIRA(NDFIT)
Number of direct hits (-5< residual <15ns)

NDIRB(NDFIT)
Number of direct hits (-5< residual <25ns)

NDIRC(NDFIT)
Number of direct hits (-5< residual <75ns)

NLATE(NDFIT)
Number of late hits (residual > 75 ns sec)

NSTRDA(NDFIT)
Number of strings with direct hits (-5< residual <15ns)

NSTRDB(NDFIT)
Number of strings with direct hits (-5< residual <25ns)

DIRCOGX(NDFIT)
X-Centre of Gravity of direct hits (-5< residual <25ns)

DIRCOGY(NDFIT)
Y-Centre of Gravity of direct hits (-5< residual <25ns)

DIRCOGZ(NDFIT)
Z-Centre of Gravity of direct hits (-5< residual <25ns)

DIRSDIST(NDFIT)
Smallest perp distance of direct hits (-5< residual <25ns)

DIRMDIST(NDFIT)
Largest perp distance of direct hits (-5< residual <25ns)

DIRLDIST(NDFIT)
Smallest perp distance of late hits (75ns< residual)

DIRRDIST(NDFIT)
Average perp distance of direct hits (-5< residual <25ns)

DIRZLENG(NDFIT)
Length of Hits projected on the track (-5< residual <25ns)

4.2.8 Astronomical coordinates - MNT_COOR

Booked with the -C option.

NCOORFIT Number of reconstructions

RA(NCOORFIT)
Equatorial coordinates right ascension

DEC(NCOORFIT)
Equatorial coordinates declenation

`LL(NCOORFIT)`
: Galactic longitude

`BB(NCOORFIT)`
: Galactic latitude

`ECLONG(NCOORFIT)`
: Ecliptic longitude

`ECLATT(NCOORFIT)`
: Ecliptic latitude

`SLL(NCOORFIT)`
: Super Galactic longitude

`SBB(NCOORFIT)`
: Super Galactic latitude

`SUNRA` RA of Sun

`SUNDEC` DEC of Sun

`MOONRA` RA of Moon

`MOONDEC` DEC of Moon

4.2.9 Topological info - `MNT_TOP`

Booked with the `-W` option.

`LOWZ` z of lowest hit pmt

`HIGHZ` z of highest hit pmt

`COGX` X-centre of gravity of hits

`COGY` Y-centre of gravity of hits

`COGZ` Z-centre of gravity of hits

`MINDT` Most negative time diff in a string (nils)

`MAXDT` Most negative time diff in a string (nils)

`MADTDR` Largest abs timedifference in detector

`MIDTDR` Smallest abs timedifference in detector

`CMADTDR` Largest abs timedifference for neighbours in a string

`CMIDTDR` Smallest abs timedifference for neighbours in a string

4.2.10 Hit-coincidence info - `MNT_HIT`

Booked with the `-U` option.

`ND` Number of channels in double coincidences

`NT` Number of channels in triple coincidences

`NQ` Number of channels in quad coincidences

`NDS` Number of channels in double skip coincidences

`NTS` Number of channels in triple skip coincidences

`NQS` Number of channels in quad skip coincidences

NSTRD	Number of strings with double coincidences
NSTRT	Number of strings with in triple coincidences
NSTRQ	Number of strings with in quad coincidences
NSTRDS	Number of strings with double skip coincidences
NSTRTS	Number of strings with triple skip coincidences
NSTRQS	Number of strings with quad skip coincidences

5 muff

```
muff: Muon/Neutrino fit filter program of Ole
muff: Version: 1.2
muff: Cuts:
Usage: muff [arguments] file
 arguments: -o file   Output file
            -v        Verbose mode
            -h        This help
            -b        Baikal like input file
            -B        Baikal like output file
             -a        ASCII input file (default)
            -A        ASCII output file (default)
            -H 13     channel 13 hitted
            -z -1.5   min pmt 'z' coordinate is < -1.5m (lowest pmt)
            -Z 12     max pmt 'z' coordinate is > 12m (highest pmt)
            -t        c * dt <= dr for all channel pares
            -d 2.5    Nils' cut min dt in a string is less then 2.5 ns/m
            -D 5.0    Nils' cut max dt in a string is less then 5.0 ns/m
            -T        Time interval criterion for all channel pairs
            -e 5.0    Max time residual is 5.0 nsec
            -E -5.0    Causality violation (no time residual < -5.0 nsec)
            -c 2.0    Max chi^2 is 2.0
            -r 0.1    Min distance from rec. track to each hitted pmt is 0.1 m
            -R 15     Max distance from rec. track to each hitted pmt is 15 m
            -p 0.6    Minimal nohit probability is 0.6
            -P 0.5    Minimal hit probability is 0.5
            -y 60     Minimal zenith angle of rec. track is 60 degrees
            -Y 90     Maximal zenith angle of rec. track is 90 degrees
            -Q 1      Min stored cut flag is 1
            -q        don't exclude cutted events; only set the cut flag
            -x        inverse filter: only events *not* passed
            file      Input file
With ASCII input/output muff can be used as filter.
If there is more than one cut, the event survives if
at least one of the cuts is good
```

6 Examples

Here's one that will give you five fits. Users of `(t)csh` please replace `HOP=TOP` with `set HOP = TOP` (mind the spaces!). Recklessly stolen from J. Lamoureux' and P. Ekstrom's `amaperl`.

```
#!/bin/bash

# fits tensor of inertia, line fit, plane wave,
# fits track likelihood
# fits shower

# some cleaning (don't take numbers to serious, it's just for the show
DEAD_OM="-y N=3 -y N=18 ..."
NOISY_OM="-y N=267 ..."

# now the first guesses:
TOI_LF_PW="-r n -i a -X g=f -p w=0. -X s=n -p t=0:10"
# explanation:
# -r n: first guess only
# -i a: does the TOI, LF, PW
# -X g=f: centres time residuals around the first of all time residuals
# -p w=0: amplitude weight is 1
# -X s=n: keep origin of track at where algorithm calculates it
# -p t=0:10: triggers on at least 0 strings, 10 oms

# the real thing for the track:
TRACK_FIT="-r m -i f -p f=2 -X g=n -m x -X s=o -z a_upandel\
 -x x:step=10,y:step=10,z:step=10,zenith:step=0.1,azimuth:step=0.25\
 -p t=0:0"
# explanation
# -r m: muon reco
# -i f: use reconstructed track as first guess. Previous by default but
# -p f=2: makes it track number 2 (the LF from TOI_LF_PW)
# -X g=n: see above
# -m x: minimise with SIMPLEX
# -X s=o: put track origin as close as possible to centre of detector
# -z a_upandel: Selects Pandel function for tracks
# -x ...: Starting values for simplex
# -p t=0:0: no triggering, all pass

# the cascade:
CASCADE_FIT="-r m -i f -p f=3 -X g=n -m x -X s=n -z a_pp_upandel\
 -l xyzt -x x:step=10,y:step=10,z:step=10,time:step=25\
 -p p=1003 -p t=0:0"
# explanation
# -r m: see above
# -i f -p f=3: use track 3 (PW) as first guess
# -X g=n -m x -X s=n: see above
# -z a_pp_upandel: selects 1 pe cascade pandel function
# -l xyzt: the result is given by the position and time of the cascade
# -p p=1003: particle id is 1003 (see phone book)
# -p t=0:0: see above
```

```
# now we're ready to go:
recoos $DEAD_OM $NOISY_OM $TOI_LF_PW\
| recoos $DEAD_OM $NOISY_OM $TRACK_FIT\
| recoos $DEAD_OM $NOISY_OM $CASCADE_FIT

# script ends.
# How did I know about the -X , -i and -p? Check recoos -hh |& less !
```

Table of Contents