Note: page is outdated

This page is an early documentation, I tried to document BBS clearer on the **BBS page**.

Introduction

This page describes the syntax of the BBS parset file. BBS consists of three components: `control`, `kernel`, and `solver`. Both `kernel` and `solver` need only a very small subset of the information that can be put in the parset (see examples below). The `control` parset file is usually much larger because it describes the processing that BBS has to do. For testing purposes, it is often useful to run all three executables on a single node. In that case it is possible to create a single combined parset, because each executable will ignore all keys it does not understand.

Below you will find a typical example parset file for each of the three executables. The rest of this page discusses all the valid keys in more detail.

Example control parset file

```
Observation = L2007_03463.gds    # Global measurement description (GDS) file

Strategy.ChunkSize = 100    # Chunk size (timeslots) [SEE DOCUMENTATION]
Strategy.Steps = [solve, subtract, correct]

BBDB.Key = run00    # (Unique) name identifying the calibration session
BBDB.Host = cepmaster0    # Hostname or IP-address of postgresql server
BBDB.Port = 5432    # Port number where the postgresql server is listening
BBDB.Name = john    # Name of the database to use
BBDB.User = postgres    # Username for accessing the postgresql server
BBDB.Password =     # Password for accessing the postgresql server

Step.solve.Operation = SOLVE    # Operation to perform
Step.solve.Model.Sources = []    # Sources to include in the model (all if
empty)
Step.solve.Model.DirectionalGain.Enable = T    # Include separate complex
gain terms for each source.
Step.solve.Model.Cache.Enable = T    # Enable caching of intermediate
results.
Step.solve.Solve.Parms = ["DirectionalGain:0:0:*", "DirectionalGain:1:1:*"]
# Parameters to fit
Step.solve.Solve.CellSize.Freq = 0    # Solution cell size (channels)
Step.solve.Solve.CellSize.Time = 1    # Solution cell size (timeslots)
Step.solve.Solve.CellChunkSize = 10    # Cell chunk size (timeslots)
Step.solve.Solve.Options.MaxIter = 10    # Maximal number of iterations
Step.solve.Solve.Options.EpsValue = 1e-9    # Convergence criterion
Step.solve.Solve.Options.EpsDerivative = 1e-9    # Convergence criterion
```

```
Step.solve.Solve.Options.ColFactor = 1e-9    # Colinearity factor
Step.solve.Solve.Options.LMFactor = 1.0    # Levenberg-Marquardt factor
Step.solve.Solve.Options.BalancedEqs = F    # Assume balanced equations?
Step.solve.Solve.Options.UseSVD = T    # Use singular value decomposition?

Step.subtract.Operation = SUBTRACT    # Operation to perform
Step.subtract.Model.Sources = []    # Sources to include in the model (phase
centre if empty)
Step.subtract.Model.DirectionalGain.Enable = T    # Include separate complex
gain terms for each source.

Step.correct.Operation = CORRECT    # Operation to perform
Step.correct.Model.Sources = ["CasA"]    # Sources to include in the model
(all if empty)
Step.correct.Model.DirectionalGain.Enable = T    # Include separate complex
gain terms for each source.
Step.correct.Output.Column = CORRECTED_DATA    # Output column (no output if
empty)
```

Example kernel parset file

```
ObservationPart.Filesystem = lioff021:/dev/sda10    # File system on which
the part of the observation to process is located.
ObservationPart.Path = /data/L2007_03463_SB0.MS    # Absolute path to the
part of the observation to process.

BBDB.Key = run00    # (Unique) name identifying the calibration session
BBDB.Host = cepmaster0    # Hostname or IP-address of postgresql server
BBDB.Port = 5432    # Port number where the postgresql server is listening
BBDB.Name = john    # Name of the database to use
BBDB.User = postgres    # Username for accessing the postgresql server
BBDB.Password =     # Password for accessing the postgresql server

ParmDB.Instrument = L2007_03463_SB0.instrument    # Instrument model
parameters
ParmDB.Sky = L2007_03463_SB0.sky    # Sky model parameters
```

Example solver parset file

```
PortRange = [6500, 6599]  # Port range to try on start-up
ConnectionBacklog = 25    # Maximal number of pending connections

BBDB.Key = run00    # (Unique) name identifying the calibration session
BBDB.Host = cepmaster0    # Hostname or IP-address of postgresql server
BBDB.Port = 5432    # Port number where the postgresql server is listening
BBDB.Name = john    # Name of the database to use
```

```
BBDB.User = postgres    # Username for accessing the postgresql server
BBDB.Password =     # Password for accessing the postgresql server
```

Global settings

BBDB [Relevant to `control`, `kernel`, `solver`]

> Information about the BlackBoard database. **NB. Do _not_ include when using the *calibrate* script**
>
> **Key** : *string* = default
> > Name that identifies the session.
>
> **Host** : *string* = localhost
>
> > Hostname or IP-address of the database server.
>
> **Port** : *integer* = <database server default>
>
> > Port number on which the database server is listening. For Postgres databases the default is 5432.
>
> **Name** : *string*
>
> > Name of the database.
>
> **User** : *string*
>
> > Username to access the database server.
>
> **Password** : *string* = <empty string>
>
> > Password to access the database server.

Observation : *string* [Relevant to `control`]

> Global measurement descrition (GDS) file that describes the parts that compose the observation to be processed. **NB. Do _not_ include when using the *calibrate* script**

ObservationPart [Relevant to `kernel`]

> Describes the part (MS) of the observation to be processed. **NB. Do _not_ include when using the *calibrate* script**
>
> **Filesystem** : *string*
> > File system on which the part of the observation to process is located. **NB. Must match the file**

> **system specified in the GDS-file exactly.**

Path : *string*

> Absolute path to the part of the observation to process. **NB. Must match the path specified in the GDS-file exactly.**

ParmDB [Relevant to `kernel`]

> Information about the parameter databases (e.g. instrument model parameters, sky model parameters). **NB. Do _not_ include when using the *calibrate* script**
>
> **Instrument** : *string*
> > Path to the instrument model parameter database.
>
> **Sky** : *string*
>
> > Path to the sky model parameter database.

PortRange : *list of integers* = [6500, 6599] [Relevant to `solver`]

> On start-up the specified range of ports will be searched for a free port on which to start listening. **NB. Do _not_ include when using the *calibrate* script**

ConnectionBacklog : *integer* = 10 [Relevant to `solver`]

> Maximal number of pending connections. This value may have to be increased when starting a large number of `kernel` processes.

Strategy

The strategy defines the operations that need to be performed on the data. It consists of one or more (multi-)steps.

InputColumn : *string* = DATA

> Name of the column in the observation part that contains the input data.

Baselines : *string* = *&

> Baselines to read. The CASA baseline selection syntax should be used here. If this key is not specified, all cross-correlations will be selected.

```
Strategy.Baselines = *&    # Select all cross-correlations (default).
Strategy.Baselines = CS*&&RS*;CS*&    # Select all cross- and auto-
correlations between core and remote stations,
```

```
                                        # and all cross-correlations between
core stations.
```

Correlations : *list of strings* = []

> Correlations to read.
>
> **NB. Specifying anything here will generate an exception because reading a subset of the available correlations is not supported yet.**

TimeRange : *list of strings* = []

> Time range to process, expressed as date/time string(s) (as returned by *msinfo*). All timeslots will be used if this field is left empty. Either a range or a start time can be provided.

```
Strategy.TimeRange = [27-Jul-2007/16:05:04]
Strategy.TimeRange = [27-Jul-2007/16:05:04, 28-Jul-2007/13:05:04]
```

ChunkSize : *integer*

> Chunk size in timeslots. A chunk represents an amount of input data that is loaded into memory and processed as a whole. This is useful when the amount of visibility data is too large to fit into main memory. A value of zero means all.

UseSolver : *bool* = F

> Will a global solver be used in this strategy?

Steps : *list of strings*

> The names of the steps that compose the strategy. It is an error to leave this field empty.

Step

A *single-step* describes one unit of work in the strategy. A step that is defined in terms of a number of other steps is known as a *multi-step*. The attributes of a multi-step should be interpreted as *default values* for the steps that compose the multi-step. These default values can always be overridden.

Steps : *list of strings*

> The names of the steps that compose this step (for multi-steps), or absent (for single steps). If specified, this key shall *not* be empty, or an exception will be thrown.

Baselines : *string* = *&

> Baselines to process. The CASA baseline selection syntax should be used here. If this key is not specified, all cross-correlations will be selected.

```
Step.subtract.Baselines = *&    # Select all cross-correlations (default).
Step.subtract.Baselines = CS*&&RS*;CS*&    # Select all cross- and auto-
correlations between core and remote stations,
                                           # and all cross-correlations
between core stations.
```

Correlations : *list of strings* = []

> Correlations to process. If this key is not specified, all correlations will be selected.

```
Step.simulate.Correlations = [XX,YY]
Step.simulate.Correlations = [RR,RL]
```

Model : *Model*

> Model configuration to use.

Operation : *string*

> The operation to be performed in this step. One of 'PREDICT', 'ADD', 'SUBTRACT', 'CORRECT', 'SOLVE'. Only relevant for single steps, should be absent for multi-steps.
> PREDICT - Simulate visibilities.
> ADD - Add simulated visibilities to the input visibilities.
> SUBTRACT - Subtract predicted visibilities from the input visibilities.
> CORRECT - Correct the input visibilities.
> SOLVE - Fit model parameters.

*Single steps should define one of the following fields, depending on the value of **Operation**:*

> **Correct** : *Correct*
>
> > Arguments of the CORRECT operation.
>
> **Solve** : *Solve*
>
> > Arguments of the SOLVE operation.

Output

> **Column** : *string* = <empty string>
>
> > Column in the observation part wherein the output values of this step should be written. If left empty, no data will be written.
>
> **WriteFlags** : *bool* = F
>
> > If set to true the flags in the observation will be updated.
>
> **WriteCovariance** : *bool* = F

> If set to true, covariance information will be written to the covariance column linked to the column specified at **Column**.

Example

```
Step.MultiStepExample.Steps = [solve, subtract]    # Steps that compose the
multi-step.
Step.MultiStepExample.Baselines = CS016*&CS008*,CS001*;CS001*&CS010*

Step.solve.Operation = SOLVE    # Operation to perform
Step.solve.Model.Sources = []    # Sources to include in the model (all if
empty)
Step.solve.Model.DirectionalGain.Enable = T    # Include separate complex
gain terms for each source.
Step.solve.Model.Cache.Enable = T    # Enable caching of intermediate
results.
Step.solve.Solve.Parms = ["DirectionalGain:0:0:*", "DirectionalGain:1:1:*"]
# Parameters to fit
Step.solve.Solve.CellSize.Freq = 0    # Solution cell size (channels)
Step.solve.Solve.CellSize.Time = 1    # Solution cell size (timeslots)
Step.solve.Solve.CellChunkSize = 10    # Cell chunk size (timeslots)
Step.solve.Solve.Options.MaxIter = 10    # Maximal number of iterations
Step.solve.Solve.Options.EpsValue = 1e-9    # Convergence criterion
Step.solve.Solve.Options.EpsDerivative = 1e-9    # Convergence criterion
Step.solve.Solve.Options.ColFactor = 1e-9    # Colinearity factor
Step.solve.Solve.Options.LMFactor = 1.0    # Levenberg-Marquardt factor
Step.solve.Solve.Options.BalancedEqs = F    # Assume balanced equations?
Step.solve.Solve.Options.UseSVD = T    # Use singular value decomposition?

Step.subtract.Operation = SUBTRACT    # Operation to perform
Step.subtract.Model.Sources = []    # Sources to include in the model (all
if empty)
Step.subtract.Model.DirectionalGain.Enable = T    # Include separate complex
gain terms for each source.

Step.correct.Operation = CORRECT    # Operation to perform
Step.correct.Model.Sources = ["CasA"]    # Sources to include in the model
(all if empty)
Step.correct.Model.DirectionalGain.Enable = T    # Include separate complex
gain terms for each source.
Step.correct.Output.Column = CORRECTED_DATA    # Output column (no output if
empty)
```

Model

Model configuration.

Sources : *list of strings* = []

> List of sources to include in the model. Shell style wildcards are allowed. An empty list will select all sources in the sky model parmdb.

Cache.Enable : *bool* = F

> If set to true, intermediate results will be cached. This is especially useful when solving, because it will prevent unnecessary recomputation at the start of each iteration.
>
> **NB. The current implementation does *not* limit the maximal amount of memory occupied by the cache. Therefore, caching is still considered *experimental*. It will work reliably in most cases, but it can cause the application to crash if the cache grows to the point where no more (virtual) memory can be allocated.**

Phasors.Enable : *bool* = F

> If set to true, complex parameters are expressed as (amplitude, phase) components instead of (real, imaginary) components. As a consequence the model is extended with a conversion for each complex parameter from (amplitude, phase) to (real, imaginary).

Bandpass.Enable : *bool* = F
NB: Bandpass correction is not supported anymore in the current implementation. A revised weighting is being worked on.

> Multiply the sum of the source coherences by a *real-valued diagonal matrix*. The naming convention for the associated parameters is “`Bandpass:0:0:<station name>`” and “`Bandpass:1:1:<station name>`”.

Gain.Enable : *bool* = F

> Multiply the sum of the source coherences by a *complex-valued Jones matrix*. The naming convention for the associated parameters is “`Gain:{0,1}:{0,1}:{Real,Imag}|{Ampl,Phase}:<station name>`”. Depending on the value of **Phasors.Enable** either `{Real,Imag}` or `{Ampl,Phase}` is selected.

DirectionalGain.Enable : *bool* = F

> Multiply the source coherence of each source with a source (direction) specific *complex-values Jones matrix*. The naming convention for the associated parameters is “`DirectionalGain:{0,1}:{0,1}:{Real,Imag}|{Ampl,Phase}:<station name>:<source name>`”. Depending on the value of **Phasors.Enable** either `{Real,Imag}` or `{Ampl,Phase}` is selected.

Beam model configuration

> **Beam.Enable** : *bool* = F
>
> Multiply the source coherence of each source with a direction dependent *complex-valued Jones*

matrix that is computed based on a specified beam model. The naming convention for the associated parameters is "`AntennaOrientation:<station name>`". These parameters specify for each station the azimuth of the positive X-dipole direction in radians North over East. The positive Y-dipole direction is then obtained by adding +90 degrees to this orientation.

Beam.Mode : *string* = DEFAULT
One of DEFAULT (include both the element beam and the array factor), ELEMENT (element beam only), ARRAY_FACTOR (array factor only). The array factor represents the effect of station (and tile) beam forming.

Beam.UseChannelFreq : *bool* = F
The frequency dependency is by default ignored. For a single subband, the reference frequency is the frequency used by the station beamformer. **When concatenating multiple subbands in frequency, this is not good: there will be an error which increases with the number of subbands.** In that case, specify UseChannelFreq=T.

Beam.Element.Path : *string* = $LOFARROOT/share

Path to a directory that contains files with coefficients for the dipole beam model (HAMAKER_*). Do not include this field unless you know what you are doing.

Ionosphere model configuration

Ionosphere.Enable : *bool* = F

Introduces a direction dependent phase shift that is computed based on a global phase screen.

Ionosphere.Type : *string*

Selects the ionospheric model to use. The following ionospheric models are currently supported:

MIM

Minimal ionospheric model developed by Maaijke Mevius. Models the ionosphere as a thin TEC layer at a specified height above the Earth's surface. The TEC value at a given pierce point is described by an N-degree polynomial in the pierce point coordinates. The naming convention for the associated parameters is as follows:

"`MIM:Height`"
Height of the thin ionospheric layer above the earth in meters.

"`MIM:Coeff:{0,1,2,…}:{0,1,2,…}`"
Coefficients of the polynomial TEC model.

EXP_ION

Ionospheric model developed by Bas van der Tol. Models the ionosphere as a thin TEC layer at a specified height. The TEC value at a given pierce point is computed from a set of basis functions that have been fitted to known TEC values at a number of calibrator pierce points. The basis functions themselves are determined based on the locations of the calibrator pierce points. The naming convention for the associated parameters is as follows:

TBD

> **Ionosphere.Degree** : *integer*
>
> Defines the degree of the polynomial used to model the ionospheric phase screen. Should be at least 1 (gradient model). **Only relevant for the MIM ionospheric model.**

Condition number flagging configuration

> **Flagger.Enable** : *bool* = F
>
> When correcting the visibility data (CORRECT), flag visibilities based on the condition number of the associated Jones matrix. The condition number is a measure of how numerically well-conditioned a problem is (see e.g. http://en.wikipedia.org/wiki/Condition_number). It ranges between 1.0 and +infinity. Lower values indicate better conditioned problems.
>
> **Flagger.Threshold** : *float*
>
> Visibilities with an associated Jones matrix that has a condition number higher than this threshold will be flagged.

Correct

> **MMSE.Enable** : *bool* = F
>
> Add sigma^2 to the diagonal of the cumulative Jones matrices before inversion. This makes the inverse more robust.
>
> **MMSE.Sigma** : *float* = 0.0
>
> Sigma value to use.

Solve

NB. For global calibration, the current implementation supports only regular iterated least squares (Solve.Algorithm = L2), comparing both amplitude and phase (Solve.Mode = COMPLEX), without outlier rejection (Solve.OutlierRejection.Enable = F).

Algorithm : *string* = L2

> Selects the solving algorithm to use. Current options are: L1 (L1 weighted iterated least squares), L2 (regular iterated least squares). You can change the epsilon values used for L1 weighting by setting **EpsilonL1**.

EpsilonL1 : *list of floats* = [1e-4, 1e-5, 1e-6]

> Epsilon values used for L1 weighted iterated least squares.

Mode : *string* = COMPLEX

> Determines how the (complex valued) data and the (complex valued) model are compared. Options are: COMPLEX, PHASE, AMPLITUDE. The COMPLEX mode compares both phase and amplitude, the PHASE mode compares phases and ignores amplitude differences, the AMPLITUDE mode compares amplitudes and ignores phase differences. NB. Comparing only phases is not necessarily equivalent to phase only calibration (and likewise for comparing amplitudes).

Parms : *list of strings*

> Parameters to fit. Shell style wildcards are recognized (?,*,{}). If specified, this key shall *not* be empty, or an exception will be thrown.

```
Solve.Parms = ["Gain:{0:0,1:1}:*"]
```

ExclParms : *list of strings* = []

> Subset of the parameters selected by **Parms** that should *not* be fitted. For example, if we would like to solve for the gain (amplitude, phase) of each station, but we would also like to fix the phase of the first station (say, CS010_dipole0) this can be specified as follows:

```
Solve.Parms = ["Gain:*"]
Solve.ExclParms = ["Gain:*:Phase:CS010_dipole0"]
```

CalibrationGroups : *list of integers* = []

> A list of numbers that specifies the partitioning of kernels into calibration groups. The n-th number in the list gives the *number of kernels* in the n-th calibration group. For example, [3,1] signifies that there are two calibration groups, where the kernels with id 0, 1, and 2 are assigned to calibration group 0 and the kernel with id 3 is assigned to calibration group 1. The sum of the numbers in the list must always equal the total number of `kernel` processes that participate in the calibration. An empty list means that there are no interdependencies and therefore each kernel can use it's own solver. The global solver will not be used in this case.

CellSize.Freq : *integer*

> Solution grid cell size (frequency): A chunk is divided into solutions cells and a solution is computed for each cell independently. This parameter specifies the (nominal) number of channels that are grouped together in a cell. A value of zero (0) selects the entire frequency range. **NB.** When performing a global solve this parameter is ignored and the frequency range of the calibration group is used instead. **NB.** When **Resample.Enable** is set to true, the number specified here is interpreted as number of channels *after* resampling.

CellSize.Time : *integer*

> Solution grid cell size (time): A chunk is divided into solutions cells and a solution is computed for each cell independently. This parameter specifies the (nominal) number of timeslots that are grouped together in a cell. A value of zero (0) selects the entire time range. **NB.** When **Resample.Enable** is set to true, the number specified here is interpreted as number of timeslots *after* resampling.

CellChunkSize : *integer*

> Specifies the number of solution cells *along the time axis* to process simultaneously. A value of zero (0) selects all solution cells in the current chunk. Can be used to tune memory usage.

PropagateSolutions : *bool* = F

> If set to true, then the solutions of the previous *cell chunk* are used as initial values for the next *cell chunk*. **NB.** Even if set to true, solutions are *not* propagated to the next *chunk of visibility data*. Furthermore, no convergence check is done, so bad solutions may contaminate the solutions in the next *cell chunk* if this parameter is set to true.

UVRange : *list of floats* = []

> A list containing either a single number (minimal UV length), or two numbers (minimal, maximal UV length). All UV lengths should be specified in wavelengths. Visibility data from baselines that do not meet the criterion are ignored when estimating model parameters. An empty list means no UV range selection will be performed.

```
Solve.UVRange = [250.0]    # Use only visibility data from baselines with a
UV length larger than 250.0 wavelengths.
Solve.UVRange = [250.0, 5000.0]    # Use only visibility data from baselines
with a UV length between 250.0 and 5000.0 wavelengths.
```

Solver statistics logging configuration

NB. Solver statistics logging only works for non-global solves that use the L2 algorithm without outlier rejection. Be careful to specify different Log.Name values for different solve steps, else the solver statistics of all steps sharing the same value for Log.Name will be interleaved.

> **Log.Name** : *string* = solver_log
>
> Name of the table that will hold the solver statistics log. This table will be created inside the observation MS.
>
> **Log.Level** : *string* = NONE
>
> Controls which statistics are logged, and how often they are logged. NONE means logging is disabled, PERSOLUTION means statistics will be logged after convergence only, PERITERATION means statistics will be logged each iteration, PERSOLUTION_CORRMATRIX and PERITERATION_CORRMATRIX are similar but after convergence the covariance matrix will be logged as well. (Computing the covariance matrix can be expensive, which is why it is offered as a separate option). Logging per iteration can result in a large table.

Outlier rejection configuration

> **OutlierRejection.Enable** : *bool* = F

> Toggles outlier rejection. With outlier rejection enabled the estimation should be less sensitive to outliers, but take more time.
>
> **OutlierRejection.Threshold** : *list of float* = [7.0, 5.0, 4.0, 3.5, 3.0, 2.8, 2.6, 2.4, 2.2, 2.5]
>
> List of consecutive RMS thresholds used for outlier rejection. Samples with a weighted residual higher than the current threshold times the RMS of the previous solution will be rejected. After convergence, the threshold is recomputed using the next value from the list. Previously flagged outliers are reset and will be reconsidered using the new threshold value.

Phase shift configuration

NB. Phase shifting has been disabled in the current implementation.

> **PhaseShift.Enable** : *bool* = F
>
> If set to true, phase shift the observed visibility data in memory to the direction specified in **PhaseShift.Direction** and use this phased shifted visibility data instead of the observed visibility data when solving.
>
> **PhaseShift.Direction** : *list of strings*
>
> Direction to phase shift the observed visibility data to. This is a list of three strings which denote the epoch, right ascension, and declination in that order.

```
Solve.PhaseShift.Direction = [J2000, 19:59:28.3, +40.44.02].
```

Resampling configuration

NB. Resampling has been disabled in the current implementation.

> **Resample.Enable** : *bool* = F
>
> If set to true, resample the observed visibility data in memory and use this resampled visibility data instead of the observed visibility data when solving. **NB.** Setting this to true influences the way the solution cell sizes (specified as **CellSize.Freq** and **CellSize.Time**, see above) are interpreted.
>
> **Resample.CellSize.Freq** : *integer* = 1
>
> Resample (down sample) factor in number of channels. The specified number of channels will be averaged into a single output channel.
>
> **Resample.CellSize.Time** : *integer* = 1
>
> Resample (down sample) factor in number of timeslots. The specified number of timeslots will be averaged into a single output timeslot.
>
> **Resample.DensityThreshold** : *float* = 1.0
>
> If the number of flagged input samples relative to the total number of input samples that were averaged to form a single output sample exceeds this threshold, the output sample will be flagged.

> The range of this field is [0.0, 1.0]. Note that flagged input samples are never used while computing the average. The threshold only defines when an output sample will be *flagged*. For instance, if 99 out of a 100 input samples are good then the output sample can be considered good as well, but if, say, only 5 out of a input 100 samples are good then maybe the output sample should be flagged.

Solver configuration

> **Options.MaxIter** : *integer* = 0
>
> Convergence criterion: Iteration is halted if the number of iterations exceeds this value. A value of zero means unlimited. **NB.** There is a known off by one bug, so in practice a value of *x* results in *x+1* iterations.
>
> **Options.EpsValue** : *float* = 1e-8
>
> Convergence criterion: Iteration is halted if the minimal relative change in the norm of the solutions is smaller than this value.
>
> **Options.EpsDerivative** : *float* = 1e-8
>
> Convergence criterion: Iteration is halted if the maximum of the known vector of the equations to be solved is less than this value.
>
> **Options.ColFactor** : *float* = 1e-6
>
> Colinearity factor. Used to test for solvability of the normal equations. If not solvable an error is returned, unless **Options.UseSVD** is set to true (in which case Singular Value Decomposition is used to compute a solution). See http://aips2.nrao.edu/docs/scimath/implement/Fitting/LSQFit.html#LSQFit.
>
> **Options.LMFactor** : *float* = 1e-3
>
> Initial Levenberg-Marquardt factor, which controls the balance between gradient descent and Newton-Raphson optimization. See also http://aips2.nrao.edu/docs/notes/224 (bottom of the page).
>
> **Options.BalancedEqs** : *bool* = F
>
> If set to true, the Levenberg-Marquardt factor is added (“in some way?”) to the diagonal elements of the normal equation matrix. Otherwise, the diagonal elements are multiplied by (1 + factor). See http://aips2.nrao.edu/docs/scimath/implement/Fitting/LSQFit.html#LSQFit.
>
> **Options.UseSVD** : *bool* = F
>
> If set to true, Singular Value Decomposition is used to compute a solution when the normal equations cannot be triangularized.