BBS

This page describes BBS (BlackBoard Selfcal), a calibration package for LOFAR observations.

BBS has two modes of running. The first is to run as a stand-alone tool to calibrate one subband. If a single subband does not contain enough signal, BBS offers the possibility to use data from multiple subbands together for parameter estimation. This is called /global/ parameter estimation. On this page, we will focus on the stand-alone version; setting up a global solve is described elsewhere.

Currently this page is work in progress, see Configuration Syntax for the old version

Usage

To calibrate a single MS, execute the `calibrate-stand-alone` script on the command line:

```
calibrate-stand-alone -f <MS> <parset> <source catalog>
```

The important arguments provided to the `calibrate-stand-alone` script in this example are:

- <MS>: name of the MS to process.
- `<parset>`: description of the reduction (see below).
- `<source catalog>`: A list of sources that can be used for calibration (see below). When the MS already contains a sourcedb, it is not necessary to specify a new source catalog if you're calibrating on the same sourcedb.
- The `-f` option, which overwrites stale information from previous runs (remove old instrument table and old source model if they exist with the measurement set).
- The `-t` option, which provides some basic multithreading.

You can run the script without arguments for a description of all the options and the mandatory arguments, such as the `-v` option to get a more verbose output.

Source catalog

A source catalog is a plain text file containing information on your model. An example catalog file is shown below:

```
# (Name,Type,Ra,Dec,I, ReferenceFrequency='55.468e6', SpectralIndex) =
format
3C196, POINT, 08:13:36.062300, +48.13.02.24900, 153.0, , [-0.56, -0.05212]
```

More information about the format is on the makesourcedb page. You can use gsm.py to get a source catalog from an existing catalog.

Creating a PARSET file describing a reduction

The parset file describes the steps in the reduction you want to perform. A very basic example parset is:

```
Strategy.ChunkSize = 100
Strategy.Steps = [predict]

Step.predict.Model.Sources     = []
Step.predict.Operation         = PREDICT
Step.predict.Output.Column     = MODEL_DATA
Step.predict.Model.Beam.Enable = False
```

An especially important key is `Strategy.ChunkSize`, which specifies the amount of data (in timeslots) that is read into main memory as a single block. If your measurement set is large, be sure to fill in a sensible value here. It should be at least as large as the largest `Solve.CellSize.Time` value in your reduction. Also, you do not want to make it too small, because repeatedly reading small chunks hurts performance. It is advisable to ensure all `Solve.CellSize.Time` values in your reduction are divisors of `Strategy.ChunkSize`, to avoid truncated solution cells.

After calibrating

- The tool `parmdbplot.py` can be used to plot the calibration solutions.
- Usually, the corrected data needs another round of flagging before imaging because of bad calibration solutions. Bad solutions may occur for various reasons, e.g. because most of the samples in a solution cell are flagged due to RFI or because some of the calibrators are not above the horizon (when not using a beam model).

Parset keys

Strategy

Parameter	type	default	description
Strategy.InputColumn	string	DATA	Name of the column that contains the input data

Strategy.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. Examples:
			`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.`

Parameter	type	default	description
Strategy.TimeRange	string vector	[]	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. Examples: `Strategy.TimeRange = [27-Jul-2007/16:05:04]` `Strategy.TimeRange = [27-Jul-2007/16:05:04, 28-Jul-2007/13:05:04]`
Strategy.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 the data will be loaded into memory.
Strategy.UseSolver	bool	F	Use a global solver? (If you specify True here, you should use the `calibrate` script.)
Strategy.Steps	string vector	[]	The names of steps that compose the strategy. If you leave this empty, BBS will not do anything (and give an error).

Parameters common to all step types

Parameter	type	default	description
Step.<StepName>.Operation	string		Operation to be performed in this step. Possible values are `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, perform a calibration
Step.<StepName>.Model			Description of the model to be used. Only subkeys of this key should be specified. See below
Step.<StepName>.Baselines	string	*&	Baselines to process. See the key `Strategy.Baselines`
Step.<StepName>.Correlations	string	[]	Correlations to process. If this key is not specified, all correlations will be selected. `Step.simulate.Correlations = [XX,YY]` `Step.simulate.Correlations = [RR,RL]`
Step.<StepName>.Output.Column	string	(empty)	Column in the observation part where the output values of this step should be written. If left empty, no data will be written.
Step.<StepName>.Output.WriteFlags	bool	F	If set to true the flags in the observation will be updated

Parameter	type	default	description
Step.<StepName>.Output.WriteCovariance	bool	F	If set to true, covariance will be written to the covariance column linked to the column specified at `Output.Column` (only for experienced users: the covariance column can be converted to a `WEIGHTS` column by using an external script)

Parameters for solve step

Parameter	type	default	description
Step.<StepName>.Solve.Parms	string vector	[]	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}:*"]`
Step.<StepName>.Solve.ExclParms	string vector	[]	Subset of the parameters selected by Parms that should not be fitted.
Step.<StepName>.Solve.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). If you want to do phase only calibration, specify PHASE here.
Step.<StepName>.Solve.CellSize.Freq	integer	**!**	Solution grid cell size (frequency): A chunk is divided into solutions cells and a solution is computed for each cell in total independently. This parameter specifies the (nominal) number of channels that are grouped together in a cell. NB. When performing a global solve this parameter is ignored and the frequency range of the calibration group is used instead.
Step.<StepName>.Solve.CellSize.Time	integer	**!**	NB. When performing a global solve this parameter is ignored and the frequency range of the calibration group is used instead.
Step.<StepName>.Solve.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.
Step.<StepName>.Solve.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.
Step.<StepName>.Solve.UVRange	float vector	[]	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.`
Step.<StepName>.Solve.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.
Step.<StepName>.Solve.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.
Step.<StepName>.Solve.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.
Step.<StepName>.Solve.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.
Step.<StepName.Solve.Options.LMFactor	float	1e-3	Initial Levenberg-Marquardt factor, which controls the balance between gradient descent and Newton-Raphson optimization.

Parameter	type	default	description
Step.<StepName>.Solve.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.
Step.<StepName>.Solve.Options.UseSVD	bool	F	If set to true, Singular Value Decomposition is used to compute a solution when the normal equations cannot be triangularized.

Parameters for the model

Parameter	type	default	description
Step.<stepname>.Model.Sources	string vector	[]	List of sources to include in the model. Shell style wildcards are allowed. An empty list will select all sources in the sky model.
Step.<stepname>.Model.Cache.Enable	bool	T	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. This will require a lot of memory, but does speed BBS up.
Step.<stepname>.Model.<effect>.Enable	bool	F	Example: `Step.Solve.Model.Clock.Enable=True` to enable the clock.

Effects in the model

Below are all the effects that can be modeled, in the order they are applied (from sky to ground).

Effect	Description	Direction dependent?	Options
`ScalarPhase`	A phase that is equal for both dipoles	Direction dependent	-
`Rotation`	Faraday Rotation without frequency dependency (note: parameter name is `RotationAngle`)	Direction dependent	-
`FaradayRotation`	Faraday Rotation (note: parameter name is `RotationMeasure`	Direction dependent	-
`DirectionalTEC`	TEC (ionosphere)	Direction dependent	-
`Beam`	The LOFAR beam model	Direction dependent	See below
`DirectionalGain`	Directional gain	Direction dependent	`Model.DirectionalGain.Phasors` (True or False, default is False): express parameters in amplitude and phase instead of real and imaginary part
`CommonScalarPhase`	Scalar Phase	Direction independent	-
`CommonRotation`	Rotation (note: parameter name is `CommonRotationAngle`)	Direction independent	-

Effect	Description	Direction dependent?	Options
TEC	TEC (ionosphere)	Direction independent	'Model.TEC.Split' (True or False, default is False): have a separate TEC for X and Y dipoles (this is not physical, but used for testing)
`Gain`	Gain	Direction independent	`Model.Gain.Phasors` (True or False, default is False): express parameters in amplitude and phase instead of real and imaginary part
`Clock`	Clock	Direction independent	`Model.Clock.Split` (True or False, default is False): have a separate clock for X and Y dipoles

Beam model configuration

The beam model tries to emulate all kinds of distortions to the signal that are caused by the beam of the station. These effects are split into two parts: the *element beam*, which is the response of a single dipole, and the *array factor*, which emulates the effect of combining the signal of the many dipoles in a station. In HBA, the array factor model also models the effect of the analog tile beam former.

To have a look at different elements of the beam, you can specifically use only the element beam or only the array factor (if you don't know the details, you need both the element beam and the array factor, which is the default). The options are:

```
Model.Beam.Mode = ELEMENT       # only element beam
Model.Beam.Mode = ARRAY_FACTOR  # only array factor
Model.Beam.Mode = DEFAULT       # both element beam and array factor
(default)
```

The tile beam former in the HBA tiles forms a beam for a certain reference frequency. When modeling this beam, the beam model should of course do this for the same frequency. Usually, this works automatically: the reference frequency is stored in the measurement set. Things are different when you compress a number of subbands as channels into one measurement set (usually done with NDPPP). Then each `channel' was beamformed at a different reference frequency. In this case, the reference frequency is only right for one of the `channels'. To handle this case well, there is an option that tells the beam model to use the channel frequency (which is usually the center frequency of the compressed subband). This option is:

```
Step.Solve.Model.Beam.UseChannelFreq = T
```

Note that the beam model is a direction dependent effect like any other in BBS. That means that over a patch, the beam is assumed to be constant (it is evaluated at the centroid of the patch). This may change in the future.

References

- BBS parameter set documentation (old)

- makesourcedb, with the description of the sky catalog file format
- BBS Exercise from the LOFAR Data Processing School
- Initial report on BBS performance (Joris van Zwieten, Jan 7, 2008)
- Parameter database description (Ger van Diepen)
- Creating (distributed) MeasurementSets (Ger van Diepen)
- Ionospheric calibration in BBS (Maaijke Mevius)