CubiCal Documentation

Author

Mar 17, 2021

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION TO CUBICAL

This is the documentation of CubiCal, a suite of fast calibration routines for radio interferometry.

CubiCal implements several accelerated gain solvers which exploit complex optimisation. Computationally intensive functions have been written in Numba to further accelerate them and multiprocessing is fully supported.

This documentation details the necessary steps for installing CubiCal, as well as how to use it. Additional information on how to get the best performance out of the package is also provided.

CHAPTER

TWO

REQUIREMENTS AND INSTALLATION

2.1 Ubuntu 18.04

CubiCal depends on python-casacore, the dependencies of which should be installed from the KERN-5 ppa. The ppa can be added as follows:

```
sudo apt install software-properties-common
sudo add-apt-repository -s ppa:kernsuite/kern-5
sudo apt-add-repository multiverse
sudo apt-add-repository restricted
sudo apt update
```

Once the ppa has been added, CubiCal's dependencies can be installed as follows:

```
CUBICAL_DEPENDENCIES=(casacore-dev \
                      casacore-data \
                      build-essential \
                      python3-pip \
                      libboost-all-dev \
                      wcslib-dev \
                      git \
                      libcfitsio-dev)
sudo apt install -y $CUBICAL_DEPENDENCIES
```

Warning: A known casacore bug can cause data corruption on large reads. If your use-case falls in this category, please build and install casacore from source.

If you wish to install CubiCal in a virtual environment (recommended), see *Using a virtual environment*.

CubiCal can be installed by running the following:

```
pip3 install git+https://github.com/ratt-ru/CubiCal.git@1.4.0
```

Note: CubiCal can predict model visiblities using Montblanc, but it is not installed by default. To install CubiCal with Montblanc, run:

```
pip3 install "cubical[lsm-support]@git+https://github.com/ratt-ru/CubiCal.git@1.4.0"
```

Warning: To install in development mode, assuming that you have already cloned the repository, run:

```
pip3 install -e path/to/repo/
```

If you require Montblanc, run:

```
pip3 install -e path/to/repo/"[lsm-support]"
```

2.2 Using a virtual environment

Installing CubiCal in a virtual enviroment is highly recommended. To install virtualenv using apt, run:

```
sudo apt install python3-virtualenv
```

To create a virtualenv, run:

```
virtualenv -p python3 path/to/env/name
```

Activate the environment using:

```
source path/to/env/name/bin/activate
```

This should change the command line prompt to be consistent with the virtualenv name.

It is often necessary to update pip, setuptools and wheel inside the environment:

```
pip3 install -U pip setuptools wheel
```

CHAPTER

THREE

BASIC USAGE

Once CubiCal has been successfully installed, it can be run from command line using:

```
gocubical
```

Adding the -h argument will print the help which provides all the command line arguments.

CubiCal can be run in one of two ways; either by specifiying all the necessary arguments via the command line or by specifying a parset file. A parset file can be populated with all the arguments required to run a specific calibration.

A basic parset file looks something like this:

```
[data]
ms = measurement_set.MS
time-chunk = 32
freq-chunk = 32

[model]
list = sky_model.lsm.html

[montblanc]
dtype = double
feed-type = circular
mem-budget = 4096

[sol]
jones = G

[out]
column = CCORRECTED_DATA

[g]
time-int = 8
freq-int = 8
```

If the above parset was named basic.parset, it could be run by invoking:

```
gocubical basic.parset
```

This simple example only uses a fraction of the available options - unspecified options are populated from the defaults. Square bracketed values are section headers which correspond to the first part of the associated command line argument e.g. the ms value in the [data] section would be specified on the command line as:

```
gocubical --data-ms D147-LO-NOIFS-NOPOL-4M5S.MS/
```

This relationship can be inverted to add options to the parset. Consider the following example:

```
gocubical --dist-ncpu 4
```

Adding this to basic.parset is as simple as adding the [dist] section (the first part of the command line argument), and specifying ncpu. basic.parset would then look as follows:

```
[data]
ms = measurement_set.MS
time-chunk = 32
freq-chunk = 32

[model]
list = sky_model.lsm.html

[montblanc]
dtype = double
feed-type = circular
mem-budget = 4096

[sol]
jones = G

[out]
column = CCORRECTED_DATA

[g]
time-int = 8
freq-int = 8

[dist]
ncpu = 4
```

Note that a parset can be combined with options specified on the command line - the command line options will take precedence, making it easy to experiment without having to create a new parset.

CHAPTER

FOUR

PARSET OPTIONS

This page details available parset options. However, invoking `gocubical -h` should be preferred as it will always be up-to-date. The following is broken up into the various sections of the parset.

Note: These parset options can be specified via command line. For inclusion in a .parset file, omit the leading `--section-` component and place the remainder in the appropriate section.

4.1 [data]

Options pertaining to data selection and chunking strategy.

--data-ms=string Name/path of input measurement set. Mandatory.

--data-column=string Name of measurement set column from which to read for input data (uncalibrated visibilities). Default: 'DATA'.

--data-time-chunk=string Data will be cut up into blocks containing this many timeslots. This limits the amount of data processed at once. Smaller chunks allow for a smaller RAM footprint and greater parallelism, but this sets an upper limit on the solution intervals that may be employed. Specify as an integer number of timeslots, or a value with a unit (e.g. '300s'). 0 means use full time axis. Default: 32.

--data-freq-chunk=string Data will be cut up into blocks containing this many channels. This limits the amount of data processed at once. Smaller chunks allow for a smaller RAM footprint and greater parallelism, but this sets an upper limit on the solution intervals that may be employed. Specify as an integer number of channels, or a value with a unit (e.g. '128MHz'). 0 means use full frequency axis. Default: 32.

--data-chunk-by=string If set, then time chunks will be broken up whenever the value in the named column(s) jumps by `--data-chunk-by-jump`. Multiple column names may be given, separated by commas. Set to None to disable. Default: SCAN_NUMBER.

--data-chunk-by-jump=int The jump size used in conjunction with `--data-chunk-by`. If 0, then any change in value is a jump. If n, then the change must be >n.

--data-rebin-time=string Rebin data in time on the fly. Specify as a number of timeslots to average together, or a value with a unit (e.g. '5s'). Default: 1

--data-rebin-freq=string Rebin data in frequency on the fly. Specify as a number of channels to average together, or a value with a unit (e.g. '4MHz'). Default: 1

--data-single-chunk=string Each data chunk is assigned a unique identifier, e.g. 'D0T0F0'. If set, processes just one chunk of data matching the identifier. Primarily a debugging option. No default.

4.2 [sel]

Options pertaining to data selection.

--sel-field=int FIELD_ID to read from the MS. Default: 0.

--sel-ddid=multi DATA_DESC_IDs to read from the MS. Can be specified as e.g. "5", "5,6,7", "5~7" (inclusive range), "5:8" (exclusive range), "5:" (from 5 to last). Default reads all.

--sel-taql=string Additional TaQL selection string. Combined with other selection options. No default.

--sel-chan=multi Channels to read (within each DDID). Can be specified as e.g. "5", "10~20" (10 to 20 inclusive), "10:21" (same), "10:" (from 10 to end), ":10:2" (0 to 9 inclusive, stepped by 2), "~9:2" (same). Default reads all.

--sel-diag=bool Use diagonal (i.e. parallel hand) correlations only. Off-diagonals will be null in the outputs. See also –JONES-diag-only.

4.3 [model]

Options related to model selection and prediction.

--model-list=multi Predict model visibilities from MS column/s and LSM/s (using Montblanc). The simplest usage is to specify a column, e.g. `MODEL_DATA`, or a Tigger LSM, e.g. `skymodel.lsm.html`, or, as a special case, `1` simply uses unity visibilities. (The LSM option is only available if Montblanc is installed). Use @TAG, e.g. `skymodel.lsm.html@dE` to group sources in the LSM by direction, according to the specified tag. You can also specify models for different directions by means of a colon. For example, `MODEL_DATA_1:MODEL_DATA_2` defines two directions, while `MODEL_DATA:skymodel.lsm.html@dE` defines a direction modelled by the `MODEL_DATA` column, and other directions as defined by the LSM. By contrast, the plus sign adds model visibilities together without splitting them into directions, e.g. `MODEL_DATA_1+MODEL_DATA_2:skymodel.lsm.html@dE` will define one direction modelled by a sum of columns, and other directions as defined by the LSM. Finally, a comma separates *model sets*. Each model set defines a separate minimization problem, weighted differently (in which case the –weight-column option must specify the same number of comma-separated *weight sets*). The priority of the separators is as follows: first, commas split up model sets. Within each set, colons split up directions. Finally, within each direction, plus signs split up its additive components. This can be quite a complex option, so check the log to make sure it is being interpreted correctly. No default.

--model-ddes=keyword Enables direction-dependent models. If auto, this is determined by `--sol-jones` and `--model-list`, otherwise, enable/disable explicitly. Keywords: never, auto, always. Default: auto.

--model-beam-pattern=string Apply beams from specified .fits files eg. "beam_$(corr)_$(reim).fits" or "beam_$(CORR)_$(REIM).fits". No default.

--model-beam-l-axis=keyword Specify which axis in the .fits file is associated with the l axis. Keywords: X, Y, -X, -Y. No default.

--model-beam-m-axis=keyword Specify which axis in the .fits file is associated with the m axis. Keywords: X, Y, -X, -Y. No default.

--model-feed-rotate=multi Apply a feed angle rotation to the model visibilities. Use 'auto' to read angles from FEED subtable, or give an explicit value in degrees. Default: auto.

--model-pa-rotate=bool Apply parallactic angle rotation to model visibilities. Enable this for alt-az mounts, unless your model visibilities are already rotated. Default: True.

4.4 [weight]

Options related to weights.

--weight-column=string Column/s to read weights from. Weights are applied by default. Specify an empty string or None to disable. Default: WEIGHT_SPECTRUM.

--weight-fill-offdiag=bool Fill off-diagonal weights from geometric mean of diagonal weights. Use this if you have missing off-diagonal weights for whatever reason. Default: False.

4.5 [montblanc]

Options which will be used during model prediction (using Montblanc.)

--montblanc-device-type=keyword Use CPU or GPU for simulation. Keywords: CPU, GPU. Default: CPU.

--montblanc-dtype=keyword Precision for simulation. Keywords: float, double. Default: float.

--montblanc-feed-type=keyword Simulate using linear or circular feeds. Keywords: linear, circular. Default: linear.

--montblanc-mem-budget=int Memory budget in MB for simulation. Default: 1024.

--montblanc-verbosity=keyword Verbosity level of Montblanc's console output. Keywords: DEBUG, INFO, WARNING, ERROR. Default: WARNING.

--montblanc-threads=int Number of OMP threads to run for Montblanc. Note that –dist-pin-io overrides this, if set. If 0, uses Montblanc's insternal default (all). Default: 0.

4.6 [degridding]

Options for the degridder. Only in use when predicting from DicoModels using DDFacet.

--degridding-OverS=int Oversampling factor. Default: 11.

--degridding-Support=int CF support size. Default: 7.

--degridding-Nw=int Number of w-planes. Default: 100.

--degridding-wmax=float Maximum w coordinate. Visibilities with larger w will not be gridded. If 0, no maximum is imposed. Default: 0.

--degridding-Padding=float Facet padding factor. Default: 1.7.

--degridding-NDegridBand=int Number of image bands for degridding. 0 means degrid each channel. Default: 16.

--degridding-MaxFacetSize=float Maximum facet size in degrees. Default: 0.25.

--degridding-MinNFacetPerAxis=int Minimum number of facets per direction. Default: 1.

--degriding-NProcess=int Number of subprocesses to use in degridding-based predict. Default: 8.

4.7 [flags]

Options controlling how flags are applied and written to.

--flags-apply=string Which flagsets will be applied prior to calibration. Use "-FLAGSET" to apply everything except the named flagset ("-cubical" is useful, to ignore the flags of a previous CubiCal run). Default: -cubical.

--flags-auto-init=string Insert BITFLAG column if it is missing, and initialize a named flagset from FLAG and FLAG_ROW. Default: legacy.

--flags-save=string Save flags to named flagset in BITFLAG. If none or 0, will not save. Default: cubical.

--flags-save-legacy=keyword Controls whether output flags are written to FLAG/FLAG_ROW. Is set to 'auto', then follows the –flag-save option. Default: auto

--flags-reinit-bitflags=bool If true, reninitializes BITFLAG column from scratch. Useful if you ended up with a botched one, but be careful what the state of the FLAG/FLAG_ROW column is when you use this option. Default: 0.

--flags-warn-thr=float If more than this fraction of data is flagged by the solver, issues gentle warnings. Default: 0.3.

--flags-see-no-evil=bool Proceed even if flag columns appear to be botched or damaged. Default: 0.

4.8 [madmax]

"Mad Max" flags visibilities on-the-fly inside the solution loop, by using a MAD filter. This computes the median absolute residual (i.e. median absolute deviation from zero), and flags visibilities exceeding the thresholds set below.

--madmax-enable=bool Enable Mad Max flagging. Default: 0

--madmax-estimate=keyword MAD estimation mode. Use 'corr' for a separate estimate per each baseline and correlation. Otherwise, a single estimate per baseline is computed using 'all' correlations, or only the 'diag' or 'offdiag' correlations. Default: 'corr'

--madmax-diag=bool Flag on on-diagonal (parallel-hand) residuals. Default: 1.

--madmax-offdiag=bool Flag on off-diagonal (cross-hand) residuals. Default: 1

--madmax-threshold=list Threshold for MAD flagging per baseline (specified in sigmas). Residuals exceeding S*MAD/1.428 (where S is the given threshold) will be flagged. MAD is computed per baseline. This can be specified as a list e.g. N1,N2,N3,... The first value is used to flag residuals before a solution starts (use 0 to disable), the next value is used when the residuals are first recomputed during the solution several iteratins later (see -chi-int), etc. A final pass may be done at the end of

the solution. The last value in the list is reused if necessary. Using a list with gradually decreasing values may be sensible. Default: 0,10.

--madmax-global-threshold=list Threshold for global MMAD flagging. MMAD is computed as the median of the per-baseline MADs. Residuals exceeding S*MMAD/1.428 (where S is the given threshold) will be flagged.Can be specified as a list, with the same semantics as –madmax-threshold. Default: 0,12.

--madmax-plot=keyword Enable plots for Mad Max flagging. Use 'show' to show figures interactively, or '1' to save plots to files instead. Plots will show the worst flagged baseline, and a median flagged baseline, provided the fraction of flagged visibilities is above some threshold. Default: 0

--madmax-plot-frac-above=float Threshold (in terms of fraction of visibilities flagged) above which Mad Max plots will be generated. Default: 0.01.

--madmax-plot-bl=str Plot given baseline regardless (multiple baseline IDs may be separated by commas), No default.

--madmax-flag-ant=bool Flag antennas with excessive residuals, based on MAD criterion. Note that currently –madmax-plot must be enabled for this to work. Default: False.

--madmax-flag-ant-thr=float Threshold (in sigmas) used to flag bad antennas. Default: 5.

4.9 [postmortem]

Postmortem flagging is done on things like chi-square statistics after a solutionis finished.

--postmortem-enable=bool If True, will do an extra round of flagging at the end (post-solution) based on solution statistics, as per the following options. Default: 0.

--postmortem-tf-chisq-median=float Intervals with chi-squared values larger than X times the median chi-square value will be flagged. Default: 1.2.

--postmortem-tf-np-median=float Intervals with a number of valid point less than X times the median number of valid points will be flagged. Default: 0.5.

--postmortem-time-density=float If more than the given fraction of data in a timeslot is flagged, flag entire timeslot. Default: 0.5.

--postmortem-chan-density=float If more than the given fraction of data in a timeslot is flagged, flag entire channel. Default: 0.5.

--postmortem-ddid-density=float If more than the given fraction of data in a DDID is flagged, flag entire DDID. Default: 0.5.

4.10 [sol]

Options pertaining to the solver.

--sol-jones=multi Comma-separated list of Jones terms to enable, e.g. "G,B,dE". These tags must correspond to the user-defined gain templates at the bottom of the .parset file. Default: G.

--sol-precision=keyword Solve in single or double precision. Keywords: 32, 64. Default: 32.

--sol-delta-g=float Theshold for gain accuracy - gains which improve by less than this value are considered converged. Default: 1e-6.

--sol-delta-chi=float Theshold for solution stagnancy - if the chi-squared is improving by less than this value, the gain is considered stalled. Default: 1e-6.

--sol-chi-int=int Number of iterations to perform between chi-suqared checks. This is done to avoid computing the expensive chi-squared test evey iteration. Default

--sol-last-rites=bool Re-estimate chi-squred and noise at the end of a solution cycle. Disabling last rites can save a bit of time, but makes the post-solution stats less informative. Default: 1.

--sol-stall-quorum=float Minimum percentage of solutions which must have stalled before terminating the solver. Default: 0.99.

--sol-term-iters=multi Number of iterations per Jones term. If empty, then each Jones term is solved for once, up to convergence, or up to its -max-iter setting. Otherwise, set to a list giving the number of iterations per Jones term. For example, given two Jones terms and `--sol-term-iters 10,20,10` it will do 10 iterations on the first term, 20 on the second, and 10 again on the first. No default.

--sol-min-bl=float Min baseline length to include in solution. Default: 0.

--sol-max-bl=float Max baseline length to include in solution. If 0, no maximum is applied. Default: 0.0.

--sol-subset=str Additional subset of data to actually solve for. Any TaQL string may be used. No default.

4.11 [bbc]

Options related to baseline-based corrections.

--bbc-load-from=str Load and apply BBCs computed in a previous run. Apply with care! This will tend to suppress all unmodelled flux towards the centre of the field. No default.

--bbc-compute-2x2=bool Compute full 2x2 BBCs (as opposed to diagonal-only). Only useful if you really trust the polarisation information in your sky model. Default: 0.

--bbc-apply-2x2=bool Apply full 2x2 BBCs (as opposed to diagonal-only). Only enable this if you really trust the polarisation information in your sky model. Default: 0.

--bbc-save-to=str Compute suggested BBCs at end of run, and save them to the given database. It can be useful to have this always enabled, since the BBCs provide useful diagnostics of the solution quality (and are not actually applied without a load-from setting). (default: "{data[ms]}/BBC- field:{sel[field]}-ddid:{sel[ddid]}.parmdb")

--bbc-per-chan=bool Compute BBCs per-channel (instead of across the entire band). Default: 1.

--bbc-plot=bool Generate output BBC plots. Default: 1.

4.12 [dist]

Options related to parallelism.

--dist-ncpu=int Max number of CPU cores to use. 0 disables parallelism. Default: 0.

--dist-nworker=int Number of worker processes to launch (excluding the IO worker). When 0, determined automatically from the `--dist-ncpu`. Default: 0.

--dist-nthread=int Number of OMP threads to use. When 0, determine automatically. Default: 0.

--dist-max-chunks=int Maximum number of time/freq data-chunks to load into memory simultaneously. If 0, then as many as possible will be loaded. Default: 0.

--dist-min-chunks=int Minimum number of time/freq data-chunks to load into memory simultaneously. If 0, determined automatically. Default: 0.

--dist-pin=multi If empty or None, processes will not be pinned to cores. Otherwise, set to the starting core number, or "N:K" to start with N and step by K. Default: 0.

--dist-pin-io=bool If not 0, pins the I/O & Montblanc process to a separate core, or cores (if `--montblanc-threads` is specified). Ignored if `--dist-pin` is not set. Default: 0.

--dist-pin-main=keyword If set, pins the main process to a separate core. If set to "io", pins it to the same core as the I/O process, if I/O process is pinned. Ignored if `--dist- pin` is not set. Keywords: 0, 1, io. Default: io.

4.13 [out]

Options controlling output locations and types.

--out-dir=str Base name of directory for output files. The suffix .cc-out will be implicitly appended, unless OUTDIR ends with a slash. Default: cubical.

--out-name=str Base name for output files. Full base path will be OUTDIR[.cc-out]/OUTNAMExxx, unless OUTNAME contains a slash, in which case OUTDIR is ignored and OUTNAME is taken to be a full base path. Default: cc.

--out-overwrite=bool Allow overwriting of existing output files. If this is set, and the output parset file exists, will raise an exception. Default: False.

--out-backup=bool Allow automatic backup of existing output directories. Automatic backup is only used when OUTDIR is used (i.e. OUTNAME doesn't contain any slashes), and it ends with .cc-out (implicitly or explicitly). In this case, existing output directories are renamed to .cc.out.0, .1, etc. Default: True.

--out-mode=keyword Operational mode. [so] solve only; [sc] solve and generate corrected visibilities; [sr] solve and generate corrected residuals; [ss] solve and generate uncorrected residuals; [ac] apply solutions, generate corrected visibilities; [ar] apply solutions, generate corrected residuals; [as] apply solutions, generate uncorrected residuals. Keywords: so, sc, sr, ss, ac, ar, as. Default: sc.

--out-apply-solver-flags=bool Apply solver flags when writing new data to measurement set. Default: True.

--out-derotate=multi Explicitly enables or disables derotation of output visibilities. Default (None) is to use the –model-pa-rotate and –model-feed-rotate settings. Options: None | 0 | 1.

--out-column=str Output MS column name (if applicable). Default: CORRECTED_DATA.

--out-model-column=str If set, model visibilities will be written to the specified column. No default.

--out-weight-column=str If set, weights from the Robust Solver will be written to the specified column. This should be set only if we are using the robust solver. No default.

--out-reinit-column=bool Reinitialize output MS column. Useful if the column is in a half-filled or corrupt state. Default: 0.

--out-subtract-model=int Index of model to subtract, if generating residuals. Default: 0.

--out-subtract-dirs=multi Which model directions to subtract, if generating residuals. ":" subtracts all. Can also be specified as "N", "N:M", ":N", "N:", "N,M,K". Default: :.

--out-plots=bool Generate summary plots. Default: 1.

--out-plots-show=bool Show summary plots interactively. Default: 0.

--out-casa-gaintables=bool Export gaintables to CASA caltable format. Tables are exported to same directory as set for cubical databases. Default: 1.

4.14 [log]

Options to allow control of logging functionality.

--log-memory=bool Log memory usage. Default: 1.

--log-boring=bool Disable progress bars and some console output. Default: 1.

--log-append=bool Append to log file if it exists. Default: 0.

--log-verbose=multi Default console output verbosity level. Can either be a single number, or a sequence of "name=level,name=level,..." assignments. Default: 0.

--log-file-verbose=multi Default logfile output verbosity level. Can either be a single number, or a sequence of "name=level,name=level,..." assignments. If None, then this simply follows the console level. Default: None.

4.15 [debug]

Options pertaining to debugging. Mainly for developers.

--debug-pdb=bool Jumps into pdb on error. Default: 0.

--debug-panic-amplitude=float Throw an error if a visibility amplitude in the results exceeds the given value. Useful for troubleshooting. Default: 0.0.

--debug-stop-before-solver=bool Invoke pdb before entering the solver. Default: 0.

4.16 [gainterm]

Options pertaining to a specific gain term. This is not a unique section in the parset. Each gain term specified in `--sol-jones` must have a (not necessarily complete) section like this one. For the example given in `--sol-jones`, there should be three separate sections like this, one for [g], [b] and [de] respectively. Their options will be specified by `--g-`, `--b-` and `--de-` respectively.

--gainterm-solvable=bool Set to 0 (and specify -load-from or -xfer-from) to load a non-solvable term from disk. Not to be confused with `--sol-jones`, which determines the active Jones terms. Default: 1.

--gainterm-type=keyword Type of Jones matrix to solve for. Note that if multiple Jones terms are enabled, then only complex- 2x2 is supported. Keywords: complex-2x2, complex-diag, phase-diag, robust-2x2, f-slope, t-slope, tf-plane. Default: complex-2x2.

--gainterm-load-from=str Load solutions from given database. The DB must define solutions on the same time/frequency grid (i.e. should normally come from calibrating the same pointing/observation). By default, the Jones matrix label is used to form up parameter names, but his may be overridden by adding an explicit "//LABEL" to the database filename. No default.

--gainterm-xfer-from=str Transfer solutions from given database. Similar to `-load-from`, but solutions will be interpolated onto the required time/frequency grid, so they can originate from a different field (e.g. from a calibrator). (default:)

--gainterm-save-to=str Save solutions to given database. Default: {data[ms]} /{JONES}-field:{sel[field]}-ddid:{sel[ddid]}.parmdb.

--gainterm-dd-term=bool Determines whether this term is direction dependent. `--model-ddes` must be enabled. Default: 0.

--gainterm-fix-dirs=multi For DD terms, makes the listed directions non- solvable. No default.

--gainterm-update-type=keyword Determines update type. This does not change the Jones solver type, but restricts the update rule to pin the solutions within a certain subspace: 'full' is the default behaviour; 'diag' pins the off-diagonal terms to 0; 'phase-diag' also pins the amplitudes of the diagonal terms to unity; 'amp-diag' also pins the phases to 0. Keywords: full, phase-diag, diag, amp-diag. Default: full.

--gainterm-time-int=int Time solution interval for this term. Default: 1.

--gainterm-freq-int=int Frequency solution interval for this term. Default: 1.

--gainterm-max-prior-error=float Flag solution intervals where the prior error estimate is above this value. Default: 0.1.

--gainterm-max-post-error=float Flag solution intervals where the posterior variance estimate is above this value. Default: 0.1.

--gainterm-low-snr-warn=float Trigger SNR warning to the user at this threshold. Default: 75.

--gainterm-high-gain-var-warn=float Trigger posterior gain variance warning to the user at this threshold. Default: 30.

--gainterm-clip-low=float Amplitude clipping - flag solutions with diagonal amplitudes below this value. Default: 0.1.

--gainterm-clip-high=float Amplitude clipping - flag solutions with any amplitudes above this value. 0 disables. Default: 10.0.

--gainterm-clip-after=int Number of iterations after which to start clipping this gain. Default: 5.

--gainterm-max-iter=int Maximum number of iterations spent on this term. Default: 20.

--gainterm-epsilon=float Convergence threshold. Solutions that change by less than this value are considered converged. Default: 1e-6.

--gainterm-delta-chi=float Threshold for solution stagnancy – if the chi-squared is improving by less (relatively), then the solution is marked as stalled. Default: 1e-6.

--gainterm-conv-quorum=float Minimum percentage of converged solutions to accept. Default: 0.99.

--gainterm-ref-ant=int Reference antenna - its phase is guaranteed to be zero. Default: None.

--gainterm-prop-flags=keyword Flag propagation policy. Determines how flags raised on gains propagate back into the data. Options are 'never' to never propagate, 'always' to always propagate, 'default' to only propagate flags from direction-independent gains. Keywords: never, always, default. Default: default.

--gainterm-estimate-pzd=bool Estimate phase-zero difference and initialize the gains with it. Use for polarization calibration. Default: False.

--gainterm-diag-only=bool Use only diagonal (parallel-hand) data and model terms for the solution. Note that gains are still applied to the full 2x2 data (unless –sel-diag is also set). Default: False.

--gainterm-offdiag-only=bool Use only off-diagonal data and model terms for the solution, and only solve for off-diagonal Jones elements, pinning the on-diagonals to 1. Default: False.

--gainterm-robust-cov=keyword Determines how the residuals covariance matrix is computed if the robust-2x2 solver is selected. Options are 'compute' to compute normaly, 'identity' to set the covariance to 1 (identity matrix) as in the Robust-t paper, and 'hybrid' which is the default computes the covaraince matrix C but sets it to 1 if the elements are greater than 1. Keywords: compute | identity | hybrid.

--gainterm-robust-scale=bool Scales down the residuals covariance matrix. Simulations show that this improves the results with unmodelled sources. Default: True.

--gainterm-robust-npol=int The number of correlations (polarizations) actually present in the visibilities. This option only applies if the robust-2x2 solver is selected. Expectings 2 or 4 correlations. Default: 2.

--gainterm-robust-int=int Number of iterations after which the v-parameter is recomputed for the robust solver. Default: 1.

--gainterm-robust-save-weights=bool Determines if the appied weights from the robust-2x2 solver are stored. This option only applies if the robust-2x2 solver is selected. If this option is set, output-weight-column must be set too. Default: False.

CHAPTER

FIVE

EXAMPLE PARSETS

This page provides some example parsets for common calibration tasks. To use one of the following parsets, simply copy the text into an empty file and save as a .parset.

Note: The parsets provided in this section will work, but do not include every available option. For a full list, please see *Parset Options*.

5.1 Phase-only selfcal

The following performs diagonal, phase-only selfcal using a measurement set column and produces corrected residuals.

```
[data]
_Help = Visibility data options
ms = ms.ms                              # Your measurement set name.
column = DATA                           # Column in which the data lives.
time-chunk = 100                        # Number of time slots in a chunk.
                                        # Implicitly controls memory footprint.
freq-chunk = 64                         # Number of frequency channels in a chunk.
                                        # Implicitly controls memory footprint.

[sel]
_Help = Data selection options
field = 0                               # Select a field/fields.
ddid = None                             # Select spectral window/s.
chan =                                  # Select channel/s.

[out]
_Help = Options for output products
dir = cubical                           # Output directory name.
name = cubical.cc-out/cubical           # Output path for non-visibility outputs.
overwrite = True                        # Overwrite existing output.
mode = sr                               # Produce corrected residuals.
                                        # Check --help for other options.
column = CORRECTED_DATA                 # Output visibility column.

[model]
_Help = Calibration model options
list = MODEL_DATA                       # Input visiblity column.
                                        # Can also be a Tigger .lsm file.
```

(continues on next page)

(continued from previous page)

```
[weight]
_Help = Weighting options
column = None                           # Weight column to use.

[flags]
_Help = General flagging options
apply = -cubical                        # Ignore existing cubical flags.

[sol]
_Help = Solution options which apply at the solver level
jones = G                               # Term/s to solve for.
                                        # Corresponding section/s below.
delta-g = 1e-06                         # Stopping criteria on gains.
delta-chi = 1e-06                       # Stopping criteria on chi-squared.
term-iters = 20                         # Iteration recipe.
                                        # Overrides individual term settings.

[dist]
_Help = Parallelization and distribution options
ncpu = 4                                # Number of processes to use.
                                        # Implicitly controls memory footprint.
max-chunks = 12                         # Maximum number of simultaneous chunks.
                                        # Ideally >= ncpu.
min-chunks = 12                         # Minimum number of simultaneous chunks.
                                        # Ideally = ncpu.
safe = 1.0                              # Fraction of total memory CubiCal may use.
                                        # Will crash deliberately if exceeded.

[g]
_Help = Options for G-Jones term
label = G                               # This term's name.
                                        # Must match [sol] section's jones option.
type = phase-diag                       # Solve for a diagonal phase-only gain.
time-int = 1                            # Number of timeslots per solution.
freq-int = 1                            # Number of channels per solution.
max-iter = 20                           # Maximum number of iterations.
                                        # May be ignored if [sol] term-iters is set.
_Templated = True                       # Other parameters polulated from defaults.
                                        # Can be safely ignored.
```

5.2 Phase and amplitude selfcal

The following performs diagonal, phase and amplitude selfcal using a Tigger .lsm sky model and produces corrected data. Note that using sky models requires CubiCal to be installed with lsm-support.

```
[data]
_Help = Visibility data options
ms = ms.ms                              # Your measurement set name.
column = DATA                           # Column in which the data lives.
time-chunk = 100                        # Number of time slots in a chunk.
                                        # Implicitly controls memory footprint.
freq-chunk = 64                         # Number of frequency channels in a chunk.
                                        # Implicitly controls memory footprint.
```

(continues on next page)

(continued from previous page)

```
[sel]
_Help = Data selection options
field = 0                               # Select a field/fields.
ddid = None                             # Select spectral window/s.
chan =                                  # Select channel/s.

[out]
_Help = Options for output products
dir = cubical                           # Output directory name.
name = cubical.cc-out/cubical           # Output path for non-visibility outputs.
overwrite = True                        # Overwrite existing output.
mode = sc                               # Produce corrected data.
                                        # Check --help for other options.
column = CORRECTED_DATA                 # Output visibility column.

[model]
_Help = Calibration model options
list = skymodel.lsm.html                # Input sky model .lsm file.
                                        # Can also be a measurement set column.

[weight]
_Help = Weighting options
column = None                           # Weight column to use.

[flags]
_Help = General flagging options
apply = -cubical                        # Ignore existing cubical flags.

[sol]
_Help = Solution options which apply at the solver level
jones = G                               # Term/s to solve for.
                                        # Corresponding section/s below.
delta-g = 1e-06                         # Stopping criteria on gains.
delta-chi = 1e-06                       # Stopping criteria on chi-squared.
term-iters = 20                         # Iteration recipe.
                                        # Overrides individual term settings.

[dist]
_Help = Parallelization and distribution options
ncpu = 4                                # Number of processes to use.
                                        # Implicitly controls memory footprint.
max-chunks = 12                         # Maximum number of simultaneous chunks.
                                        # Ideally >= ncpu.
min-chunks = 12                         # Minimum number of simultaneous chunks.
                                        # Ideally = ncpu.
safe = 1.0                              # Fraction of total memory CubiCal may use.
                                        # Will crash deliberately if exceeded.

[g]
_Help = Options for G-Jones term
label = G                               # This term's name.
                                        # Must match [sol] section's jones option.
type = complex-diag                     # Solve for a diagonal complex gain.
time-int = 1                            # Number of timeslots per solution.
freq-int = 1                            # Number of channels per solution.
max-iter = 20                           # Maximum number of iterations.
                                        # May be ignored if [sol] term-iters is set.
```

(continues on next page)

(continued from previous page)

```
_Templated = True                       # Other parameters polulated from defaults.
                                        # Can be safely ignored.
```

5.3 Gain and bandpass selfcal

The following performs gain and bandpass calibration simultaneously, using a measurement set column as input and produces uncorrected residuals.

```
[data]
_Help = Visibility data options
ms = ms.ms                              # Your measurement set name.
column = DATA                           # Column in which the data lives.
time-chunk = 100                        # Number of time slots in a chunk.
                                        # Implicitly controls memory footprint.
freq-chunk = 64                         # Number of frequency channels in a chunk.
                                        # Implicitly controls memory footprint.

[sel]
_Help = Data selection options
field = 0                               # Select a field/fields.
ddid = None                             # Select spectral window/s.
chan =                                  # Select channel/s.

[out]
_Help = Options for output products
dir = cubical                           # Output directory name.
name = cubical.cc-out/cubical           # Output path for non-visibility outputs.
overwrite = True                        # Overwrite existing output.
mode = ss                               # Produce uncorrected residuals.
                                        # Check --help for other options.
column = CORRECTED_DATA                 # Output visibility column.

[model]
_Help = Calibration model options
list = MODEL_DATA                       # Input visiblity column.
                                        # Can also be a Tigger .lsm file.

[weight]
_Help = Weighting options
column = None                           # Weight column to use.

[flags]
_Help = General flagging options
apply = -cubical                        # Ignore existing cubical flags.

[sol]
_Help = Solution options which apply at the solver level
jones = B,G                             # Term/s to solve for.
                                        # Corresponding section/s below.
delta-g = 1e-06                         # Stopping criteria on gains.
delta-chi = 1e-06                       # Stopping criteria on chi-squared.
term-iters = [20, 20, 20, 20]           # Iteration recipe. Loops over jones above.
                                        # This will do 20 iterations on B,
                                        # 20 on G, 20 on B and finally 20 on G.
```

(continues on next page)

(continued from previous page)

```
                                        # Overrides individual term settings.

[dist]
_Help = Parallelization and distribution options
ncpu = 4                                # Number of processes to use.
                                        # Implicitly controls memory footprint.
max-chunks = 12                         # Maximum number of simultaneous chunks.
                                        # Ideally >= ncpu.
min-chunks = 12                         # Minimum number of simultaneous chunks.
                                        # Ideally = ncpu.
safe = 1.0                              # Fraction of total memory CubiCal may use.
                                        # Will crash deliberately if exceeded.

[g]
_Help = Options for G-Jones term
label = G                               # This term's name.
                                        # Must match [sol] section's jones option.
type = complex-2x2                      # Solve for a full 2x2 complex gain.
                                        # This can be restricted using update type.
update-type = phase-diag                # Discard amplitude and off diagonal
                                        # components of the solution.
                                        # This makes the term phase-only.
time-int = 0                            # Number of timeslots per solution.
                                        # 0 is the entire chunk axis.
freq-int = 1                            # Number of channels per solution.
max-iter = 20                           # Maximum number of iterations.
                                        # May be ignored if [sol] term-iters is set.
_Templated = True                       # Other parameters polulated from defaults.
                                        # Can be safely ignored.

[b]
_Help = Options for G-Jones term
label = b                               # This term's name.
                                        # Must match [sol] section's jones option.
type = complex-2x2                      # Solve for a full 2x2 complex gain.
                                        # This can be restricted using update type.
time-int = 1                            # Number of timeslots per solution.
freq-int = 0                            # Number of channels per solution.
                                        # 0 is the entire chunk axis.
max-iter = 20                           # Maximum number of iterations.
                                        # May be ignored if [sol] term-iters is set.
_Templated = True                       # Other parameters polulated from defaults.
                                        # Can be safely ignored.
```

5.4 Direction-independent and direction-dependent selfcal

The following performs DI and DD gain calibration simultaneously, using a tagged sky model as input and produces corrected residuals. Note that using sky models requires CubiCal to be installed with lsm-support.

Note: DD model specification in CubiCal (via `--model-list` or the appropriate section of the parset) is flexible, allowing the use of both sky models and measurement set columns in fairly complex configurations. Here are some examples:

- `COL_NAME1:COL_NAME2` This will create a model with two directions, one for each of the supplied measure-

ment set columns.

- `skymodel.lsm.html+-COL_NAME:COL_NAME` This will create a model with two directions, one containing the visibilities assosciated with the sky model minus the contribution of the MS column and the other containing just the MS column.
- `skymodel.lsm.html:COL_NAME1:COL_NAME2` This will create a model with three directions, one containing the visibilities assosciated with the sky model, the second containing the visibilities from the first MS column and the third containing the visibilities of the second MS column.
- `COL_NAME1+COL_NAME2:skymodel.lsm.html@dE` This will create a model with at least two directions. This first will contain the sum of the specified MS columns and the remaining will be generated from the dE tagged sources in the sky model.

The following example makes use of a tagged Tigger .lsm file to predict visibilities in several directions.

```
[data]
_Help = Visibility data options
ms = ms.ms                                  # Your measurement set name.
column = DATA                               # Column in which the data lives.
time-chunk = 100                            # Number of time slots in a chunk.
                                            # Implicitly controls memory footprint.
freq-chunk = 64                             # Number of frequency channels in a chunk.
                                            # Implicitly controls memory footprint.

[sel]
_Help = Data selection options
field = 0                                   # Select a field/fields.
ddid = None                                 # Select spectral window/s.
chan =                                      # Select channel/s.

[out]
_Help = Options for output products
dir = cubical                               # Output directory name.
name = cubical.cc-out/cubical               # Output path for non-visibility outputs.
overwrite = True                            # Overwrite existing output.
mode = sr                                   # Produce DI corrected residuals.
                                            # Corrected data cannot be produced for DD␣
↪gains.
                                            # Check --help for other options.
column = CORRECTED_DATA                     # Output visibility column.

[model]
_Help = Calibration model options
list = skymodel.lsm.html+-skymodel.lsm.html@dE:skymodel.lsm.html@dE
                                            # Input recipe.
                                            # This creates a direction dependent model.
                                            # Directions are separated by colons.
                                            # In this example, direction 0 will be
                                            # the entire sky model minus the contributions
                                            # from the tagged dE sources. The remaining
                                            # directions will be those tagged in The
                                            # sky model. Multiple columns can be specified
                                            # in a similar fashion.

[weight]
_Help = Weighting options
column = None                               # Weight column to use.
```

(continues on next page)

(continued from previous page)

```
[flags]
_Help = General flagging options
apply = -cubical                        # Ignore existing cubical flags.

[sol]
_Help = Solution options which apply at the solver level
jones = G,dE                            # Term/s to solve for.
                                        # Corresponding section/s below.
delta-g = 1e-06                         # Stopping criteria on gains.
delta-chi = 1e-06                       # Stopping criteria on chi-squared.
term-iters = [20, 20, 20, 20]           # Iteration recipe. Loops over jones above.
                                        # This will do 20 iterations on G,
                                        # 20 on dE, 20 on G and finally 20 on dE.
                                        # Overrides individual term settings.

[dist]
_Help = Parallelization and distribution options
ncpu = 4                                # Number of processes to use.
                                        # Implicitly controls memory footprint.
max-chunks = 12                         # Maximum number of simultaneous chunks.
                                        # Ideally >= ncpu.
min-chunks = 12                         # Minimum number of simultaneous chunks.
                                        # Ideally = ncpu.
safe = 1.0                              # Fraction of total memory CubiCal may use.
                                        # Will crash deliberately if exceeded.

[g]
_Help = Options for G-Jones term
label = G                               # This term's name.
                                        # Must match [sol] section's jones option.
type = complex-2x2                      # Solve for a full 2x2 complex gain.
                                        # This can be restricted using update type.
time-int = 1                            # Number of timeslots per solution.
                                        # 0 is the entire chunk axis.
freq-int = 1                            # Number of channels per solution.
max-iter = 20                           # Maximum number of iterations.
                                        # May be ignored if [sol] term-iters is set.
_Templated = True                       # Other parameters polulated from defaults.
                                        # Can be safely ignored.

[de]
_Help = Options for G-Jones term
label = dE                              # This term's name.
                                        # Must match [sol] section's jones option.
type = complex-2x2                      # Solve for a full 2x2 complex gain.
                                        # This can be restricted using update type.
dd-term = 1                             # This term is diretion dependent.
time-int = 20                           # Number of timeslots per solution.
freq-int = 32                           # Number of channels per solution.
                                        # 0 is the entire chunk axis.
max-iter = 20                           # Maximum number of iterations.
                                        # May be ignored if [sol] term-iters is set.
_Templated = True                       # Other parameters polulated from defaults.
                                        # Can be safely ignored.
```

CHAPTER

SIX

PERFORMANCE TUNING

One of CubiCal's main features is high-speed gain calibration. However, its performance is highly depenedent on its input parameters. This is unfortunate, but unavoidable given variety of problems which CubiCal can solve.

A very simple performance consideration for multi-core architectures is the number of processes which CubiCal spawns. This is specified by –dist-ncpu from command line or in the [dist] section of a parset. If this number is greater than one, multiprocessing will be used and one process will dedicated to simulation/IO. Practially, the means that there is always one fewer process performing compute than is specified. This becomes important when tiling the problem, as ideally the number of chunks per tile will be divisible by the number of processes minus one.

The second, and probably most crucial, step in making CubiCal as fast as possible is in the selection of time and frequency chunk sizes. If these chunks are too large, it will lead to a massive increase in cache-misses on the CPU. This will degrade performance quite substantially. There is no hard-and-fast rule for selecting the sizes, but users should be aware of the negative impact of setting them too large, as well as the wasted compute if they are too small. Due to the dependence of this problem on architecture, it may take users a while to get a feel for the optimal. Note that solution intervals can only be as large as a chunk; for large time/frequency solution intervals there is no alternative but to accept the decreased performance.

CHAPTER

SEVEN

LICENCE

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see http://www.gnu.org/licenses/.