IMAGER

A GILDAS software

Nov, 15th, 2022

Version 3.6-00

corresponding to code language versions:

```
Language DISPLAY   1.7-00 08-Nov-2022  S.Guilloteau
Language CLEAN     6.5-07 14-Nov-2022  S.Guilloteau
Language CALIBRATE 1.8-02 17-Jan-2022  S.Guilloteau
Language ADVANCED  2.6-00 08-Nov-2022  S.Guilloteau
Language BUNDLES   1.6-02 29-Sep-2022  S.Guilloteau
Language IMAGER    1.1-00 17-Oct-2022  S.Guilloteau
```

Questions? Comments? Bug reports? Mail to: imager-hotline@services.cnrs.fr

The GILDAS team welcomes an acknowledgment in publications
using GILDAS software to reduce and/or analyze data.
Please use the following reference in your publications:
http://www.iram.fr/IRAMFR/GILDAS
and for IMAGER
https://imager.oasu.u-bordeaux.fr

Documentation In charge: E. Di Folco (1)
Active contributor: S. Guilloteau (1)
Previous contibutors: J. Pety (2,3)

Software In charge: S. Guilloteau (1)
Active contributor: S. Guilloteau (1)
Previous contibutors: S. Bardeau (2), J. Pety (2,3)

1. Laboratoire d'Astrophysique de Bordeaux
2. IRAM
3. Observatoire de Paris

Notations

Throughout this document, colors are used to highlight the names of categories of entities:

- PROGRAM is a GILDAS program name.
- LANGUAGE\ is a SIC Language name
- NAME is a command name, keyword or option
- SICVAR is a SIC variable
- ScriptName is an IMAGER script (external text file) that can be executed by the @ command
- FileName is an external file, e.g. a data file.

Related information on IMAGER is available in:

- IRAM Plateau de Bure Interferometer: Introduction
- IRAM Plateau de Bure Interferometer: OBS Users Guide
- IRAM Plateau de Bure Interferometer: Atmospheric Calibration
- IRAM Plateau de Bure Interferometer: Calibration Cookbook
- CLIC: Continuum and Line Interferometric Calibration
- GREG: Graphical Possibilities
- SIC: Command Line Interpretor

Contents

1 Pre-amble **20**

2 IMAGER in short **21**

3 IMAGER for you... **21**
3.1 IMAGER for beginners . . . 21
3.2 IMAGER for previous users of MAPPING . . . 22
3.2.1 Forget INPUT and GO command . . . 22
3.3 IMAGER for huge data sets: 100 000 channel imaging with NOEMA or ALMA 22
3.4 IMAGER for lazy or overbooked ones . . . 23
3.5 IMAGER for ALMA data . . . 23
3.6 IMAGER for image analysis and publication plots . . . 23

4 IMAGER principles **23**
4.1 Overview of the data reduction and analysis . . . 23
4.2 The structure of the IMAGER program . . . 24
4.2.1 Structure and recommendations . . . 24
4.2.2 Implementation issues . . . 25
4.3 Imaging/deconvolution: a brief sequence of commands . . . 25
4.4 Getting Help: the HELP command and the ? token . . . 26
4.5 IMAGER versus CASA: controlling the number of synthesized beams, BEAM_STEP 28

5 The input data: UV Tables **30**
5.1 In a nutshell . . . 30
5.2 UV table description . . . 31
5.2.1 UV table data format . . . 31
5.2.2 *uv* header . . . 31
5.3 NOEMA: UV Tables from CLIC . . . 33
5.4 ALMA: UV Tables from CASA . . . 33
5.4.1 On the CASA side . . . 34
5.4.2 On the IMAGER side . . . 34
5.4.3 Current assumptions and limitations . . . 34
5.5 Reading UV tables . . . 35
5.6 UV Table handling . . . 35

6 Single-field imaging and deconvolution **37**
6.1 In a nutshell . . . 37
6.2 Measurement equation and other definitions . . . 37
6.3 Imaging . . . 38
6.3.1 Image size and pixel size . . . 38
6.3.2 Weighting and Tapering . . . 39
6.3.3 Implementation (READ UV, UV_MAP and UV_STAT) . . . 40
6.3.4 Defining the Projection Center of the image . . . 42
6.3.5 Typical imaging session . . . 42
6.4 Deconvolution . . . 42
6.5 The family of CLEAN algorithms (HOGBOM, CLARK, MX, SDI, MRC, MULTI) . . . 43
6.5.1 CLEAN ideas . . . 43

6.5.2 Basic CLEAN algorithms (HOGBOM, CLARK and MX) . . . 45
6.5.3 Advanced CLEAN algorithms to deal with extended structures (SDI, MULTI and MRC) . . . 46
6.5.4 Implementation and typical use . . . 47
6.5.5 Restoring step: UV_RESTORE . . . 49
6.5.6 Typical deconvolution session . . . 49
6.6 Practical advices . . . 51
6.6.1 Comparison of deconvolution algorithms . . . 51
6.6.2 A few (obvious) practical recommendations . . . 51

7 Wide-field imaging and deconvolution 53
7.1 In a nutshell . . . 53
7.2 General considerations about wide-field imaging . . . 53
7.3 Mosaicing . . . 55
7.3.1 Observations and processing . . . 55
7.3.2 Imaging . . . 55
7.3.3 Deconvolution . . . 56
7.3.4 Typical use . . . 56
7.4 Mosaic and JvM factor . . . 57

8 Short and Zero spacings 58
8.1 In a nutshell . . . 58
8.2 Principle . . . 58
8.3 Algorithms to merge single-dish and interferometer information . . . 58
8.3.1 Hybridization technique ("feathering") . . . 59
8.3.2 Pseudo-visibility technique . . . 60
8.3.3 Comparison . . . 61
8.4 Hybridization technique and ALMA . . . 62
8.5 The Zero spacing: an important subset . . . 62
8.6 Short Spacings in practice: command UV_SHORT . . . 62
8.7 Practical considerations . . . 63
8.7.1 When are short-spacing information needed? . . . 63
8.7.2 How to optimize single-dish observations? . . . 64

9 Self Calibration 65
9.1 Self-Calibration in a nutshell . . . 65
9.2 Self-Calibration Principle . . . 65
9.3 Self-Calibration Implementation . . . 67
9.4 Basic use . . . 69
9.4.1 Timescale for averaging solution interval . . . 69
9.4.2 Quality assessment and data flagging . . . 69
9.5 Advanced use . . . 70
9.6 Transfering the solution to other *uv* data sets. . . . 71
9.7 Data re-weighting . . . 71
9.8 Data flagging . . . 71

10 UV plane analysis 73

11 Continuum emission 75
11.1 Continuum imaging . . . 75
11.2 Split Continuum and Spectral line imaging . . . 76
11.3 Combined Continuum and Spectral line imaging . . . 76
11.4 The CONTINUUM image . . . 77

12 Visual Checks and Image Displays 78
12.1 The VIEWER program . . . 78
12.1.1 The SHOW command . . . 78
12.1.2 the VIEW command . . . 83
12.1.3 the INSPECT_3D command . . . 85
12.1.4 the EXPLORE command . . . 86
12.2 Specific IMAGER visualisation tools . . . 87
12.2.1 The UV_PREVIEW command . . . 87
12.2.2 the SELFCAL SHOW command . . . 89

13 The Imager PIPELINE tools 91
13.1 Preparing the data . . . 92
13.2 The PIPELINE Widget . . . 92
13.2.1 Getting started: PIPELINE /WIDGET . . . 92
13.2.2 ORGANIZE step . . . 92
13.2.3 FIND step . . . 92
13.2.4 SELECT step . . . 92
13.2.5 CHECK step . . . 93
13.2.6 SELF step . . . 93
13.2.7 SHOW step . . . 94
13.2.8 COLLECT step . . . 94
13.2.9 APPLY step . . . 94
13.2.10 TIME step . . . 94
13.2.11 IMAGING step . . . 94
13.2.12 TABLES step . . . 95
13.2.13 SKY step . . . 96
13.3 The non interactive PIPELINE . . . 96
13.4 Mosaics and other limitations. . . . 96

14 Really Huge Problems 96

15 Polarization Handling 98
15.1 Data Handling . . . 98
15.2 The STOKES command . . . 99

16 Advanced Data Analysis 101
16.1 Matched Filtering . . . 101
16.2 Line Stacking . . . 101
16.3 Rotationally symmetric (thin) structures: Keplerian disks . . . 101
16.4 Image Handling and Comparison . . . 101

17 DISPLAY Language Internal Help 102
17.1 Language . . . 102
17.2 CATALOG . . . 102
17.2.1 CATALOG Default . . . 103
17.2.2 CATALOG Astro . . . 103
17.2.3 CATALOG Linedb . . . 103
17.2.4 CATALOG LINEDB%ENERGY . . . 104
17.3 COLOR . . . 104
17.4 EXPLORE . . . 104
17.4.1 EXPLORE /ADD . . . 105
17.4.2 EXPLORE /NOPAUSE . . . 105
17.4.3 EXPLORE Variables: . . . 105
17.4.4 CENTER . . . 105
17.4.5 DO_BIT . . . 106
17.4.6 DO_CONTOUR . . . 106
17.4.7 DO_GREY . . . 106
17.4.8 DO_MASK . . . 106
17.4.9 RANGE . . . 106
17.4.10 SIZE . . . 107
17.4.11 SCALE . . . 107
17.4.12 SPACE_TYPE . . . 107
17.4.13 SPACING . . . 107
17.4.14 SPAN . . . 108
17.5 EXTRACT . . . 108
17.6 FIND . . . 108
17.6.1 FIND /SPECIES . . . 109
17.7 INSPECT_3D . . . 109
17.7.1 INSPECT_3D History . . . 109
17.7.2 INSPECT_3D Variables: . . . 110
17.7.3 SPAN . . . 110
17.8 LOAD . . . 110
17.8.1 LOAD /FREQUENCY . . . 110
17.8.2 LOAD /PLANES . . . 110
17.8.3 LOAD /RANGE . . . 111
17.9 POPUP . . . 111
17.10 SET . . . 111
17.10.1 SET ANGLE_UNIT . . . 112
17.10.2 SET OtherKeyword . . . 112
17.11 SHOW . . . 112
17.11.1 SHOW /SIDE . . . 113
17.11.2 SHOW History . . . 113
17.11.3 SHOW Keywords: . . . 113
17.11.4 BEAM . . . 114
17.11.5 CCT . . . 114
17.11.6 COMPOSITE . . . 114
17.11.7 CONTINUUM . . . 115
17.11.8 COVERAGE . . . 115

17.11.9 FIELDS . . . 115
17.11.10 FLUX . . . 115
17.11.11 KEPLER . . . 116
17.11.12 MOMENTS . . . 116
17.11.13 NOISE . . . 116
17.11.14 PRIMARY . . . 116
17.11.15 PV . . . 117
17.11.16 SED . . . 117
17.11.17 SELFCAL . . . 117
17.11.18 SNR . . . 118
17.11.19 SOURCES . . . 118
17.11.20 SPECTRA . . . 118
17.11.21 UV_DATA . . . 118
17.11.22 UV_FIT . . . 119
17.11.23 Variables: . . . 119
17.11.24 BOX_LIMITS . . . 120
17.11.25 DO_BIT . . . 120
17.11.26 DO_CONTOUR . . . 120
17.11.27 DO_COVERAGE . . . 120
17.11.28 DO_DIRTY . . . 120
17.11.29 DO_FIELDS . . . 121
17.11.30 DO_GREY . . . 121
17.11.31 DO_HEADER . . . 121
17.11.32 DO_LABEL . . . 121
17.11.33 DO_MASK . . . 121
17.11.34 DO_NICE . . . 121
17.11.35 DO_WEDGE . . . 122
17.11.36 CENTER . . . 122
17.11.37 CROSS . . . 122
17.11.38 MARK . . . 122
17.11.39 RANGE . . . 122
17.11.40 SCALE . . . 122
17.11.41 SIZE . . . 123
17.11.42 SPACE_TYPE . . . 123
17.11.43 SPACING . . . 123
17.11.44 SPAN . . . 124
17.11.45 SHOW_SIDE . . . 124
17.12 SLICE . . . 124
17.13 SPECTRUM . . . 124
17.13.1 SPECTRUM /CORNER . . . 125
17.13.2 SPECTRUM /MEAN . . . 125
17.13.3 SPECTRUM /PLANE . . . 125
17.13.4 SPECTRUM /SUM . . . 125
17.14 STATISTIC . . . 126
17.14.1 STATISTIC /NOISE . . . 126
17.14.2 STATISTIC /EDGE . . . 126
17.15 VIEW . . . 127

17.15.1 VIEW /NOPAUSE . . . 127
17.15.2 VIEW /OVERLAY . . . 128
17.15.3 VIEW Keys . . . 128
17.15.4 VIEW Scripts . . . 129
17.15.5 VIEW Variables . . . 129
17.15.6 VIEW Keywords: . . . 129
17.15.7 BEAM . . . 129
17.15.8 PRIMARY . . . 130

18 CLEAN Language Internal Help 130
18.1 Language . . . 130
18.2 ALMA . . . 131
18.3 BUFFERS . . . 131
18.3.1 BUFFERS AGAINS . . . 132
18.3.2 BUFFERS BEAM . . . 132
18.3.3 BUFFERS CCT . . . 132
18.3.4 BUFFERS CGAINS . . . 132
18.3.5 BUFFERS CLEAN . . . 132
18.3.6 BUFFERS CLIPPED . . . 132
18.3.7 BUFFERS CONTINUUM . . . 133
18.3.8 BUFFERS DIRTY . . . 133
18.3.9 BUFFERS EXTRACTED . . . 133
18.3.10 BUFFERS FIELDS . . . 133
18.3.11 BUFFERS MASK . . . 133
18.3.12 BUFFERS M_AREA . . . 133
18.3.13 BUFFERS M_PEAK . . . 133
18.3.14 BUFFERS M_VELO . . . 134
18.3.15 BUFFERS M_WIDTH . . . 134
18.3.16 BUFFERS PRIMARY . . . 134
18.3.17 BUFFERS RESIDUAL . . . 134
18.3.18 BUFFERS SHORT . . . 134
18.3.19 BUFFERS SINGLE . . . 134
18.3.20 BUFFERS SKY . . . 134
18.3.21 BUFFERS SLICED . . . 135
18.3.22 BUFFERS SPECTRUM . . . 135
18.3.23 BUFFERS UV . . . 135
18.3.24 BUFFERS UV_FIT . . . 135
18.3.25 BUFFERS UVCONT . . . 135
18.3.26 BUFFERS UVRADIAL . . . 135
18.3.27 BUFFERS WEIGHT . . . 135
18.4 CCT_CLEAN . . . 135
18.5 CCT_CONVERT . . . 136
18.6 CCT_MERGE . . . 136
18.7 CLEAN . . . 136
18.7.1 CLEAN /FLUX . . . 137
18.7.2 CLEAN /PLOT . . . 137
18.7.3 CLEAN /QUERY . . . 137
18.7.4 CLEAN /NITER . . . 138

18.7.5 CLEAN /ARES . . . 138
18.7.6 CLEAN /RESTART . . . 138
18.7.7 CLEAN Methods: . . . 139
18.7.8 CLARK . . . 139
18.7.9 HOGBOM . . . 139
18.7.10 MRC . . . 139
18.7.11 MULTI . . . 140
18.7.12 SDI . . . 140
18.7.13 Variables: . . . 141
18.7.14 BEAM_SIZE . . . 141
18.7.15 CLEAN_ARES . . . 142
18.7.16 CLEAN_FRES . . . 142
18.7.17 CLEAN_GAIN . . . 143
18.7.18 CLEAN_INFLATE . . . 143
18.7.19 CLEAN_NCYCLE . . . 143
18.7.20 CLEAN_NGOAL . . . 143
18.7.21 CLEAN_NITER . . . 143
18.7.22 CLEAN_NKEEP . . . 144
18.7.23 CLEAN_POSITIVE . . . 144
18.7.24 CLEAN_RESIDUAL . . . 144
18.7.25 CLEAN_SEARCH . . . 145
18.7.26 CLEAN_SIDELOBE . . . 145
18.7.27 CLEAN_SMOOTH . . . 145
18.7.28 CLEAN_SPEEDY . . . 145
18.7.29 CLEAN_STOP . . . 145
18.7.30 CLEAN_WORRY . . . 146
18.7.31 METHOD . . . 146
18.7.32 Old_Names: . . . 147
18.7.33 BLC . . . 147
18.7.34 TRC . . . 147
18.7.35 MAJOR . . . 147
18.7.36 MINOR . . . 147
18.7.37 ANGLE . . . 147
18.7.38 BEAM_PATCH . . . 148
18.8 DISCARD . . . 148
18.9 DUMP . . . 148
18.10 FIT . . . 148
18.10.1 FIT FixedValues . . . 149
18.10.2 FIT Results . . . 149
18.10.3 FIT /JVM_FACTOR . . . 149
18.10.4 FIT /THRESHOLD . . . 150
18.10.5 FIT BEAM_SIZE . . . 150
18.10.6 FIT CLEAN_SIDELOBE . . . 151
18.11 MAP_COMBINE . . . 151
18.11.1 MAP_COMBINE CODE . . . 152
18.11.2 MAP_COMBINE /BLANKING . . . 152
18.11.3 MAP_COMBINE /FACTOR . . . 152

18.11.4 MAP_COMBINE /THRESHOLD . . . 152
18.12 MAP_COMPRESS . . . 153
18.12.1 MAP_COMPRESS Output . . . 153
18.13 MAP_INTEGRATE . . . 153
18.13.1 MAP_INTEGRATE Output . . . 154
18.14 MAP_REPROJECT . . . 154
18.14.1 MAP_REPROJECT /BLANKING . . . 155
18.14.2 MAP_REPROJECT /LIKE . . . 155
18.14.3 MAP_REPROJECT /PROJECTION . . . 155
18.14.4 MAP_REPROJECT /SYSTEM . . . 156
18.14.5 MAP_REPROJECT /X_AXIS . . . 156
18.14.6 MAP_REPROJECT /Y_AXIS . . . 156
18.15 MAP_RESAMPLE . . . 157
18.15.1 MAP_RESAMPLE /LIKE . . . 157
18.16 MAP_SMOOTH . . . 157
18.16.1 MAP_SMOOTH /ASYMMETRIC . . . 158
18.17 SPECIFY . . . 158
18.17.1 SPECIFY BLANKING . . . 158
18.17.2 SPECIFY FREQUENCY . . . 158
18.17.3 SPECIFY LINENAME . . . 159
18.17.4 SPECIFY VELOCITY . . . 159
18.17.5 SPECIFY TELESCOPE . . . 159
18.18 MOSAIC . . . 159
18.19 MX . . . 159
18.19.1 MX Variables: . . . 160
18.20 PRIMARY . . . 161
18.20.1 PRIMARY /TRUNCATE . . . 162
18.21 READ . . . 162
18.21.1 READ Buffers . . . 163
18.21.2 READ FITS . . . 163
18.21.3 READ Optimisation . . . 164
18.21.4 READ /COMPACT . . . 164
18.21.5 READ /FREQUENCY . . . 164
18.21.6 READ /NOTRAIL . . . 164
18.21.7 READ /PLANES . . . 164
18.21.8 READ /RANGE . . . 165
18.21.9 READ SINGLE . . . 165
18.22 SUPPORT . . . 165
18.22.1 SUPPORT /CURSOR . . . 166
18.22.2 SUPPORT /MASK . . . 166
18.22.3 SUPPORT /PLOT . . . 167
18.22.4 SUPPORT /RESET . . . 167
18.22.5 SUPPORT /THRESHOLD . . . 167
18.22.6 SUPPORT /VARIABLE . . . 167
18.23 UV_BASELINE . . . 167
18.23.1 UV_BASELINE /CHANNELS . . . 168
18.23.2 UV_BASELINE /FILE . . . 168

18.23.3 UV_BASELINE /FREQUENCY . . . 168
18.23.4 UV_BASELINE /RANGE . . . 169
18.23.5 UV_BASELINE /VELOCITY . . . 169
18.23.6 UV_BASELINE /WIDTH . . . 169
18.24 UV_CHECK . . . 169
18.24.1 UV_CHECK /FILE . . . 170
18.24.2 UV_CHECK BEAM_RANGES . . . 170
18.25 UV_COMPRESS . . . 170
18.25.1 UV_COMPRESS /CONTINUUM . . . 171
18.25.2 UV_COMPRESS /FILE . . . 171
18.26 UV_CONTINUUM . . . 171
18.26.1 UV_CONTINUUM /INDEX . . . 172
18.26.2 UV_CONTINUUM /RANGE . . . 172
18.27 UV_EXTRACT . . . 172
18.27.1 UV_EXTRACT /FILE . . . 173
18.27.2 UV_EXTRACT /RANGE . . . 173
18.28 UV_FIELDS . . . 173
18.28.1 UV_FIELDS /CENTER . . . 174
18.28.2 UV_FIELDS /FILE . . . 174
18.29 UV_FILTER . . . 175
18.29.1 UV_FILTER /CHANNELS . . . 175
18.29.2 UV_FILTER /FILE . . . 175
18.29.3 UV_FILTER /FREQUENCY . . . 176
18.29.4 UV_FILTER /RANGE . . . 176
18.29.5 UV_FILTER /RESET . . . 176
18.29.6 UV_FILTER /VELOCITY . . . 176
18.29.7 UV_FILTER /WIDTH . . . 177
18.29.8 UV_FILTER /ZERO . . . 177
18.30 UV_FLAG . . . 177
18.30.1 UV_FLAG /ANTENNA . . . 178
18.30.2 UV_FLAG /DATE . . . 178
18.30.3 UV_FLAG /FILE . . . 179
18.30.4 UV_FLAG /RESET . . . 179
18.30.5 UV_FLAG CHANNEL . . . 179
18.31 UV_MAP . . . 179
18.31.1 UV_MAP Mosaics . . . 180
18.31.2 UV_MAP /CONT . . . 180
18.31.3 UV_MAP /FIELDS . . . 181
18.31.4 UV_MAP /INDEX . . . 181
18.31.5 UV_MAP /RANGE . . . 181
18.31.6 UV_MAP /TRUNCATE . . . 182
18.31.7 UV_MAP Variables: . . . 182
18.31.8 BEAM_STEP . . . 182
18.31.9 MAP_CELL . . . 183
18.31.10 MAP_CENTER . . . 183
18.31.11 MAP_CONVOLUTION . . . 183
18.31.12 MAP_FIELD . . . 184

18.31.13 MAP_POWER . . . 184
18.31.14 MAP_PRECIS . . . 184
18.31.15 MAP_ROBUST . . . 184
18.31.16 MAP_ROUNDING . . . 185
18.31.17 MAP_SIZE . . . 185
18.31.18 MAP_TAPEREXPO . . . 186
18.31.19 MAP_TRUNCATE . . . 186
18.31.20 MAP_UVTAPER . . . 186
18.31.21 MAP_UVCELL . . . 186
18.31.22 MAP_VERSION . . . 186
18.31.23 Old_Names: . . . 187
18.31.24 convolution . . . 187
18.31.25 map_angle . . . 188
18.31.26 map_beam_step . . . 188
18.31.27 map_dec . . . 188
18.31.28 map_ra . . . 188
18.31.29 mcol . . . 188
18.31.30 uv_cell . . . 189
18.31.31 uv_shift . . . 189
18.31.32 uv_taper . . . 189
18.31.33 taper_expo . . . 189
18.31.34 wcol . . . 189
18.31.35 weight_mode . . . 189
18.32 UV_RESAMPLE . . . 190
18.32.1 UV_RESAMPLE /FILE . . . 190
18.32.2 UV_RESAMPLE /LIKE . . . 190
18.33 UV_RESIDUAL . . . 191
18.33.1 UV_RESIDUAL Note . . . 191
18.34 UV_RESTORE . . . 191
18.34.1 UV_RESTORE /COPY . . . 192
18.34.2 UV_RESTORE /SPEED . . . 192
18.35 UV_REWEIGHT . . . 192
18.35.1 UV_REWEIGHT APPLY . . . 193
18.35.2 UV_REWEIGHT DO . . . 193
18.35.3 UV_REWEIGHT ESTIMATE . . . 193
18.35.4 UV_REWEIGHT TIME . . . 194
18.35.5 UV_REWEIGHT /RANGE . . . 194
18.35.6 UV_REWEIGHT /FILE . . . 194
18.35.7 UV_REWEIGHT /FLAG . . . 194
18.36 UV_SHIFT . . . 195
18.36.1 UV_SHIFT /FILE . . . 195
18.37 UV_SMOOTH . . . 195
18.37.1 UV_SMOOTH /ASYMMETRIC . . . 196
18.37.2 UV_SMOOTH /FILE . . . 196
18.38 UV_SPLIT . . . 196
18.38.1 UV_SPLIT /CHANNELS . . . 197
18.38.2 UV_SPLIT /FILE . . . 197

18.38.3 UV_SPLIT /FREQUENCY . . . 198
18.38.4 UV_SPLIT /RANGE . . . 198
18.38.5 UV_SPLIT /VELOCITY . . . 198
18.38.6 UV_SPLIT /WIDTH . . . 199
18.39 UV_STAT . . . 199
18.39.1 UV_STAT Casa . . . 199
18.39.2 UV_STAT Mosaics . . . 200
18.39.3 UV_STAT Arguments: . . . 200
18.39.4 ADVISE . . . 200
18.39.5 ALL . . . 201
18.39.6 BEAM . . . 201
18.39.7 BRIGGS . . . 201
18.39.8 CELL . . . 201
18.39.9 DEFAULT . . . 202
18.39.10 HEADER . . . 202
18.39.11 RESET . . . 202
18.39.12 SETUP . . . 202
18.39.13 TAPER . . . 202
18.39.14 WEIGHT . . . 202
18.40 UV_TIME . . . 203
18.40.1 UV_TIME /WEIGHT . . . 203
18.40.2 UV_TIME /FILE . . . 203
18.41 UV_TRIM . . . 203
18.41.1 UV_TRIM /FILE . . . 204
18.42 UV_TRUNCATE . . . 204
18.43 WRITE . . . 204
18.43.1 WRITE Format . . . 205
18.43.2 WRITE /RANGE . . . 205
18.43.3 WRITE /APPEND . . . 205
18.43.4 WRITE /REPLACE . . . 205
18.43.5 WRITE /TRIM . . . 206

19 CALIBRATE Language Internal Help 207
19.1 Language . . . 207
19.2 APPLY . . . 207
19.2.1 APPLY AMPLI . . . 207
19.2.2 APPLY DELAY . . . 208
19.2.3 APPLY PHASE . . . 208
19.2.4 APPLY /FLAG . . . 208
19.3 COLLECT . . . 208
19.3.1 COLLECT /FLAG . . . 209
19.4 DERIVE . . . 209
19.4.1 DERIVE Example . . . 209
19.5 SCALE_FLUX . . . 210
19.6 MODEL . . . 211
19.6.1 MODEL /MINVAL . . . 211
19.6.2 MODEL /MODE . . . 212
19.7 SOLVE . . . 212

19.7.1 SOLVE /MODE . . . 212
19.8 TRANSFORM . . . 213
19.8.1 TRANSFORM FFT . . . 213
19.8.2 TRANSFORM WAVE . . . 214
19.9 UV_SELF . . . 214
19.9.1 UV_SELF /RANGE . . . 214
19.9.2 UV_SELF /RESTORE . . . 215
19.10 UV_SELECT . . . 215
19.11 UV_SORT . . . 215
19.11.1 UV_SORT /FILE . . . 216

20 ADVANCED Language Internal Help 217
20.1 Language . . . 217
20.2 FEATHER . . . 217
20.2.1 FEATHER /REPROJECT . . . 218
20.2.2 FEATHER /FILE . . . 218
20.2.3 FEATHER Algorithm . . . 218
20.2.4 FEATHER Variables: . . . 219
20.2.5 FEATHER_RADIUS . . . 219
20.2.6 FEATHER_EXPO . . . 219
20.2.7 FEATHER_RATIO . . . 219
20.2.8 FEATHER_RANGE . . . 219
20.3 FLUX . . . 220
20.3.1 FLUX Limitations . . . 220
20.3.2 FLUX Results . . . 220
20.4 HOW_TO . . . 220
20.5 MAP_CONTINUUM . . . 221
20.5.1 MAP_CONTINUUM /METHOD . . . 221
20.5.2 MAP_CONTINUUM GLOBAL . . . 221
20.5.3 MAP_CONTINUUM GAUSS . . . 222
20.5.4 MAP_CONTINUUM SCM . . . 222
20.5.5 MAP_CONTINUUM C-SCM . . . 223
20.5.6 MAP_CONTINUUM EGM . . . 223
20.6 MAP_POLAR . . . 223
20.6.1 MAP_POLAR /COMPUTE . . . 224
20.6.2 MAP_POLAR /BACKGROUND . . . 224
20.6.3 MAP_POLAR /STEP . . . 224
20.7 MASK . . . 225
20.7.1 MASK Tricks . . . 225
20.7.2 MASK ADD . . . 226
20.7.3 MASK APPLY . . . 226
20.7.4 MASK CHECK . . . 226
20.7.5 MASK INITIALIZE . . . 226
20.7.6 MASK INTERACTIVE . . . 226
20.7.7 MASK OVERLAY . . . 227
20.7.8 MASK READ . . . 227
20.7.9 MASK REMOVE . . . 227
20.7.10 MASK SHOW . . . 227

20.7.11 MASK THRESHOLD . . . 228
20.7.12 MASK USE . . . 229
20.7.13 MASK WRITE . . . 229
20.8 MFS . . . 229
20.9 MOMENTS . . . 229
20.9.1 MOMENTS /CUTS . . . 230
20.9.2 MOMENTS /MASK . . . 230
20.9.3 MOMENTS /METHOD . . . 230
20.9.4 MOMENTS /RANGE . . . 230
20.9.5 MOMENTS /THRESHOLD . . . 231
20.10 PROPER_MOTION . . . 231
20.10.1 PROPER_MOTION /FILE . . . 231
20.11 STOKES . . . 231
20.11.1 STOKES /FILE . . . 232
20.12 UV_ADD . . . 232
20.12.1 UV_ADD /FILE . . . 232
20.13 UV_CIRCLE . . . 232
20.13.1 UV_CIRCLE /SAMPLING . . . 233
20.13.2 UV_CIRCLE /ZERO . . . 233
20.14 UV_CORRELATE . . . 233
20.14.1 UV_CORRELATE /FILE . . . 234
20.15 UV_DEPROJECT . . . 234
20.16 UV_FIT . . . 234
20.16.1 UV_FIT /QUIET . . . 235
20.16.2 UV_FIT /SAVE . . . 235
20.16.3 UV_FIT /WIDGET . . . 236
20.16.4 UV_FIT History . . . 236
20.16.5 UV_FIT ResultTable . . . 236
20.16.6 UV_FIT ResultValues . . . 237
20.17 UV_MERGE . . . 237
20.17.1 UV_MERGE /FILES . . . 237
20.17.2 UV_MERGE /MODE . . . 238
20.17.3 UV_MERGE /SCALES . . . 238
20.17.4 UV_MERGE /WEIGHTS . . . 239
20.18 UV_MOSAIC . . . 239
20.18.1 UV_MOSAIC MERGE . . . 239
20.18.2 UV_MOSAIC SPLIT . . . 240
20.19 UV_PREVIEW . . . 240
20.19.1 UV_PREVIEW /BROWSE . . . 240
20.19.2 UV_PREVIEW /FILE . . . 241
20.19.3 UV_PREVIEW Algorithm . . . 241
20.19.4 UV_PREVIEW Limitations . . . 242
20.19.5 UV_PREVIEW Output . . . 242
20.20 UV_RADIAL . . . 243
20.20.1 UV_RADIAL /SAMPLING . . . 243
20.20.2 UV_RADIAL /ZERO . . . 243
20.21 UV_SHORT . . . 244

20.21.1 UV_SHORT /CHECK . . . 244
20.21.2 UV_SHORT /REMOVE . . . 244
20.21.3 UV_SHORT Algorithm . . . 244
20.21.4 UV_SHORT Zero_Spacing . . . 245
20.21.5 UV_SHORT Step_1 . . . 245
20.21.6 UV_SHORT Step_2 . . . 246
20.21.7 UV_SHORT Variables: . . . 247
20.21.8 SHORT_DO_SINGLE . . . 247
20.21.9 SHORT_DO_PRIMARY . . . 247
20.21.10 SHORT_FTOLE . . . 247
20.21.11 SHORT_IP_BEAM . . . 247
20.21.12 SHORT_IP_DIAM . . . 248
20.21.13 SHORT_MCOL . . . 248
20.21.14 SHORT_MIN_WEIGHT . . . 248
20.21.15 SHORT_MODE . . . 248
20.21.16 SHORT_SD_BEAM . . . 249
20.21.17 SHORT_SD_DIAM . . . 249
20.21.18 SHORT_SD_FACTOR . . . 249
20.21.19 SHORT_SD_WEIGHT . . . 250
20.21.20 SHORT_TOLE . . . 250
20.21.21 SHORT_UV_MAX . . . 250
20.21.22 SHORT_UV_MIN . . . 251
20.21.23 SHORT_WCOL . . . 251
20.21.24 SHORT_WEIGHT_MODE . . . 251
20.21.25 SHORT_XCOL . . . 251
20.21.26 SHORT_YCOL . . . 251

21 BUNDLES Language Internal Help 253
21.1 Language . . . 253
21.2 COMBINE . . . 253
21.2.1 COMBINE CODE . . . 253
21.2.2 COMBINE /BLANKING . . . 254
21.2.3 COMBINE /FACTOR . . . 254
21.2.4 COMBINE /THRESHOLD . . . 254
21.3 KEPLER . . . 254
21.3.1 KEPLER INIT . . . 255
21.3.2 KEPLER SHOW . . . 255
21.3.3 KEPLER /HFS . . . 255
21.3.4 KEPLER /MASK . . . 255
21.3.5 KEPLER /RESET . . . 256
21.3.6 KEPLER /VELOCITY . . . 256
21.3.7 KEPLER /VSYSTEM . . . 256
21.3.8 KEPLER /WIDGET . . . 257
21.3.9 KEPLER Variables: . . . 257
21.3.10 KEPLER_X0 . . . 257
21.3.11 KEPLER_Y0 . . . 257
21.3.12 KEPLER_ROTA . . . 257
21.3.13 KEPLER_INCLI . . . 257

21.3.14 KEPLER_DIST . . . 257
21.3.15 KEPLER_VMASS . . . 257
21.3.16 KEPLER_RMIN . . . 257
21.3.17 KEPLER_RMAX . . . 258
21.3.18 KEPLER_RINT . . . 258
21.3.19 KEPLER_ROUT . . . 258
21.3.20 KEPLER_STEP . . . 258
21.3.21 KEPLER_THETA . . . 258
21.3.22 KEPLER_AZIMUT . . . 258
21.3.23 KEPLER_VDISK . . . 259
21.3.24 KEPLER_STRICT . . . 259
21.3.25 Results: . . . 259
21.3.26 KEPLER_PROFILE . . . 259
21.3.27 KEPLER_PV . . . 260
21.3.28 KEPLER_SPECTRUM . . . 260
21.3.29 KEPLER_VELO . . . 260
21.3.30 Display: . . . 260
21.3.31 KEPLER_SHOW . . . 260
21.3.32 KEPLER_SHOW%V . . . 261
21.3.33 KEPLER_SHOW%R . . . 261
21.3.34 KEPLER_SHOW%T . . . 261
21.3.35 KEPLER_SHOW%F . . . 261
21.3.36 KEPLER_SHOW%LAYOUT . . . 261
21.4 SELFCAL . . . 261
21.4.1 SELFCAL /WIDGET . . . 261
21.4.2 SELFCAL Arguments: . . . 262
21.4.3 AMPLITUDE . . . 262
21.4.4 APPLY . . . 262
21.4.5 FLAG . . . 263
21.4.6 INPUT . . . 263
21.4.7 PHASE . . . 263
21.4.8 SAVE . . . 263
21.4.9 SHOW . . . 263
21.4.10 SUMMARY . . . 264
21.4.11 Results: . . . 264
21.4.12 SELF_APPLIED . . . 264
21.4.13 SELF_STATUS . . . 265
21.4.14 SELF_DYNAMIC . . . 265
21.4.15 SELF_RMSCLEAN . . . 265
21.4.16 Variables: . . . 265
21.4.17 SELF_CHANNEL . . . 265
21.4.18 SELF_COLOR . . . 266
21.4.19 SELF_LOOP . . . 266
21.4.20 SELF_NITER . . . 266
21.4.21 SELF_TIMES . . . 267
21.4.22 SELF_MINFLUX . . . 267
21.4.23 SELF_REFANT . . . 268

21.4.24 SELF_FLUX . . . 268
21.4.25 SELF_PRECISION . . . 268
21.4.26 SELF_RESTORE . . . 268
21.4.27 SELF_DISPLAY . . . 268
21.4.28 SELF_FLAG . . . 269
21.4.29 SELF_SNR . . . 269
21.4.30 SELF_SNOISE . . . 269
21.4.31 CLEAN_ARES . . . 269
21.4.32 CLEAN_FRES . . . 269
21.4.33 CLEAN_NITER . . . 269
21.5 SPECTRAL_CLEAN . . . 269
21.5.1 SPECTRAL_CLEAN DUAL . . . 270
21.5.2 SPECTRAL_CLEAN FFT . . . 270
21.5.3 SPECTRAL_CLEAN MULTI . . . 271
21.5.4 SPECTRAL_CLEAN WAVE . . . 271
21.6 UV_DETECT . . . 271
21.6.1 UV_DETECT /FILE . . . 272
21.6.2 UV_DETECT Algorithm . . . 272

22 IMAGER Language Internal Help 273
22.1 Language . . . 273
22.2 TIPS . . . 273
22.3 PIPELINE . . . 273
22.3.1 PIPELINE /MODE . . . 273
22.3.2 PIPELINE /WIDGET . . . 274
22.3.3 PIPELINE Arguments: . . . 274
22.3.4 ORGANIZE . . . 275
22.3.5 FIND . . . 275
22.3.6 SELECT . . . 275
22.3.7 CHECK . . . 275
22.3.8 COMPUTE . . . 276
22.3.9 COLLECT . . . 276
22.3.10 APPLY . . . 276
22.3.11 TIME . . . 276
22.3.12 TABLES . . . 276
22.3.13 IMAGE . . . 277
22.3.14 SHOW . . . 277
22.3.15 SAVE . . . 277
22.3.16 SKY . . . 278
22.3.17 NEXT . . . 278
22.3.18 LAST . . . 278
22.3.19 Variables . . . 278
22.3.20 ALL%CATALOG . . . 278
22.3.21 ALL%DROP . . . 279
22.3.22 ALL%FILTER . . . 279
22.3.23 ALL%AMPLI_NITER . . . 279
22.3.24 ALL%AMPLI_TIMES . . . 279
22.3.25 ALL%COLLECT . . . 280

22.3.26 ALL%COMBINE . . . 280
22.3.27 ALL%ITIME . . . 280
22.3.28 ALL%MINFRES . . . 280
22.3.29 ALL%PHASE_NITER . . . 281
22.3.30 ALL%PHASETIMES . . . 281
22.3.31 ALL%RANGE . . . 281
22.3.32 ALL%SPLIT . . . 281
22.4 HOGBOM . . . 282
22.5 CLARK . . . 282
22.6 MRC . . . 282
22.7 MULTISCALE . . . 282
22.8 SDI . . . 282

23 Bugs and Release Notes 283
23.1 Recent changes in repository . . . 283
23.2 Known Bugs . . . 283
23.3 Functional changes from previous versions . . . 284

A IMAGER versus CASA 286
A.1 Imaging Philosophy and Data Architecture . . . 286
A.2 Frequency and Velocity scales . . . 286
A.3 Transfering UV data from CASA . . . 286
A.4 In CASA . . . 286
A.5 In IMAGER . . . 287
A.6 Transfering Image data . . . 288

B Properties of the Fourier Transform 288

C Revision History 289
C.1 DISPLAY Language . . . 289
C.2 CLEAN Language . . . 289
C.3 ADVANCED Language . . . 289
C.4 CALIBRATE Language . . . 289
C.5 BUNDLES Language . . . 289
C.6 IMAGER Language . . . 289

1 Pre-amble

IMAGER is an interferometric imaging package, tailored for usage simplicity and efficiency for multi-spectral data sets.

IMAGER **is *** NOT ***** MAPPING

IMAGER was created because the initial infrastructure for imaging in the MAPPING program was not adapted to the implementation of imaging methods that became possible and/or were required for NOEMA (and useable for ALMA), such as self-calibration, wide band imaging, or routine processing of Mosaics and Short spacings.

The IMAGER design takes great care of efficiency, by using parallel programming and minimizing Input/Output on files. This lead to a concept with a single `READ` of data, a few (in general only 2) simple processing commands with built-in intelligent parameter guesses, a visual user control, and a single `WRITE` of the results once the user is satisfied of it.

We took the opportunity to revise and streamline the user interface, providing access to all tools through simple commands. A special effort was put on the visualization tools, which provide enhanced speed and capabilities, yet preserving a very high level of compatibility with those previously offered in MAPPING.

IMAGER was derived from MAPPING by S. Guilloteau. T. Jacq and E. di Folco made numerous suggestions about the user interface, and helped testing and documenting the program.

2 IMAGER in short

IMAGER is an interferometric imaging package in the GILDAS software, tailored for usage simplicity and efficiency for multi-spectral data sets.

The main goals of IMAGER are

1. to offer a proper implementation of imaging in case of wide relative bandwidth, where the natural angular resolution changes with frequency.
2. to implement a simple and efficient scheme to process Mosaics, including short spacings from single dish data
3. to take advantage of improved capabilities of NOEMA and ALMA, by offering new tools like self-calibration or wide bandwidth analysis.
4. to simplify user interfaces, by providing sensible defaults.
5. to minimize image sizes
6. to minimize processing time by using parallel programming as much as possible and reducing Input/Output to the strict minimum.

IMAGER was developed and optimized to minimize Input/Output that are the bottleneck of current computers. Therefore, IMAGER works mostly on internal buffers and avoids as much as possible saving data to intermediate files. File saving is done ultimately once the data analysis process is complete, which offers an optimum use of the disk bandwidth.

IMAGER also includes advanced display and image analysis tools, such as simple overlaying of different data cubes, etc...

3 IMAGER for you...

If you do not know how to do something, just use the HOW_TO command to express your question in a (semi-) natural way, e.g.

```
how to combine alma and aca
```

Use the HELP command for more details on commands. Use the TIP command for useful suggestions.

3.1 IMAGER for beginners

If you have never done any interferometry before, or if you do not know what is a *uv* table, we (**strongly**) suggest you read Sections 4.2, 5 and 6 before doing anything at all. These sections are short enough to give you the basic tools (HELP plus the 5 main commands: READ, UV_MAP, CLEAN, VIEW and WRITE), and will give you hints on why IMAGER behaves like this.

Remember two things: look at the results (command VIEW) at all steps, and ask more experienced users if these results make sense if any doubt arise.

3.2 IMAGER for previous users of MAPPING

IMAGER offers a number of advantages against MAPPING : higher speed, simpler and more consistent user interface, integrated pipeline processing, and a number of improved image analysis tools. However, the principles have not changed. Most commands remain quasi identical in IMAGER and MAPPING, with basically the same control parameters.

Actions which were driven by widgets have often been replaced by simple commands (with intuitive names). A quick look at the HELP will guide you to the commands that will do what widgets were doing before. So you can skip over the fundamentals, and focus on the new tools offered by IMAGER. In practice, for you, commands will make fairly reasonable choices of default parameters. Just do not forget to WRITE your data.

In short, the biggest changes for MAPPING users are
- the lack of use of the GO command, and thus of the NAME and TYPE variables
- the suppression of MAP_RA MAP_DEC MAP_ANGLE and UV_SHIFT which have been replaced by MAP_CENTER
- the replacement of all tasks by simpler, standard commands.

Changing the defaults may be needed only for practical reasons:

- Optimizing computation time. This includes adjusting the field of view (variable MAP_FIELD), resampling in velocity (command UV_RESAMPLE) and time averaging (command UV_TIME)
- Comparing strictly different images (on a pixel per pixel basis). Adjusting MAP_SIZE and MAP_CELL will become necessary in such a case.

The names of variables and most commands have been kept from MAPPING, old names appear in the HELP whenever they have been replaced.

3.2.1 Forget INPUT and GO command

However, the GO command is totally obsolete. In general any GO SOMETHING is simply replaced by the equivalent command SOMETHING: e.g. type VIEW instead of GO VIEW. GO PLOT (or its variants GO BIT, GO NICE and GO MAP) and GO UVSHOW are replaced by the SHOW command, which offers many more possibilities.

The capabilities given by the INPUT MAPPING command have in general be replaced by the more flexible use of the question mark (?, or ??, or even ???) as arguments to individual IMAGER commands. Most commands display their control parameters with such arguments.

Take a little time to browse through HELP SHOW and HELP VIEW to get familiar with IMAGER display capabilities.

3.3 IMAGER for huge data sets: 100 000 channel imaging with NOEMA or ALMA

For NOEMA, with the advent of PolyFiX, the paradigm has changed: a spectral window may contain a substantial number of interesting spectral lines, not just one as with the previous narrow band spectral correlator. The same can happen (although to a lesser extent) with ALMA.

That is perhaps where IMAGER is most helpful. It contains new tools to help you automating this tedious part. The primary tool is command UV_PREVIEW which, provided one or more spectral line catalogs have been defined (see command CATALOG), allows to semi-automate the line identification among many UV tables.

The script @ noema-scan used after a UV_PREVIEW on a wide band UV table will help you identify who is who among the 64 spectral windows that PolyFix may provide.

A widget (driven by script imager_init) also allows simple image creation from any UV table, including continuum subtraction, referencing to different frequencies and velocities, and spectral resampling. Script @imager-one offers another alternative. Both can be useful examples for more advanced processing.

A much more advanced tool is the PIPELINE command that streamlines and even fully automates the imaging process of a whole ensemble of UV tables. See Section 13 for details.

3.4 IMAGER for lazy or overbooked ones

If you have many spectral lines, and need results very quickly, the PIPELINEs command. It can provide you one image for each spectral line specified in your line catalog that fall into the bandwidth of your data.

The default will do a pretty good job on most data sets, but can be customized for a better result. See Section 13 for details.

3.5 IMAGER for ALMA data

IMAGER implements in CASA the `casagildas()` tool that automates the transfer of calibrated visibilities from a CASA Measurement Set to UVFITS files that can be readily imaged with IMAGER.

3.6 IMAGER for image analysis and publication plots

IMAGER also includes general data-cube handling tools, such as spectral or spatial resampling, re-projection, image combinations, and several elaborate viewers that can be used to explore data cubes and/or prepare publications quality plots. See Sections 12

The display commands (but not the data handling tools) are also separately available in the VIEWER program.

4 IMAGER principles

4.1 Overview of the data reduction and analysis

Once the data has been acquired by an interferometer such as the NOrthern Extended Millimter Array (NOEMA) or ALMA, two different approaches may be used for its reduction and analysis:

- The first possibility is to clearly separate 1) the calibration, 2) the imaging and deconvolution and 3) the analysis.
- The second possibility is to merge in a single step calibration and imaging. This possibility is known as self-calibration.

While CASA uses the second paradigm, GILDAS mainly implements the first approach, as the program and the data format used for each step is different, but still allows convenient self-calibration. The basic instrumental calibration of NOEMA data is done inside CLIC on the NOEMA raw data format and the outcome is a *uv* table, which contains only calibrated visibilities of the astronomical source. The imaging and deconvolution are done inside IMAGER on the calibrated *uv* table and deliver mainly an `lmv` spectral cube (2 axes of coordinates and 1 axis

of velocity/frequency). IMAGER includes VIEWER, a superset of GREG, for the visualization, and provides additional image analysis functionalities, which are not specific to an interferometric use (*e.g.* they can be used with the IRAM 30 m spectral cubes as well).

The choice of clearly separating calibration and imaging+deconvolution was taken at start of the Plateau de Bure Interferometer, when the limiting number of antennas prevented the use of self-calibration. While many points of the calibration algorithms inside CLIC are specific to NOEMA data (in particular its range of Signal-to-Noise ratio), the algorithms of imaging+deconvolution can be used in many different contexts and the visualization and analysis of spectra cubes is mainly independent of the instrument that delivered the data. This last point implies that users can import data from ALMA (mainly through FITS format) in IMAGER for imaging and deconvolution, and in VIEWER for visualization. But the reverse is also true. While calibration of NOEMA data should be done inside CLIC, imaging+deconvolution and visualization+analysis can be done in other softwares (*e.g.* MIRIAD, AIPS, CASA for the imaging and deconvolution and KARMA for the visualization and analysis).

With the improvement of NOEMA (increase of the number of antennas and better receiver sensitivities) and with the advent of a new generation of interferometer (ALMA), the additional step of self-calibration may improve the consistency of the final results by imposing additional consistent constraints on the calibration. This self-calibration step is further presented in Section9.

4.2 The structure of the IMAGER program

4.2.1 Structure and recommendations

The IMAGER program supports

- The manipulation (*e.g.* resampling), visualization and flagging of *uv* tables;
- The imaging of *uv* tables in dirty maps and beams;
- The deconvolution of dirty maps;
- The inclusion of short-spacings;
- The visualization and analysis of spectral cubes;
- The self-calibration.

It consists in a collection of commands, either dedicated to image display (the `DISPLAY\` language), UV handling and imaging with deconvolution (the `CLEAN\` language), implementing basic functionalities (the `SIC\`, `GREG\`, or `CALIBRATE\` families of languages), advanced methods and image analysis tools (`ADVANCED\` language) or even complex suites of operations (the `BUNDLES\` language). A complete pipeline is available through the `IMAGER\` language.

A few additional widgets are grouped in the IMAGER main menu to provide more integrated interfaces to the above possibilities, or more elaborate control. While these widgets may still evolve to offer more flexibility, the commands are in general very stable, though minor syntax adjustments may occur (usually to implement a more convenient syntax, but keeping as much as possible backward compatibility as a major constraint).

4.2.2 Implementation issues

The new implementation of IMAGER and in particular of the UV_MAP and CLEAN commands uses most of the older code, but re-arranged such that ensembles of contiguous channels ("chunks") are treated at once and share the same synthesized beam. Deconvolution with CLEAN then proceeds by using the synthesized beam with the appropriate frequency for each channel. The user can control the "chunk" size, and hence the precision of the process given the desired field of view.

As a result of the new concept, beams (whether primary or synthesized) can be 4-D arrays, as they may depend on Frequency and Field.

4.3 Imaging/deconvolution: a brief sequence of commands

For the user, IMAGER reduces the number of actions to the strict minimum. The imaging sequence is always the same:

```
1- Reading data
   READ UV MyData.uvt /RANGE Min Max Type
   ! here, optionally use UV_TIME, UV_COMPRESS, UV_BASELINE to average data
   ! or UV_FILTER, UV_CONT to filter lines or remove continuum
2- Imaging
   UV_MAP
3- Deconvolving
   CLEAN
   ! here, optionally use UV_RESTORE
4- Looking at the result
   VIEW CLEAN      ! or SHOW CLEAN
5- Writing the result
   WRITE * MyData
```

- **Step 1:** Reading the specified internal buffer (here UV) from the input file (`.uvt` file type), loading only the channels falling in the range defined by the variables Min and Max, of Type CHANNEL, VELOCITY or FREQUENCY. IMAGER recognizes whether the UV table is for a single field or a mosaic. The only difference between the single field and mosaic cases is that IMAGER yields a Sky brightness image for Mosaics, while the computed sky brightness of a single field is not automatically corrected for the primary beam attenuation. Imaging for multiple fields will be presented in Section 7. Single Dish data can also be loaded in the following way : `READ SINGLE File`.

- **Step 2:** Computing a dirty map and beam from a UV data. UV_MAP processes single fields as well as Mosaics.

- **Step 3:** Deconvolving the DIRTY image map (a Single-field or Mosaic) using the dirty BEAM with the current METHOD. The default for the SIC variable METHOD is HOGBOM, the other supported methods being CLARK, MRC, MULTI and SDI. See `CLEAN ?` for the other SIC variables controlling the deconvolution process. The outputs are the CLEAN and RESIDUAL images, and the Clean Component Table CCT, all being stored in dedicated SIC variables.

- **Step 4:** Plotting the result in the specified internal buffer (CLEAN). Optionaly, the user can restrict the plot to a subset of channels through the optional arguments First and Last. `SHOW CLEAN` can also be used instead, and produces a different type of plot.

- **Step 5:** Writing all modified image-like buffers (not the UV tables) under the common file name "MyData". In the case of the present example, the following files are produced: MyData.lmv, MyData.lmv-clean, MyData.cct, MyData.beam, which correspond to the buffers: DIRTY, CLEAN, CCT, and BEAM, respectively. WRITE UV MyData would only write the internal buffer (UV) in the file MyData.uvt (the default extension corresponds to the specified buffer data type).

4.4 Getting Help: the HELP command and the ? token

As with any SIC based program, simple call to HELP will display the various languages (e.g., SIC\, GREG\, CALIBRATE\, CLEAN\) accessible to the help documentation and list some commands with available documentation. Note that CLEAN is a command and CLEAN\ a language (i.e., a family of commands). The language of a given command is recalled in the help of each command.

Example: the command APPLY belongs to the language CALIBRATE\, it has one argument which can be AMPLI or PHASE exclusively, one optional argument gain, and one option /FLAG.

```
IMAGER> help apply
        [CALIBRATE\]APPLY [AMPLI|PHASE [gain]] [/FLAG]
```

In the provided description, arguments within [] are optional. Upper case arguments are fixed keywords that can (in general) be abbreviated. A | character separates the allowed keyword values. Lower-case arguments are expected to be numbers, text, filenames, SIC variables or expressions that can be evaluated.

A brief description of the imager program can be obtained through:

```
IMAGER> help imager
USER\IMAGER = "@ welcome.ima"

      IMAGER is a interferometric imaging package, tailored for usage
  simplicity and efficiency for multi-spectral data sets.

     The basic concept of IMAGER is the use of a simple
         READ data  - ACTION(s) - [SHOW or VIEW] - WRITE
  sequence of commands, minimizing the data I/O as much as possible.
  Automatic guesses of appropriate default values for the ACTIONs
  parameters is implemented whenever possible.

Additional Help Available:
 Actions       BEAM_Handlin MAP_Handling UV_Handling  WebPage
 HOW_TO        MAPPING
```

In addition, finding documentation and help for the IMAGER commands can be done in three different ways:

- a simple call to the HELP command provides a description of the command and its arguments and options
- the command name followed by one or more questions marks will display some partial help on the command and its most useful parameters ("Command ?"), its second level parameters for advanced users ("Command ??"), all its parameters ("Command ???").

Documentation on subtopics of a given command (e.g., Variables, Arguments, or Results) can be obtained though: HELP command subtopic. The list of available subtopics is found at the bottom of the documentation of each command:

```
IMAGER> help uv_map
[...]
Additional Help Available:
 Mosaics      /FIELDS      /RANGE       /TRUNCATE
 Variables:
              BEAM_STEP    MAP_CELL     MAP_CENTER   MAP_CONVOLUT MAP_FIELD
 MAP_POWER    MAP_PRECIS   MAP_ROBUST   MAP_ROUNDING MAP_SHIFT    MAP_SIZE
 MAP_TAPEREXP MAP_TRUNCATE MAP_UVTAPER  MAP_UVCELL   MAP_VERSION
 Old_Names:
              convolution  map_angle    map_dec      map_ra       mcol
 uv_cell      uv_shift     uv_taper     taper_expo   wcol         weight_mode
```

Subtopics that appear in UpperCase in the `Additional Help Available` list are case insensitive. Subtopics that are in LowerCase of mixed case are strictly case sensitive.

Example: the following command will list the control variables of the UV_MAP function and describe the associated parameter(s):

```
IMAGER> help uv_map Variables
UV_MAP Variables:
                [CLEAN\]UV_MAP ?
        Will list all MAP_* variables controlling the UV_MAP parameters.

    The list of control variables is (by alphabetic order, with  the  corre-
    sponding old names used by Mapping on the right)

    New names       [   unit]       -- Description --    % Old Name
    BEAM_STEP       [       ]  Channels per dirty beam  % MAP_BEAM_STEP
    MAP_CELL        [ arcsec]  Image pixel size
    MAP_CENTER      [ string]  RA, Dec of map center, and Position Angle
    MAP_CONVOLUTION [       ]  Convolution function     % CONVOLUTION
    MAP_FIELD       [ arcsec]  Map field of view
    MAP_POWER       [       ]  Maximum exponent of 3 and 5 allowed in MAP_SIZE
    MAP_PRECIS      [       ]  Fraction of pixel tolerance on beam matching
    MAP_ROBUST      [       ]  Robustness factor        % UV_CELL[2]
    MAP_ROUNDING    [       ]  Precision of MAP_SIZE
    MAP_SIZE        [       ]  Number of pixels
    MAP_TAPEREXPO   [       ]  Taper exponent           % TAPER_EXPO
    MAP_TRUNCATE    [      %]  Mosaic truncation level
    MAP_UVCELL      [      m]  UV cell size             % UV_CELL[1]
    MAP_UVTAPER     [m,m,deg]  Gaussian taper           % UV_TAPER
    MAP_VERSION     [       ]  Code version (0 new, -1 old)

    See HELP UV_MAP Old_names:  for deprecated variable names.
```

A more detailed description (type, size) of a given variable can be obtained through `help command variable`, as in this example:

```
IMAGER> help uv_map map_uvtaper
UV_MAP MAP_UVTAPER

      MAP_UVTAPER[3]  Real

    Parameters of the tapering function (Gaussian if MAP_TAPEREXPO = 2): ma-
    jor axis at 1/e level [m], minor axis at 1/e level [m], and position an-
    gle [deg].
```

MAP_UVTAPER is an array that requires 3 values of type Real.

The default values of the useful parameters are checked through Command ?[1]

```
 UV_MAP makes a dirty image and a dirty beam from the UV data

 Behaviour is controlled by a number of SIC Variables
 - BEAM_STEP and MAP_PRECIS control the dirty beam precision
 - MAP_CENTER controls shifting ang rotation
 - MAP_CELL[2], MAP_SIZE[2], MAP_FIELD[2] control the map sampling
 - MAP_UVTAPER[3], MAP_UVCELL and MAP_ROBUST
     control the beam shape and weighting scheme

---- Basic parameters                          Selected       Recommended
  Map Size (pixels)              MAP_SIZE       [ 0 0 ]    [ 0 0 ]
  Field of view (arcsec)         MAP_FIELD      [ 0 0 ]    [ 0 0 ]
  Pixel size (arcsec)            MAP_CELL       [ 0 0 ]    [ 0 0 ]
  Map center                     MAP_CENTER     [  ]
  Robust weighting parameter     MAP_ROBUST     [ 0 ]
  UV cell size (meter)           MAP_UVCELL     [ 0 ]          [ 0 ]
  UV Taper (m,m,)               MAP_UVTAPER    [ 0 0 0 ]
```

For commands that support the ? mechanism, Command ? ControlVariable and, in most cases, Command ControlVariable ? are shortcuts to HELP Command ControlVariable.

4.5 IMAGER versus CASA: controlling the number of synthesized beams, BEAM_STEP

In aperture synthesis, the angular resolution scales with the frequency: in fact the whole Fourier plane scales with this frequency, so that the angular scaled defined by the baselines varies across the frequency coverage. Accordingly, when using sufficiently wide bandwidths (and/or imaging sufficiently large areas), it becomes important to account correctly for this effect.

The CASA and IMAGER approaches to this problem differ. CASA systematically produces one synthesized beam per spectral channel, leading to a frequency variable spatial resolution. In this approach, CASA can handle different weighting for individual channels, optimizing signal to noise, but the interpretation of the data becomes more difficult since there is no unique angular resolution after deconvolution.

The approach offered by IMAGER is different, and can be controlled by variable BEAM_STEP.

[1]For users familiar with MAPPING, the question mark replaces the capabilities of the INPUT Command although the output format may be slightly different.

- BEAM_STEP = 0 instructs IMAGER to produce a single synthesized beam for all channels. This is appropriate for narrow bands and spectral channels with very similar noise (ideally identical). Unavoidably, the synthesized beam is only an approximation, exact only at the reference frequency.
- BEAM_STEP = N instructs IMAGER to ensure that N consecutive channels share the same synthesized beam. This can be used when channels have the same weights (same effective UV coverage) but the overall bandwidth is large.
- BEAM_STEP = -1 instructs IMAGER to derive the numnber N of consecutive channels that can shared the same beam to within a certain precision in the imaged field of view. This precision is given by MAP_PRECIS. The derived N will depend of this number, on the relative bandwidth and on the size of the synthesized field (hence MAP_FIELD or MAP_CELL $\times$ MAP_SIZE).
- BEAM_STEP = 1 mimicks to some extent the behaviour of CASA.

However, contrary to CASA, IMAGER can use a common clean beam in all cases. This is achieved through the use of the so-called **JvM** factor, which approximately scales the undeconvolved residuals to match the flux scale of the Clean components. This behaviour is only obtained through the UV_RESTORE command.

BEAM_STEP is a key control parameter for IMAGER. The IMAGER pipelines (see Section13 use sensible default for each telescope, but users should be aware of this possible difference when comparing with CASA results.

5 The input data: UV Tables

The main goal of IMAGER is to convert interferometric measurements stored in *uv* tables into images suitable for astrophysical interpretation. *uv* tables contain a set of visibilities. *uv* tables being the starting point, IMAGER contains a number of commands to read then and handle them.

5.1 In a nutshell

For NOEMA, create with CLIC a UV table from the current index of cross-correlation observations and selected spectral window, and read it with IMAGER

```
$ clic
CLIC> (FILE ; FIND ; SET )   ! Build your index and spectral Window selection
CLIC> TABLE MyTable.uvt [NEW] [/MOSAIC]  !  Create a UV table
CLIC> EXIT
$ imager
IMAGER> READ MyTable.uvt [/Options ]
```

A complete set of UV tables for all spectral windows and all sources can be automatically created from an ensemble of `.hpb` data files by

```
$ clic
CLIC> @ all-tables
CLIC> EXIT
$ imager
IMAGER> SIC FIND *.uvt
IMAGER> READ UV 'dir%file[1]'  ! For the first one...
```

For ALMA, create with CASA a list of UVFITS files from a Measuremet Set, convert them with IMAGER

```
$ casa
CASA <2>: vis='MyMeasurementSet.ms'
CASA <3>: casagildas()
CASA <4>: casagildas("Do")
CASA <5>: exit()
$ imager
IMAGER> sic find *.uvfits
IMAGER> for string /in dir%file
IMAGER:   @ fits_to_uvt 'string'
IMAGER: next
```

or, even simpler, by using the PIPELINE command.

```
$ imager
IMAGER> PIPELINE ORGANIZE
```

Conventions for file naming are described in Section 5.4. Each *uv* table can later be read separately.

5.2 UV table description

5.2.1 UV table data format

A *uv* table is a specific 2-D Gildas table, with a few additional informations in the header, and a special interpretation of the data organisation.

In a standard *uv* table, each *line* describes a visibility. Here a *line* designate either the first or second axis of the table, and a *column* the other one. *uv* tables may appear in both orders. The default one is *line* on 1st axis (`.uvt` ordering, used by most application). The `.tuv` ordering obtained by a 21 transposition is used essentially for display, as in this case the *column* has the same meaning as for the COLUMN of GREG .

The number of lines of a *uv* table is thus the number of visibilities described in the table. Each *column* of the table stores a particular property of the visibilities, namely:

Column 1 U in meters;

Column 2 V in meters;

Column 3 W in meters or Scan number;

Column 4 Observation date (integer CLASS/CLIC Day Number[2]);

Column 5 Time in seconds since 0:00 UT of above date;

Column 6 Number of the first antenna used to measure the visibility;

Column 7 Number of the second antenna used to measure the visibility;

Column 8 Real part for the first frequency channel;

Column 9 Imaginary part for the first frequency channel;

Column 10 Weight for the first frequency channel;

Columns 11-13 Same as column 8-10 but for the second frequency channel, or for the second Stokes parameter of this channel.

... etc. . . for all channels

Columns N-ntrail+1 ... N Trailing columns after the channel visibilities.

If a *uv* table describes `nvis` visibility spectra composed of `nchan` frequency channels, each with `nstokes` Stokes parameters, the size of the table will thus be: `nvis` lines of `7+3*nchan*nstokes+ntrail` columns, where `ntrail` is the number of trailing columns.

5.2.2 *uv* header

A *uv* table header contains all the informations of a GDF header but some of these informations have a special meaning in this context. Command HEADER is the standard way inside GILDAS to display in a human readable way the header of GDF file. For instance, the command

```
IMAGER> header gag_demo:demo-line.uvt
```

would display

[2]The CLASS/CLIC is a "radio Julian date" (or "Jansky Julian date"), which starts as -2^{15} on the date of the first radio observation by Karl Jansky. It is thus the Modified Julian date minus 60549. That choice was made to maximize the time interval over which radio astronomical data could be usefully stored in an `integer*2`, back when 2 bytes of header space per spectrum were a significant consideration. This date has little meaning outside the rather sparse community of souls gathered around the CLASS and CLIC programs, however...

```
 1  W-GDF,  UNKNOWN Velocity type defaulted to LSR
 2  File : /Users/guilloteau/gildas/gildas-exe-dev/demo/demo-line.uvt  REAL*4
 3  Size        Reference Pixel           Value                   Increment
 4       103   16.0000000000000       220398.688000000     -0.183792725205421
 5      9146   0.00000000000000       0.00000000000000       1.00000000000000
 6  Blanking value and tolerance      1.23455997E+34   0.0000000
 7  Source name         GG_TAU
 8  Map unit            Jy
 9  Axis type           UV-DATA      RANDOM
10  Coordinate system   EQUATORIAL          Velocity    LSR
11  Right Ascension   04:32:30.34200        Declination       17:31:40.5230
12  Lii         0.000000000000000           Bii       0.000000000000000
13  Equinox             2000.0000
14  Projection type     AZIMUTHAL           Angle     0.000000000000000
15  Axis 0     A0     04:32:30.34200        Axis 0     D0     17:31:40.5230
16  Baselines               0.0       0.0
17  Axis 1 Line Name    13CO(21)            Rest Frequency   220398.6880000000
18  Resolution in Velocity   0.25000000     in Frequency        -0.18379273
19  Offset in Velocity        6.3000002     Doppler Velocity     -40.755900
20  Beam                    0.00               0.00                 0.00
21  NO Noise level
22  NO Proper motion
23  NO Telescope section
24  UV Data    Channels:     32, Stokes: 1 None        Visibilities:        9146
25  Column            1 (Size 1) contains U
26  Column            2 (Size 1) contains V
27  Column            4 (Size 1) contains DATE
28  Column            5 (Size 1) contains TIME
29  Column            6 (Size 1) contains IANT
30  Column            7 (Size 1) contains JANT
31  Column            3 (Size 1) contains SCAN
```

Comments:

Line 1 Indicates the velocity frame. If not present in the table (as here), it is assumed to be LSR.

Line 2 Indicates the filename associated to the currently displayed header.

Lines 3-5 Display the dimensions of the associated array. Here it is a rank 2 array of dimension 103 *columns* times 9146 *lines*, *i.e.* 9146 visibility spectra of 32 frequency channels. Line 4 describes the frequency axis of the visibility spectra stored in the *uv* table. Be careful that this is a convention, *i.e.* it must be decoded using the particular form of the table. In our case, each spectra has 32 frequency channels of width -183.8 kHz, the frequency of the reference pixel 16.0 corresponding to 220398.688 MHz. This last frequency is the frequency delivered by the correlator, *i.e.* seen by the observatory. In particular, this is the frequency that must be used to compute the primary beam of the interferometer.

Line 8 Indicates the unit of the real and imaginary parts of the visibilities, normally the Jansky (Jy).

Line 9 Indicates that this is *uv* table (`UV-DATA` and `RANDOM`).

Lines 10-13 Describe the coordinate system.

Lines 14-15 Describe the projection system. In the *uv* table format, `AO` and `DO` indicate the phase center while `Right Ascension` and `Declination` indicate where the antenna pointed when acquiring the signal. These informations are in general identical for single field imaging and different for mosaicing.

Lines 16 Indicates the baseline range in meters (m).

Lines 17-19 Describe additional information about the frequency axis of the visibility spectra. In particular, the rest frequency (here 220398.688 MHz, that of the 13CO J=2-1 line) corresponding to a velocity of 6.3 km/s in the velocity frame indicated at line 1 (in general LSR). Frequencies are always in MHz, and velocities always in km/s.

Line 20 Indicates the primary beam size of the interferometer in radian. This is an obsolescent way to pass the size of the interferometer antennas.

Line 21 The noise section has no meaning for the UV table.

Line 22 If present, proper motions are given in mas/yr. The epoch is used as the time origin.

Line 23 If the TELESCOPE section is present, this line would indicate telescope name, its geographic coordinates and the antenna diameter (in m). This section contains also the information to compute the primary beam.

Line 24 UV data section: number of channels, number of Stokes parameters and number of visibilities.

Line 25 to end Special columns description, including the 7 first ones and the `ntrail` trailing ones.

In particular, **Mosaic** *uv* tables contain two trailing columns named `L_PHASE_OFF, M_PHASE_OFF` for the so-called "Phase Offset Mosaics", or `X_POINT_OFF, Y_POINT_OFF` for the "Pointing Offset Mosaics", which contains the angular offsets of the field centers with respect to the Phase reference.

5.3 NOEMA: UV Tables from CLIC

For NOEMA, creating UV tables suitable for IMAGER is straightforward in CLIC. From a list of selected cross-correlation scans, the command `TABLE` creates a *uv* table with the appropriate format (or adds to an existing one). Mosaics are treated in much the same way: it is sufficient to add the `/MOSAIC` option to it.

5.4 ALMA: UV Tables from CASA

Importing *uv* data from CASA to IMAGER is a little more complex, because of the totally different design philosophy of the two packages. CASA intends to solve the *Measurement Equation*, whatever the complexity of this process. It is a all-in-one package for this purpose, where calibration and imaging are deeply intermixed and use a unified data format. As a result, a CASA Measurement Set is a complex architecture encompassing relations between many components stored as Tables in a directory-like tree. It can handle calibrated data, calibration tables, multisource data sets, raw data in the same architecture, allowing to retain all information to process complex images, such as multi-frequency synthesis of polarized emission observed in a mosaic of fields.

On the contrary, IMAGER works on calibrated data only, and with a single source (though possibly a mosaic) and single spectral setup at once.

The importation process goes through UVFITS data files, produced by `exportuvfits` from CASA, and imported through the FITS command of SIC. However, the UVFITS format is an incomplete standard, and to recover properly all associated informations, these two steps have been encapsulated in sophisticated scripts on each side.

5.4.1 On the CASA side

The proper exportation is done by invoking the `casagildas()` tool. [3]. `casagildas()` scans the Measurement Set using the `listobs()` tool, then uses a Python script that parses the output of `listobs()` to prepare a Python script named `casa_uvfits.py`.

It is then up to the user to execute this Python script, using the `casagildas("Do")` command that creates one UVFITS file per Source and Spectral window in the input Measurement Set. Thus, the sequence

```
vis='MyMeasurementSet.ms'   # Setup the input Measurement Set filename
casagildas()                # List its content and create the export script
casagildas("Do")            # Execute the export script
```

will create a set of

```
Source-Frequency.uvfits
```

files, where `Source` is the source name and `Frequency` is the central frequency of the spectral window in MHz, for all combinations of sources and spectral windows.

5.4.2 On the IMAGER side

The UVFITS files have first to be converted to *uv* tables. This can be done by a simple script

```
sic find *.uvfits
for string /in dir%file
  @ fits_to_uvt 'string'
next
```

Each `Source-Frequency.uvfits` file will be converted to a `Source-Frequency.uvt` *uv* table (the original `.uvfits` file is kept too).

The `fits_to_uvt` script is based on the FITS command of SIC, but augmented with a number of tests to recognize the proper layout of the UVFITS file, as this layout depends on which CASA version was used, and whether it is a single source or multi source file.

The `fits_to_uvt` script has a number of options. Help can be obtained by typing
`@ fits_to_uvt ?`

Direct reading of UVFITS data is also partially supported: see Section 5.5.

5.4.3 Current assumptions and limitations

GILDAS works in the `LSR` velocity frame and has limited polarization capabilities (IMAGER can only process one polarization state at a time, see Section 15 for details). Thus the *casagildas()* `-fits_to_uvt` sequence makes several assumptions:

[3] This tool is made available to CASA by IMAGER (IMAGER does not need to be active, but must have been executed once before)

- FDM (Frequency Division Mode) spectral windows are converted to LSR frame using the *mstransform* tool
- TDM (Time Division Mode) spectral windows, which have low spectral resolution (15 MHz), are assumed to be pure continuum data, and remain in the default frame of the original measurement set.
- The conversion from UVFITS to *uv* tables assumes that the data is unpolarized[4], and merges the initial polarization states in an optimal way from the signal to noise point of view (i.e. using the respective weights of the two parallel hand states).
- *casagildas()* takes the same input parameters as the *listobs()* facility. Source and/or spectral window selection can thus be made by the user at this stage
- *casagildas()* takes the same input parameters as the *mstransform()* facility. Time integration may be done at this stage, but this may limit the performance of self-calibration at a later stage.

5.5 Reading UV tables

uv tables are simply read into IMAGER by READ UV. The rest frequency to be used to compute the velocity scale is normally found in the table, but can be overridden using the READ /FREQUENCY option. By default, the whole table is read, but a subset of the channels can be read using the READ /RANGE option (with the velocity scale as above).

UVFITS file may even be read directly as *uv* tables, applying the same conversion methods than the fits_to_uvt script. However, this capability is still only partially supported. While it will work for basic (i.e. in memory) handling, full access to the *uv* data is only possible on the native format: this includes subset selection, as well as direct operations on *uv* data files through the /FILE option of many UV_... commands.

5.6 UV Table handling

Besides the READ UV and WRITE UV commands to read or write *uv* tables, IMAGER has a number of commands to manipulate the current *uv* table buffer. These commands have names starting by UV_. Most of them are in the CLEAN\ language, some in the ADVANCED\ one.

IMAGER works using UV buffers. Most commands only work on the current UV buffer, but some of them keep track of the previous buffer to allow the user to revert the operation.

Data inspection and editing:
- SHOW COVERAGE display the *uv* coverage
- SHOW UV display the *uv* data
- UV_FLAG allow flagging visibilities
- UV_PREVIEW provides a quick view of the visibilities as a function of frequencies, and attempts to automatically find the continuum level and parts of the bandwidth with spectral line emissions.

Data size reduction routines:
- UV_COMPRESS is a simple spectral smoothing, providing only channel averaging by integer number of channels.
- UV_RESAMPLE provides a more flexible spectral smoothing and resampling facility.

[4] unless the /STOKES option is present, see Section 15

- UV_TIME can be used to time-average the UV data set, leading to faster processing. However, using UV_TIME too early may limit your ability to perform accurate phase self-calibration.

Continuum processing commands:
- UV_BASELINE allows to remove the continuum baseline, by 0th or 1st order baseline fitting of each visibility.
- Conversely, UV_FILTER will filter the spectral line range to leave only the channels with continuum emission. Both UV_BASELINE and UV_FILTER can use the results provided by UV_PREVIEW to specify where spectral lines may be found.
- UV_CONTINUUM converts a spectral line *uv* table into a bandwidth synthesis continuum *uv* table. UV_CONTINUUM is only useful for UV plane analysis: see Section 10 for details.
- UV_MERGE /FILE can merge several UV continuum tables with a specified spectral index to optimize the sensitivity.

Image preparation:
- UV_CHECK inspects the *uv* data to figure out how many different synthesized beams are needed.
- UV_SHORT adds the short (or zero) spacing information provided by an additional single dish data, read by READ SINGLE.
- UV_STAT evaluates the impact of robust weighting and tapering on the synthesized beam. It provides recommendations for the image and pixel sizes.
- UV_TRUNCATE restricts the *uv* baseline length range.

UV Plane analysis:
- UV_FIT fit simple source models to the visbilities.
- SHOW UV_FIT display the fit results, usually as a function of frequency, but also in other ways.

Miscellaneous:
- UV_DEPROJECT de-projects the (u, v) coordinates given a specified phase center, orientation and inclination. This can be useful for inclined, flattened, nearly axi-symmetric structures such as proto-planetary disks or galaxies.
- UV_CIRCLE and UV_RADIAL compute the azimutal average of the visibilities. They are useful for rotationally symmetric structures such as proto-planetary disks or circumstellar envelopes, for example.
- UV_REWEIGHT changes the visibility weights.
- UV_SHIFT changes the phase center. This can be useful for UV plane analysis.
- UV_MERGE /FILE can merge several UV tables, Line or Continuum. It also allows stacking different spectral lines, by aligning them in velocity, as well as stacking emission from different sources.

The remaining UV_... commands are related to imaging and deconvolution: UV_MAP computes the dirty image, UV_RESTORE computes the Clean image from a Clean component list by removal of the Clean components in the *uv* plane, and imaging of the residuals. UV_RESIDUAL just computes the residuals by subtraction of the Clean components.

Finally, UV_SELF, in the CALIBRATE\ language, is a specific variant of UV_MAP used to compute the intermediate images required for self-calibration. It is not intended for direct use by normal users.

6 Single-field imaging and deconvolution

6.1 In a nutshell

```
1 read uv YourData
2 uv_stat
3 uv_map
4 clean
5 view clean
6 write * YourResult
```

1. Read your *uv* data
2. Have a look at its header, and get recommendations on image characteristics.
3. Image it
4. Deconvolve
5. see the result
6. Save the result if OK.

You are done. And this is often good. However, it may take a while, and the angular resolution and/or the brightness sensitivity may not be optimal. So, it may be worth for you to read the information below and adjust the control variables of UV_MAP

6.2 Measurement equation and other definitions

The measurement equation of an instrument is the relationship between the sky intensity and the measured quantities. The measurement equation for a millimeter interferometer is to a good approximation (after calibration)

$$\mathrm{V(u,v)} = \mathrm{FT}\{B_{\mathrm{primary}}.I_{\mathrm{source}}\}\mathrm{(u,v)+N} \quad (1)$$

where $\mathrm{FT}\{F\}(u,v)$ is the bi-dimensional Fourier transform of the function F taken at the spatial frequency (u,v), I_{source} the sky intensity distribution, B_{primary} the primary beam of the interferometer (almost a Gaussian whose FWHM is the natural resolution of the single-dish antenna composing the interferometer), N some thermal noise and $V(u,v)$ the calibrated visibility at the spatial frequency (u,v). This measurement equation implies different kinds of problems.

1. The presence of noise leads to sensitivity problems.
2. The presence of the Fourier transform implies that visibilities belongs to the Fourier space while most (radio)astronomers are used to interpret images. A step of *imaging* is thus required to go from the *uv* plane to the image plane.
3. The multiplication of the sky intensity by the primary beam implies a distortion of the information about the intensity distribution of the source.
4. Finally, the main problem implied by this measurement equation is certainly the irregular, limited sampling of the *uv* plane because it implies that the information about the source intensity distribution is incomplete.

Deconvolution techniques are needed to overcome the incomplete sampling of the *uv* plane. To show how this can be done, we need additional definitions

- Let us call $V = \mathrm{FT}\{B_{\mathrm{primary}}.I_{\mathrm{source}}\}$ the continuous visibility function.
- The sampling function S is defined as
  - $S(u,v) = 1/\sigma^2$ at (u,v) spatial frequencies where visibilities are measured by the interferometer. σ is the rms noise predicted from the system temperature, antenna efficiency, integration time and bandwidth. The sampling function thus contains information on the relative weights of each visibility.
  - $S(u,v) = 0$ elsewhere.
- We finally call $B_{\mathrm{dirty}} = \mathrm{FT}^{-1}\{S\}$ the dirty beam.

If we forget about the noise, we can thus rewrite the measurement equation as

$$I_{\mathrm{dirty}} = \mathrm{FT}^{-1}\{S.V\}. \tag{2}$$

Using the property #1 of the Fourier transform (see Appendix), we obtain

$$I_{\mathrm{dirty}} = B_{\mathrm{dirty}} * \{B_{\mathrm{primary}}.I_{\mathrm{source}}\}, \tag{3}$$

where $*$ is the convolution symbol. Thus, the incompleteness of the *uv* sampling translates into the image plane as a convolution by the dirty beam, implying the need of deconvolution. From the last equation, it is easy to show that the dirty beam is the point spread function of the interferometer, *i.e.* its response at a point source. Indeed, for a point source at the phase center, $\{B_{\mathrm{primary}}.I_{\mathrm{source}}\} = I_{\mathrm{point}}$ at the phase center and 0 elsewhere and the convolution with a point source is equal to the simple product: $I_{\mathrm{dirty}} = B_{\mathrm{dirty}}.I_{\mathrm{point}} = B_{\mathrm{dirty}}$ for a point source of intensity $I_{\mathrm{point}} = 1$ Jy.

We note that Fourier transform are in general done through Fast Fourier Transform, which implies first a stage of re-interpolation of the visibilities on a regular grid in the *uv* plane, a process called *gridding*. This gridding step introduces a convolution in the *uv* space, and thus a multiplication by the Fourier transform of the gridding function in the image plane, which needs to be corrected later by division by this Fourier transform. It can be shown that despite this step, the convolution property mentionned before still holds.

6.3 Imaging

The process known as *imaging* consists in computing the dirty image and the dirty beam from the measured visibilities and the sampling function.

6.3.1 Image size and pixel size

Link between image size and *uv* cell size The gridding stage requires at least Nyquist sampling of the *uv* plane to avoid the artifact known as aliasing. This sampling depends on the source size.

For the signal, the source size is limited by the primary beam, so that the Nyquist sampling in the *uv* plane is obtained with a size of the grid cells equals to half the size of the antenna diameter. (this is the smallest spatial frequency that the interferometer can be sensitive to, *i.e.*

the natural resolution in the *uv* plane). In the image plane, this implies to make an image at least twice as large as the primary beam size (see Fourier transform property #2 in Appendix).

Unfortunately, the spatial frequencies of the noise are not bound: the noise increases at the edges of the produced image because of the noise aliasing and gridding correction.

Unless you have good reasons (such as a strong confusion source close to the primary beam), you should not choose too large an image size, since that would slow down imaging and deconvolution.

Link between pixel size and largest *uv* spatial frequency The largest sampled spatial frequency is directly linked to the synthesized beam size (*i.e.* the interferometer spatial resolution). The pixel size must be at least 1/2 the synthesized beam size to ensure Nyquist sampling in the image plane. However, Nyquist sampling is enough only when dealing with linear processes while deconvolution techniques are non-linear. It is thus recommended to select a pixel size between 1/3 and 1/4 of the synthesized beam to ease the deconvolution. Smaller pixel sizes would lead to larger images, and unduly slow down the imaging and deconvolution process.

6.3.2 Weighting and Tapering

The use of the visibility weights ($1/\sigma^2$) in the definition of the sampling function is called natural weighting as it is natural to weight each visibility by the inverse of noise variance. Natural weighting is also the way to maximize the point source sensitivity in the final image. However, the exact scaling of the sampling function is an additional degree of freedom in the imaging process. In particular, the user may change this scaling to give more or less weight to the long or short spatial frequencies.

We can thus introduce a weighting function $W(u, v)$ in the definitions of $B_{\rm dirty}$ and $I_{\rm dirty}$

$$B_{\rm dirty} = {\rm FT}^{-1}\left\{W.S\right\} \tag{4}$$

and

$$I_{\rm dirty} = {\rm FT}^{-1}\left\{W.S.V\right\}. \tag{5}$$

There are two main categories of weighting functions

Robust weighting In this case, W is computed to enhance the contribution of the large spatial frequencies. This is done by first computing the natural weight in each cell of the *uv* plane. Then W is derived so that

- The product $W.S$ in a *uv* cell is set to a constant if the natural weight is larger that a given threshold;
- $W = 1$ (*i.e.* natural weighting) otherwise.

This decreases the weight of the well measured *uv* cells (*i.e.* very low noise cells) while it keeps natural weighting of the noisy cells. It happens that the cells of the outer *uv* plane (corresponding to the large interferometer configurations) are often noisier than the cells of the inner *uv* plane (just because there are less cells in the inner *uv* plane). Robust weighting thus increases the spatial resolution by emphasizing the large spatial frequencies at moderate cost in sensitivity for point sources (but with a larger loss for extended sources, see below).

Tapering is the apodization of the *uv* coverage by simple multiplication by a Gaussian

$$W = \exp\left\{-\frac{\left(u^2+v^2\right)}{t^2}\right\}, \tag{6}$$

where t is the tapering distance. This multiplication in the *uv* plane translates into a convolution by a Gaussian in the image plane, *i.e.* a smoothing of the result. The only purpose of this is to increase the sensitivity to extended structure. Tapering should never be used **alone** as this somehow implies that you throw away large spatial frequencies measured by the interferometer. It is only a way to extract the most information from the given data set. If you need more sensitivity to extended structures, use compact configuration of the arrays rather than extended configurations and tapering.

For more details on the whole imaging process the interested reader is referred to Guilloteau (2000).

6.3.3 Implementation (READ UV, UV_MAP and UV_STAT)

In IMAGER, gridding in the *uv* plane and computation of the dirty beam and image are coded in the UV_MAP command. This command works on an internal buffer containing the *uv* table read from a file through the READ UV command.

The UV_MAP command is controlled by a set of SIC variables named with the prefix MAP_. Suitable defaults are provided, so that only specific cases should require customization by the user. A description of the all variables can be obtained through HELP UV_MAP. UV_MAP ? also gives their default and current values.

Basic usage - image characterization:

Name	Dim/Type	Description
BEAM_STEP	Int	Number of channels per common dirty beam, if > 0. If 0 (default value), only one beam is produced in total. If -1, an automatic guess is performed from the map size and requested precision (MAP_PRECIS).
MAP_CELL	[2] Real	Pixel size in arcsecond. Enter 0,0 to let the task find the best values.
MAP_FIELD	[2] Real	Image size in arcsecond. MAP_FIELD has precedence over the number of pixels MAP_SIZE to define the actual map size when both variables are non-zero.
MAP_SIZE	[2] Int	Image size in pixels
MAP_CENTER	Char.	A character string to specify the new Map center and the new map orientation, see the next subsection related to the definition of the projection center of the image.

Weighting:

MAP_ROBUST	Real	Robust weighting factor, in range 0 - $+\infty$. 0 means Natural weighting (as $+\infty$, actually). 0.5 or 1 is usually a good choice for Robust Weighting. Default is 0,i.e. natural weighting. (Old name UV_CELL[1])
MAP_UVTAPER	[3] Real	Array of 3 values controlling the UV taper: major/minor axis at 1/e level [m,m] (first two values), and position angle ([deg], third value). By default (0,0,0). (Old name UV_TAPER[3]).
MAP_UVCELL	[2] Real	UV Cell size for Robust weighting. Default is 0, meaning that the cell size is derived from the antenna diameter. (Old name UV_CELL[2])
MAP_TAPEREXPO	Real	the taper exponent. Default 2, indicating Gaussian function. (Old name TAPER_EXPO).

Advanced:

MAP_PRECIS	Real	Position precision at the map edge, in fraction of pixel size, used (with the actual image size) to derive the actual number of channels which can share the same beam. Default value is 0.1.
MAP_POWER	Int	Rounding scheme for default image size, to numbers like $2^n 3^p 5^q$. p and q are less than or equal to MAP_POWER. Default value is 2, for smallest image size. For MAP_POWER = 0 MAP_SIZE is just a power of 2.
MAP_ROUNDING	Real	Maximum error between optimal size (MAP_FIELD / MAP_CELL) and rounded (as a power of $2^k 3^p 5^q$) MAP_SIZE.
MAP_TRUNCATE	Real	For a Mosaic, truncate the primary beam to the specified level (in fraction). Default value is 0.2.

Debug:

MAP_CONVOLUTION	Int	Gridding convolution mode in the uv plane. (default 5 for speroidal functions) (old name CONVOLUTION)
MAP_VERSION	Int	Version of code to be used. This is a temporary variable to allow comparison between the new and old codes without quitting IMAGER.

Parameters can be listed by the commands UV_MAP ? and CLEAN ?. More details are obtained using ?? or ??? as arguments instead of ?. A thorough description of each parameter can be obtained by typing: help UV_MAP MAP_* or help CLEAN CLEAN_*.

```
IMAGER> UV_MAP ?
 UV_MAP makes a dirty image and a dirty beam from the UV data

 Behaviour is controlled by a number of SIC Variables
 - BEAM_STEP and MAP_PRECIS control the dirty beam precision
 - MAP_CENTER controls shifting ang rotation
 - MAP_CELL[2], MAP_SIZE[2], MAP_FIELD[2] control the map sampling
 - MAP_UVTAPER[3], MAP_UVCELL and MAP_ROBUST
     control the beam shape and weighting scheme

---- Basic parameters                          Selected       Recommended
  Map Size (pixels)              MAP_SIZE      [ 0 0 ]   [ 0 0 ]
  Field of view (arcsec)         MAP_FIELD     [ 0 0 ]   [ 0 0 ]
  Pixel size (arcsec)            MAP_CELL      [ 0 0 ]   [ 0 0 ]
  Map center                     MAP_CENTER    [  ]
```

```
Robust weighting parameter     MAP_ROBUST      [ 0 ]
UV cell size (meter)           MAP_UVCELL      [ 0 ]          [ 0 ]
UV Taper (m,m,)               MAP_UVTAPER     [ 0 0 0 ]
```

6.3.4 Defining the Projection Center of the image

The command UV_MAP handles phase tracking center through its arguments, or through the string variable MAP_CENTER. The syntax of the arguments is the following:

UV_MAP [CenterX CenterY UNIT] [Angle] [/FIELDS FieldList] [/TRUNCATE Percent]

The allowed syntax for MAP_CENTER is described by HELP UV_MAP MAP_CENTER.

6.3.5 Typical imaging session

```
 1 read uv gag_demo:demo-single-imager
 2 uv_map ?
 3 uv_stat weight
 4 let map_robust 0.5
 5 uv_map ?
 6 uv_map
 7 show beam
 8 show dirty
 9 let map_size 128
10 uv_map
11 show beam
12 show dirty
13 hardcopy demo-dirty /dev eps
14 write dirty demo
15 write beam demo
```

Comments:

Step 1 Read the demo-single-imager.uvt *uv* table in an internal buffer.

Step 2 Check current state of the variables that control the imaging.

Steps 3-5 Select the robust weighting threshold (step 4) from the result of the UV_STAT command (step 3) and recheck the current state of the variables that control the imaging (step 5).

Steps 6-8 Image and plot the dirty beam and image.

Steps 9-12 Try a smaller size of the map as the default imaged field-of-view looked too large from previous plots.

Steps 13 Make a Post-Script file from the dirty image.

Steps 14-15 Write dirty image and beam in `demo.lmv` and `demo.beam` files for deconvolution in a future IMAGER session. These steps are optional as you may directly proceed to the deconvolution stage without writing the files.

6.4 Deconvolution

Once the dirty beam and the dirty image have been calculated, we want to derive an astronomically meaningful result, ideally the sky brightness. However, it is extremely difficult to recover

the intrinsic brightness distribution with an interferometer. Mathematically, the incomplete sampling of the *uv* plane implies that there is an infinite number of intensity distributions which are compatible with the constraints given by the measured visibilities. Fortunately, physics allow us to select some solutions from the infinite number that mathematics authorize. The goal of deconvolution is thus to find a sensible intensity distribution compatible with the measured visibilities. To reach this goal, deconvolution needs 1) some *a priori*, physically valid, assumptions about the source intensity distribution and 2) as much knowledge as possible about the dirty beam and the noise properties (in radioastronomy, both are well known). The best solution would obviously be to avoid deconvolution, *i.e.* to get a Gaussian dirty beam. For instance, the design of the compact configuration of ALMA has been thought with this goal in mind. However, this goal is out of reach for today's millimeter interferometers, even ALMA.

The simplest *a priori* knowledge that the user can feed to deconvolution algorithm is a rough idea of the emitting region in the source. The user defines a support inside which the signal is to be found while the outside is only made of sidelobes. The definition of a support considerably helps the convergence of deconvolution algorithms because it decreases the complexity of the problem (*i.e.* the size of the space to be searched for solutions). However, it can introduce important biases in the final solution if the support excludes part of the sky region that is really emitting. Support must be thus used with caution.

6.5 The family of `CLEAN` algorithms (`HOGBOM`, `CLARK`, `MX`, `SDI`, `MRC`, `MULTI`)

Radio astronomy interferometry made a significant step forward with the introduction of a robust deconvolution algorithm, known as `CLEAN`, by Högbom (1974).

6.5.1 `CLEAN` ideas

The family of `CLEAN` algorithms is based on the *a priori* assumption that the sky intensity distribution is a collection of point sources. The algorithms have three main steps

Initialization

- of the residual map to the dirty map;
- and of the clean component list to a `NULL` (*i.e.* zero) value.

Iterative search for point sources on the residual map. As those point sources are found,

- they are subtracted from the residual map;
- and then they are logged in the clean component list.

Restoration of the clean map 1) by convolution of the clean component list with the clean beam, *i.e.* a Gaussian whose size matches the synthesized beam size and 2) by addition of the residual map.

Stopping criteria Several criteria may be used to stop the iterative search of this "matching pursuit":

1. When the maximum of the absolute value of the residual map is lower than a fraction of the noise. This stopping criterion is adapted to *noise limited situations*, *i.e.* when empirical measures of the noise in the cleaned image give a value similar to the noise value estimated from the system temperatures.

2. When the maximum of the absolute value of the residual map is lower than a fraction of the maximum intensity of the original dirty map. This stopping criterion is adapted to *dynamic range limited situations*, *i.e.* when some part of the source is so intense that the associated side lobes are larger than the thermal noise. In this case, any empirical measure of the noise in the cleaned image will give a value larger than the noise value estimated from the system temperatures.

3. The total number of clean components. This is a sanity criterium in case the other ones would be badly tuned.

4. When the total flux remains stable.

Choosing the good stopping criterion is important because the deconvolution must go deep enough to recover weak extended flux but `CLEAN` algorithms start to diverge when the noise is cleaned too deep. Criterium 4 is thus in general preferable, but may lead to insufficient cleaning when the dirty beam is poor (by lack of *uv* coverage and/or because of phase noise). If (# 4) fails, a good compromise is to clean down to or slightly below (typically 0.8σ) the noise level.

Stability criterion The simplest way to control the convergence is the `CLEAN_STOP` character variable. It allows to express the stopping threshold in natural units, e.g. Jy, mJy, K, milliK, Noise or Sigma (the noise levels), or even % for peak percentage.

But Clean convergence can also be controlled by the usual `CLEAN_ARES` (#1, maximum Absolute RESidual value) , `CLEAN_FRES` (#2, maximum Fractional RESidual value) and `CLEAN_NITER` (#3, maximum Number of ITERations) criteria, plus `CLEAN_NCYCLE` for methods with major cycles. A fourth criterium (#4) is *convergence*, which is controlled by `CLEAN_NKEEP`, a number of components. Deconvolution of a given channel stops if the cumulative flux at iteration number N is smaller (resp. larger) than at iteration N-`CLEAN_NKEEP` for positive signals (resp. negative). In essence, `CLEAN_NKEEP` is the number of components when the signal is just above the noise. Experimentation with various types of images has shown that `CLEAN_NKEEP=70` is a good compromise.

However, criteria #1-3 can be set to 0, allowing IMAGER to automatically guess when to stop. In this case, IMAGER uses an absolute residual threshold equals to the noise level (available in `dirty%gil%noise`), and estimates a (very conservative) maximum number of Clean components.

Using `CLEAN_STOP` has precedence over the other ways of specifying the stopping criteria.

Formation of the `CLEAN` map The clean component list may be searched on an arbitrarily fine spatial grid without too much physical sense as the interferometer has a finite spatial resolution. The convolution by the clean beam thus reintroduces the finite resolution of the observation, an information which is missing from the list of clean components alone. This step is often called *a posteriori* regularization.

The shape (principally its size) of the clean beam used in the restoration step plays an important role. The clean beam is usually a fit of the main lobe (*i.e.* the inner part) of the dirty beam. This ensures that 1) the flux density estimation will be correct and 2) the addition of the residual map to the convolved list of clean component makes sense (*i.e.* the unit of the clean and residual maps approximately matches).

The final addition of the residual map plays a double role. First, it is a first order correction to insufficient deconvolution. Second, it enables noise estimate on the cleaned image since the residual image should be essentially noise when the deconvolution has converged.

Some odd dirty beams may lead to incorrect flux measurements. For example, if the dirty beam has a very narrow central peak superimposed on a rather broad plateau, the volume of the Gaussian fitted to the central peak does not match that of the dirty beam, and the flux scale will be incorrect. Data-reweighting is required to cure these peculiar situations, and this always implies a loss of sensitivity.

Super-resolution is the fact of restoring with a clean beam size smaller that the fit of the main lobe of the dirty beam. The underlying idea is to get a bit finer spatial resolution. However, it is a bad practice because it breaks the flux estimation and the usefulness of the addition of the residual maps. It is better to use robust weighting to emphasize the largest measured spatial frequencies.

6.5.2 Basic CLEAN algorithms (HOGBOM, CLARK and MX)

The main difference between the different basic CLEAN algorithms is the strategy for searching the point sources.

HOGBOM The simplest strategy of the iterative search was introduced by Högbom (1974). It works as follows

1. Localization of the strongest intensity pixel in the current residual map: $\max(|I_{\rm res}|)$.
2. Add $\gamma.\max(|I_{\rm res}|)$ and its spatial position to the clean component list.
3. Convolution of $\gamma.\max(|I_{\rm res}|)$ by the dirty beam.
4. Subtract the resulting convolution from the residual map in order to clean out the side lobes associated to the localized clean component.

γ is the loop gain. It controls the convergence of the method. In theory, $0 < \gamma < 2$. $\gamma = 1$ would in principle give faster convergence, since the remaining flux at one position is $\propto (1-\gamma)^{n_{\rm comp}}$, where $n_{\rm comp}$ is the number of clean components found at this position. But, in practice, one should use $\gamma \simeq 0.1 - 0.2$, depending on sidelobe levels, source structure and dynamic range. Indeed, deviations (such as thermal noise, phase noise or calibration errors) from an ideal convolution equation force to use low gain values in order to avoid non linear amplifications of errors.

An important property of HOGBOM algorithm is that only the inner quarter of the dirty image can be properly cleaned when dirty beam and images are computed on the same spatial grid. Indeed, the subtraction of the dirty sidelobes associated to any clean component is possible only in the spatial extent of the dirty beam image. When the user defines a support (*a priori* knowledge), the cleaned region becomes even smaller than the inner quarter of the dirty map.

CLARK The most popular variant to the HOGBOM algorithm is due to Clark (1980). The iterative search for point sources involves minor and major cycles.

In minor cycles, an HOGBOM search is performed with two limitations: 1) Only the brightest pixels are considered in the above step 1, and 2) the convolution of the found point sources (step 3 above) is done with a spatially truncated dirty beam [5]. Both limitations fasten the search but may lead to difficult convergence in cases where the secondary side lobes are a large fraction (*e.g.* 40%) of the main side lobe.

[5] It is also theoretically possible to do so with an intensity truncated beam.

In major cycles, the clean components found in the last minor cycle are removed in a single step from the residual map in the Fourier plane. The use of the Fourier transform enable to clean slightly more than the inner quarter of the map.

CLARK is faster than HOGBOM, but less stable.

MX The MX (or Cotton-Schwab, from the names of its authors) algorithm, due to Schwab (1984), is a variant of the CLARK algorithm in which the clean components are removed from the *uv* table at each major cycle. This is the most precise way of removing the found clean components because it avoids aliasing of the dirty sidelobes. A direct consequence is that this method enables to clean the largest region of the dirty map. However, this may be a relatively slow algorithm because the imaging step must be redone at each major cycle, although this speed issue could be compensated by the ability to use smaller images.

6.5.3 Advanced CLEAN algorithms to deal with extended structures (SDI, MULTI and MRC)

When the spatial dynamic of the imaged source is large (*i.e.* when the ratio of the largest source structure over the synthesized resolution is large), the basic CLEAN algorithms may (rarely) turn smooth area of the source into a serie of ridges and stripes. Indeed, when the dirty beam pattern is subtracted from a smooth feature of the dirty map, the sidelobes patterns appear in the residual map. The search for the next clean component will then pick first the pixels in the sidelobes pattern amplifying this pattern. Several variants of CLEAN have been devised to solve this problem.

SDI In the CLEAN variant proposed by Steer et al. (1984), extended features (instead of point sources) around the current maximum of the residual map are selected and removed in a single step. The simplest implementation redefines the notion of minor and major cycles of the CLARK algorithm. In the minor cycles, only the selection of the clean components is done by including all the pixels in the residual map that rise above a contour set at some fraction of the current peak level. In major cycles, all those components are removed together in the Fourier plane. The SDI algorithm may be instable if the fraction used for the selection of the clean components is badly chosen. SDI is very sensitive to the support selection.

MRC The Multi Resolution Clean MRC from Wakker & Schwarz (1988) is the first attempt to introduce the notion of cleaning at different scales. MRC works on two intermediate maps (strictly speaking MRC is a double–resolution CLEAN algorithm). The first map is a smoothed version of the dirty map and the second map, called difference map, is obtain by subtraction of the smoothed map from the original dirty map. Since the measurement equation is linear, both maps can be cleaned independently (using a smoothed and a difference dirty beam, respectively). The underlying idea is that extended sources in the dirty map will look like more "point-like" with respect to the smoothed dirty beam in the smoothed map. MRC is faster than the basic CLEAN algorithms because fewer clean components are needed to reproduce an extended source feature in the smoothed map than in the original map.

MULTI The Multi-scale CLEAN algorithm has also been designed to improve the performance of CLEAN for extended sources. It is a straightforward extension of CLEAN that models the sky brightness by the summation of blobs of emission having different size scales. It is equivalent

to simultaneously deconvolve images obtained with different synthesized beams derived from the highest resolution one by convolution kernels. This algorithms works simultaneously in a range of specified scales. Multi-scale CLEAN can produce good images with a loop gain of 0.5 or even higher.

The implementation of Multi-scale CLEAN in IMAGER slightly differs from that of CASA . It is less optimized in terms of speed, but uses a better convergence scheme in which the scale chosen at each iteration is the one with best signal to noise ratio. Accordingly, it is more stable. Only 3 scales are used so far in IMAGER, with a size ratio controlled by CLEAN_SMOOTH.

6.5.4 Implementation and typical use

Deconvolution parameters are controlled by CLEAN_* variables. Progress has been made on automatic guess for Cleaning parameters. The table below presents the current naming scheme, with previous or equivalent names mentioned in parentheses, since these names were (or are still) used by several older packages such as MAPPING, AIPS or CASA. The equivalent "old" names (mentioned in Upper case below) will remain as aliases, while those mentioned in mixed case have disappeared as they were seldom used before.

CLEAN_STOP	Stopping value and its Unit
CLEAN_ARES	Absolute residual (ARES)
CLEAN_FRES	Fractional residual (FRES)
CLEAN_GAIN	Loop gain (GAIN)
CLEAN_INFLATE	Inflation factor allowed to display MultiScale clean components
CLEAN_METHOD	Cleaning Method (METHOD)
CLEAN_NCYCLE	Maximum number of Major cycles (Nmajor)
CLEAN_NITER	Maximum number of iterations (NITER)
CLEAN_NGOAL	A number of components for ALMA joint deconvolution only (Ngoal)
CLEAN_NKEEP	Number of iterations used to check convergence (see below)
CLEAN_POSITIVE	Minimum number of positive Clean components
CLEAN_RATIO	Ratio for Dual Resolution clean (Ratio)
CLEAN_SEARCH	Minimum primary beam threshold for searching (Search_W)
CLEAN_SMOOTH	Smoothing factor for Multi Scale Clean (Smooth)
CLEAN_SPEEDY	Speeding factor for Clark (Spexp)
CLEAN_TRUNCATE	Minimum primary beam threshold for restoring (Restore_W)
CLEAN_WORRY	"Worry" factor for Clark (Worry)

Implementation In IMAGER, the variants of the CLEAN algorithms discussed above are coded as the following commands: HOGBOM, CLARK, MX, SDI, MULTI and MRC. All those commands work on two internal buffers containing the dirty beam and dirty image. Both buffers are created directly from *uv* table through the UV_MAP command, or they can be loaded from files through the READ BEAM and READ DIRTY commands. The behavior of those commands is controlled through the following common SIC variables:

Iterative search

CLEAN_POSITIVE Number of positive clean components to be found before enabling the search for negative components. Default is 0.

CLEAN_GAIN Loop gain. Default is 0.2, good compromise between stability and speed.

Stopping criteria

CLEAN_STOP Compact way to specify the stopping criterium. Defaut is currently empty, but might be set to 2 SIGMA as a good general default.

CLEAN_NITER Maximum number of clean components. Default is 0.

CLEAN_FRES Maximum amplitude of the absolute value of the residual image. This maximum is expressed as a fraction of the peak intensity of the dirty image. Default value is 0.

CLEAN_ARES Maximum amplitude of the absolute value of the residual image. This maximum is expressed in the image units (Jy/Beam). Default value is 0.

CLEAN_NKEEP Minimum number of Clean components be- fore testing if Cleaning has converged. Default value is 70.

Support

BLC **and** TRC Bottom Left Corner and Top Right Corner of a square support in pixel units. Default is 0, which means using only the inner quarter if no other support is defined.

SUPPORT A command that defines the support where to search for clean components. The support can be a Mask, or a Polygon. For a Polygon, the definition can be interactive, using the GREG cursor. This definition can be stored in a file through the WRITE SUPPORT command and read back in memory from the file with the SUPPORT command. The polygon support definition is stored in the SUPPORT% structure. Command SUPPORT /MASK instructs IMAGER to use the Mask instead of the polygon for the Clean support.

MASK Command MASK is used to define a Mask-like support. This can be interactive, or automatic using a thresholding technique in command MASK THRESHOLD. The computed Mask can be saved by command WRITE MASK. The Mask can also be read by command READ MASK. Command MASK USE is equivalent to command SUPPORT /MASK, and instructs IMAGER to use the Mask instead of the polygon for the Clean support.

Clean beam parameters

BEAM_SIZE is a 3-element array that gives the FWHM size of the major and minor axes (in arcsec) and position angle (in degree) of the Gaussian used to restore the clean image from the clean component list. Default is all parameters at 0, meaning use the fit of the main lobe of the dirty image. Changing the default value of those parameters is dangerous. The beam size effectively used is available in BEAM_FITTED.

Other variables control specific aspects of a subclass of the CLEAN algorithm:

CLEAN_NCYCLE Maximum number of major cycles in all algorithms using this notion (CLARK, MX, SDI). Default is 50.

BEAM_PATCH Size (in pixel units) of the dirty beam used to deconvolve the residual image in minor cycles. It is used in CLARK and MRC algorithms only. Default value is 0. This is for development only.

CLEAN_SMOOTH Smoothing factor between different scales in the MULTISCALE methods. Default value is sqrt(3).

CLEAN_RATIO Smoothing factor between different scales in the MRC method. Default value is 0, for which the code automatically derives the best power of 2 adequate for the current problem.

6.5.5 Restoring step: UV_RESTORE

CLEAN (except for MRC) has three outputs: the CLEAN image itself, the RESIDUAL image, and the list of point sources (the Clean Components) that reproduce the observed visibilities, the CCT table.

An aesthetically better results, with better noise properties, can be obtained after CLEAN by removing the Clean Components from the measured visibilities, and re-imaging this to produce the RESIDUAL image. Aliasing at map edges is then minimized, as it only concerns noise if the deconvolution was reasonable. As mentioned previously, this is implicitely done when using MX instead of CLEAN. It can also be done after CLEAN using command UV_RESTORE.

This step can also be used to properly scale the residuals when the synthesized beam is not well fit by a Gaussian, or when all channels do not share the same beam (see Section 4.5). This is done through the use of the so-called JvM factor. This factor, introduced in Jorsater & van Moorsel (1995), estimates the ratio of clean beam area to dirty beam area, and is stored in variable BEAM_JVM and (on a per beam basis) BEAM_VALUES[4] which are computed by FIT /JVM_FACTOR. Residuals are multiplied by this factor (on a channel per channel basis if needed), allowing to first order to scale the residuals to the same unit as the Cleaned data, i.e. Jy/beam area.

In general, this number is fairly close to 1. However, when several observing configurations have been merged together, the synthesized beam often exhibits a central peak over a larger plateau, because of the higher weights of short baselines. Robust weighting limits the effect, but is not able to suppress it totally. In such cases, the JvM factor can be of order 0.5 to 0.6.

- **Caveat: This scaling is only meaningful to first order for (reasonably) extended sources and well-behaved beams. It should NOT be used to analyze data for point sources.**

UV_RESTORE uses the JvM factor if FIT /JVM_FACTOR has been used before. It also uses the mean Clean beam geometry defined by BEAM_FITTED.

6.5.6 Typical deconvolution session

```
 1 read beam demo
 2 read dirty demo
 3 clean ?
 4 hogbom /flux 0 1
 5 show residual
 6 show clean
 7 write clean demo
 8 let name demo
 9 show noise
10 let ares 0.5*noise
11 clean ?
12 hogbom /flux 0 1
13 let niter 2000
14 clean ?
15 hogbom /flux 0 1
16 show residual
17 show clean
18 for iplane 1 to 10
```

```
19    show clean iplane
20    support
21    hogbom iplane /flux 0 1
22    write support "demo-"'iplane'
23 next iplane
24 show residual
25 view cct
26 view clean
27 write residual demo
28 write clean demo
29 write cct demo
```

Comments:

Steps 1-2 Read dirty beam and dirty image from the `demo.beam` and `demo.lmv` files. Those steps are not needed if the dirty beam and image are already stored in the internal buffer, *i.e.* if you have imaged the *uv* table just before in the same IMAGER session.

Steps 3-6 Print the current state of the control parameters, deconvolve the dirty image using the `HOGBOM` algorithm (step 3) and look at the results (residual and clean images). The `/flux 0 1` option pop-up the visualization of the cumulative flux deconvolved as the clean components are found.

Steps 8-12 Estimate the empirical noise through the `SHOW NOISE` command after this first deconvolution and set the `ares` stopping criterion accordingly. Check that the new value of `ares` has been correctly set (step 11) and restart deconvolution.

Steps 13-17 Increase the number of clean components as the previous deconvolution stopped before the residual image reached the `ares` value. Restart deconvolution and look at results.

Steps 18-23 Attempt to improve deconvolution by definition of a support per plane and deconvolve this plane accordingly. The support is stored in a file for further re-use. The deconvolution results are then displayed.

Steps 24-26 Display the residual images, visualize the cumulative flux as a function of the clean component number and visualize the clean spectra cube in an interactive way.

Steps 27-29 Write residual image, clean image and clean component list in `demo.lmv-res`, `demo.lmv-clean` and `demo.cct` files for later use.

Typical deconvolution session using other `CLEAN` algorithm would look very similar. The main difference would be the possible tuning of other control parameters. A deconvolution session using `MX` would start differently as the imaging and deconvolution are done in the same step:

```
 1 read uv demo
 2 mx ?
 3 mx /flux 0 1
 4 show residual
 5 show clean
 6 write * demo
%  6 write beam demo
%  7 write dirty demo
%  8 write clean demo
```

```
%       9 write residual demo
%      10 write cct demo
```

Comments:

Step 1 Read the `demo.uvt` *uv* table in an internal buffer.

Step 2 Check current state of the variables that control the imaging and deconvolution.

Steps 3-5 Deconvolve and look at the results.

Steps 6-10 Write all the internal buffers on disk files.

All the tuning of the typical imaging and deconvolution sessions could be used in this MX session although they are not repeated here.

6.6 Practical advices

6.6.1 Comparison of deconvolution algorithms

HOGBOM is the basic CLEAN algorithm. It is very robust, but somewhat slow although this is partially compensated by good parallel programming. CLARK introduces a faster strategy for the search and removal of clean component. However, it may converge more slowly and can even be instable when dirty sidelobes are high, or the phase noise still significant. MX cleans the largest region of the dirty map because the source removal happens in the *uv* plane. For the same map size, it is slower than CLARK because of the repeated imaging step, but smaller image sizes can be used. It shares some of the CLARK instabilities because it uses the same search strategy, but the removal strategy counteracts this. MX can be very slow for large number of visibilities.

HOGBOM, CLARK and MX may introduce artifacts as parallel stripes in the clean map when dealing with smooth, extended structures. SDI, MRC and MULTI introduce a different (in principle better) handling of those extended sources. SDI is a rough attempt and requires a good prior of the support to work correctly. MRC and MULTI introduce the notion of cleaning at different spatial scales. MRC has unfortunately no notion of Clean Components. MULTI is the most advanced tool, but is significantly slower than any other method.

6.6.2 A few (obvious) practical recommendations

Map size Make an image about twice the size of the primary beam (*e.g.* $2 \times 55''$ at 90 GHz and $2 \times 22''$ at 230 GHz for NOEMA antenna) to ensure that all the area of the primary beam (inner quarter of the dirty map) will be cleaned whatever the deconvolution algorithm is used. However, avoid making a too large dirty image because the CLEAN algorithms will then try to deconvolve region outside the primary beam area where the noise dominates.

Support Start your first deconvolution *without* any support to avoid biasing your clean image. If the source is spatially bound, you can define a support around the source and restart the deconvolution with this *a priori* information. Be careful to check that there is no low signal-to-noise extended structure that could contain a large fraction of the source flux outside your support... Avoid defining a support too close to the natural edges of your source. Indeed, deconvolving noisy regions around your source is advisable because it ensures that you do not bias your deconvolution too much.

Stopping criterion Choose the right stopping criterion.

Use the stability CLEAN_NKEEP parameter preferentially, combined with a

CLEAN_STOP = r SIGMA with r between 1 or 2. That keeps CLEAN_ARES, CLEAN_FRES and CLEAN_NITER to zero. If it does not work, then

- Estimate an empirical noise on your first deconvolved cleaned image with STATISTIC, CLEAN, or SHOW NOISE.
- If this empirical noise value is similar to the value computed from the visibility weights (this noise value is one of the outputs of the UV_MAP command), your observation is not dynamic range limited. Apart from using a user-specified support (SUPPORT or MASK) there is not much you can do to improve your result,
- If not, you are dynamic range limited. You may use CLEAN_STOP = r % where r depends on the dynamic range as stopping criterium. Alternatively, you can select the effective noise level as the true Sigma, CLEAN_STOP = r 'clean%rms' with r being 1 to 2. The problem in such cases is that the noise level may be channel dependent, an issue that is not well handled.

Convergence checks Ensure that your deconvolution converged by checking that:

- The cumulative flux as a function of the number of clean component has reached a roughly constant level (use /FLUX option of the deconvolution commands to see this curves, or SHOW CCT or VIEW CCT).
- The residuals are similar or smaller in the source region (where Clean components were found) compared to elsewhere.

If not, change the values of the stopping criterion, whichever you used.

Deconvolution methods If you want a robust result in all cases, start with HOGBOM. If you prefer obtaining a quick result, use CLARK but you then first need to check that the dirty sidelobes are not too large on the dirty beam. If you obtain stripes in your Clean image:

- First check that your deconvolution converged.
- Then check that there is no spurious visibilities that should be flagged : use command UV_FLAG as a last resort.
- If it is clear that you have an extended source structure, you should first ask yourself whether you are in the wide-field imaging case and act accordingly (see next chapter). Else you can try a CLEAN variant which better deals with cases that implies a large spatial dynamic. This is rare at NOEMA, but may happen with ALMA.

Outside help Always consult an expert until you become one.

7 Wide-field imaging and deconvolution

We are often asked why the wide-field imaging and deconvolution steps are more difficult than their equivalent for single-field. The main answer is that doing wide-field observations with an interferometer is kind of paradoxical. Indeed, (sub)millimeter interferometers are before all tuned to get the best possible spatial resolution. A natural consequence is the lack of measurement of the low spatial frequencies which are extremely important in wide-field observations. Hence the paradox.

Progress in the design (ALMA was designed with wide-field imaging as a main goal) or in performances (NOEMA and the 30-m) has led to wide-field images being now customary. The tools have become much simpler and user-friendly (see below !) but because of its paradoxical nature, wide-field imaging with an interferometer implies a knowledgeable use of those tools.

7.1 In a nutshell

```
 1  read uv gag_demo:demo-mosaic-imager.uvt
(2)  read single gag_demo:demo-single-imager.tab
(3)  uv_short
 4  uv_map
 5  clean
(6) fit /jvm;  uv_restore
 7  show sky
 8  write * MyDemo
```

1. Read your *uv* data
2. Optionally, read your single-dish data
3. Optionally, merge it with with the *uv* data set, by using the single-dish data to provide short spacings for the interferometer data.
4. Image as usual
5. deconvolve as usual
6. Optionally, improve result by refining the residuals
7. look at it
8. Save the result.

You are done. If the *uv* data set is a **Mosaic** data set, it works as if it is a single-field, except that the result appears as the SKY brightness distribution, because it is always corrected for primary beams.

Now, if the result is crazy, do not blame the software. Rather read carefully the information below: **Mosaics** and UV_SHORT are simple to use, but can be tricky to use well !...

7.2 General considerations about wide-field imaging

The measurement equation for a millimeter interferometer is to a good approximation (after calibration)

$$V(u,v) = \mathrm{FT}\left\{B_{\mathrm{primary}}.I_{\mathrm{source}}\right\}(u,v) + N \tag{7}$$

where $\mathrm{FT}\left\{F\right\}(f)$ is the bi-dimensional Fourier transform of the function F taken at the spatial frequency f, I_{source} the sky intensity, B_{primary} the primary beam of the interferometer (*i.e.* a

Gaussian of FWHM the natural resolution of the single-dish antenna composing the interferometer), N some thermal noise and $V(u,v)$ the calibrated visibility at the spatial frequency u,v. The product of the sky intensity by the primary beam, which "quickly" decreases to zero, implies that an interferometer looking at a particular direction of the sky will have its field-of-view limited by the size of the primary beam.

To image a field-of-view larger than the primary beam size, the antennae of an interferometer will be successively pointed in different directions of the sky typically separated by half the size of the primary beam. This process is called mosaicing and the result requires specific imaging and deconvolution steps. Another possibility is to acquire data as the interferometer antenna continuously slew through a portion of the sky. This second observing mode is called interferometric On-The-Fly (OTF). While mosaicing is standard at NOEMA (see section 7.3), some efforts are currently done to commission the OTF observing mode.

Mosaicing and OTF clearly belongs to wide-field imaging. However considerations about wide-field imaging start as soon as the size of the source is larger than about 1/3 to 1/2 of the interferometer primary beam. Indeed, a multiplicative interferometer (*e.g.* all interferometer in the (sub)mm range) is a bandpass instrument, *i.e.* it filters not only the large spatial frequencies (this is the effect of the finite resolution of the instrument) but also the small spatial frequencies (all the frequencies smaller than typically the diameter of the interferometer antennas). An important consequence is that a multiplicative interferometer do *not* measure the total flux of the observed source. This derives immediately from the following property of the Fourier Transform: The Fourier transform of a function evaluated at zero spacial frequency is equal to the integral of your function. Adapting this to our notation, this gives

$$V(u=0,v=0) \overset{\text{FT}}{\rightleftharpoons} \sum_{ij \,\in\, \text{image}} \{B_{\text{primary}}.I_{\text{source}}\}_{ij} \,. \tag{8}$$

i.e. the visibility at the center of the uv plane is the total intensity of the source. As a multiplicative interferometer filters out in particular $V(u=0,v=0)$, the information about the total flux of the observed source is lost. In summary, a multiplicative interferometer only gives information about the way the flux of the source is distributed in the spatial frequencies larger than the primary beam but no information about the total flux.

Deconvolution algorithms use, in one way or another, the information of the flux at the smallest *measured* spatial frequencies to extrapolate the total flux of the source. This works correctly when the size of the source is small compared to the primary beam of the interferometer. The extreme case is a point source at the phase center for which the amplitude of all the visibilities is constant and equal to the total flux of the source: Extrapolation is then exact. However, the larger the size of the source, the worst the extrapolation, which then underestimates the total source flux. This is the well-known problem of the missing flux that observers sometimes note when comparing the flux of the source delivered by a mm interferometer with the flux observed with a single-dish antenna. The transition between right and wrong extrapolation is not well documented. It depends on the repartition of the flux with spatial frequencies but also of the signal-to-noise ratio of the measured spatial frequencies. It is often agreed that the transition happens for sizes between 1/3 and 1/2 of the interferometer primary beam. For larger source size, information from a single-dish telescope is needed to fill in the missing information and to thus obtain a correct result. This is the object of section 8.

7.3 Mosaicing

7.3.1 Observations and processing

In a single-field observation, an interferometer tracks a particular direction of the sky, named the phase center. The portion of the sky which can be image around this direction is directly linked to the size of the primary beam. The easiest way to image field-of-view larger than the primary beam size is to track one direction of the sky after another until the desired field-of-view is filled with small images made around many different tracking directions. This observing mode is called mosaicing and the tracked observations which constitute the mosaic are called fields.

There are many constraints to optimize mosaicing.

Nyquist sampling of the mosaic field-of-view and mosaic pattern The mosaic field-of-view must at least be Nyquist-sampled to obtain a reliable image. Each observed field can produce a reliable image of the same shape than the primary beam, *i.e.* a circular Gaussian (This assumes that the short-spacing problem has been solved). Nyquist sampling thus implies that the mosaic fields follow an hexagonal compact pattern as this ensures a distance between all neighboring fields of half the primary beam size. When the total observing time is fixed, Nyquist sampling is the best compromise between sensitivity and total field-of-view. Indeed, the distance between neighboring fields could be less (in which case the mosaic would be oversampled) than half the primary beam size. In this case, the sensitivity on each pixel of the final image would increase with the share of the time spent to observe this direction.

Uniform imaging properties and quick loop around the fields Getting uniform imaging properties is a desirable feature in the final result. This implies that a *uv* coverage and a noise level as uniform as possible among the different fields. Quickly looping around the different fields is the easiest way to reach this goal. However, dead time to travel from one field to another must almost be minimized. At NOEMA, the compromise is to pause at least 1 minute on each field and to try to loop over all the fields between two calibrations every 20 minutes. Hence, mosaic done in a single observing run is made of at most 20 fields. Larger mosaic must be observed by group of fields in different observing runs.

7.3.2 Imaging

When combining together (dirty or clean) images, it is important to correct the primary beam attenuation to avoid modulation of the signal in the combined image. If we forget for the moment the dirty beam convolution, the images associated to each fields are noisy measurement of the same quantity (the sky brightness distribution) weighted by the primary beam. The best estimation of the measured quantity is thus given by the least mean square formula

$$M(\alpha,\delta) = \frac{\sum_i \frac{B_i(\alpha,\delta)}{\sigma_i^2} F_i(\alpha,\delta)}{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}}, \tag{9}$$

where $M(\alpha,\delta)$ is the brightness of the dirty/cleaned mosaic image in the direction (α,δ), B_i are the response functions of the primary antenna beams in the tracking direction of field i, F_i are the brightness distributions of the individual dirty/cleaned maps, and σ_i are the corresponding noise values. As may be seen on this equation, the intensity distribution of the mosaic is corrected for

primary beam attenuation. This implies that noise is inhomogeneous. Indeed, if $N(\alpha,\delta)$ is the noise distribution and $\sigma(\alpha,\beta)$ is its standard deviation in the direction (α,β), we have

$$N(\alpha,\delta) = \frac{\sum_i \frac{B_i(\alpha,\delta)}{\sigma_i^2} N_i(\alpha,\delta)}{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}}, \tag{10}$$

and

$$\sigma(\alpha,\delta) = \frac{\sqrt{\sum_i \frac{B_i(\alpha,\delta)}{\sigma_i^2}}}{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}} = \frac{1}{\sqrt{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}}} \tag{11}$$

Not only, the noise strongly increases near the edges of the mosaic field-of-view. But also, the center of each field is contaminated by increased noise level coming from the external regions of the neighboring fields. Indeed, the noise corrected for the primary beam attenuation is largely increasing where the primary beam is going to zero. To limit these effects, both the primary beams used in the above formula and the resulting mosaic are truncated.

7.3.3 Deconvolution

Standard `CLEAN` algorithms must be slightly modified to work on a dirty mosaics. Indeed, the use of truncated primary beam in the above equations is only a first order measure to avoid noise artifacts. However, the noise level still increases at the edges of the mosaic, implying that at some point the `CLEAN` algorithms will confuse noise peaks at the mosaic edges with true signal. To avoid this, the iterative search is made on a signal-to-noise image $M(\alpha,\beta)/\sigma(\alpha,\beta)$ instead of the residual image. At the restoration step, the clean component list is used to produce the residual map and clean map. Nothing particular is done with the remaining signal-to-noise image.

7.3.4 Typical use

The processing of mosaics for NOEMA is essentially similar to that of single fields. There are only two small changes

Creation of *uv* table A mosaic UV table should be created using the `/MOSAIC` option of command `TABLE` in CLIC.

Imaging is done through `UV_MAP` as for a single field. However, the process is different. The command takes into account the various fields, the primary beams, and select an optimum projection center (phase center). The later may need to be specified by the user using the `MAP_CENTER` string, or as argument to `UV_MAP`

Deconvolution is also similar, but not all algorithms are available: `MX, SDI` and `MRC` do not work for mosaics. The change of behavior of the `CLEAN` algorithms is visualized through the change of prompt from `IMAGER>` to `MOSAIC>`. Two additional variables are used for mosaic deconvolution

`CLEAN_SEARCH` The minimum fraction of a primary beam below which no Clean component is searched for.

`CLEAN_TRUNCATE` The minimum fraction of a primary beam cumulated response below which no image restoration is performed. Below this threshold, the Clean image is blanked.

Finally, note that the mosaic deconvolution produces sky brightness images (SKY variable) while single-field deconvolution produces images attenuated by the primary beam.

IMAGER makes no specific assumption about the *uv* coverage of individual fields. However, it uses a single Clean beam in the deconvolution, so Mosaics where fields have widely different *uv* coverage will not conserve flux properly. Mosaicing deconvolution will work better if all fields are equivalent in *uv* coverage and noise level.

A mosaicing session would thus just be like a single-field imaging:

```
1 read uv gag_demo:demo-mosaic
2 uv_map
3 hogbom /flux 0 10
4 fit /jvm
5 uv_restore
6 show sky
7 write * demo
```

Comments:

Step 1 Read the UV table

Step 2 Image the mosaic

Step 3 Deconvolve

Step 4-5 Optimize residuals (see Sec.7.4)

Step 6 Look at the result. The result is in SKY.

Steps 7 Save the result SKY, and the intermediate files (BEAM, DIRTY, PRIMARY).

However, mosaics can be huge and require more memory than available (see Section 14) To overcome this issue, IMAGER provides the script `@ image-mosaic` that splits the UV_MAP step into slices that fit into memory. The user can then read the resulting images and deconvolve them, eventually by blocks of channels if needed (in particular when a uv_restore step is desired) using a similar logic.

7.4 Mosaic and JvM factor

By construction, the various fields covered in a mosaic have different *uv* coverages, and thus different synthesized beams. Furthermore, mosaic covers wider fields, so the impact of frequency variable angular resolution is more important than in single fields.

Yet, the mosaic is restored with a unique Clean beam. This situation leads to a mismatch between the use Clean beam and the dirty beams in some or even most of the fields. Furthermore, because Mosaics often use short spacings (see Section 8), the dirty beams often present a wide "plateau" that is not well fit by a Gaussian Clean beam. These are situations which are prone to lead to inaccurate large scale flux restoration, because the Clean beam and dirty beams area do not match.

To minimize these issues, it is recommended in IMAGER to use the FIT /JVM_FACTOR and UV_RESTORE commands that correct this mismatch to first order by re-scaling the residual by the Clean to Dirty beam ratio, the so-called JvM factor (Jorsater & van Moorsel (1995)). Only do so once you are satisfied with your deconvolution, as UV_RESTORE is often time consuming on Mosaics.

8 Short and Zero spacings

8.1 In a nutshell

```
1 read uv gag_demo:demo-mosaic-imager.uvt
2 read single gag_demo:demo-single-imager.tab
3 uv_short
4 uv_map; clean
5 view clean
6 write * MyDemo
```

1. Read your *uv* data
2. read your single-dish data
3. Merge it with with the *uv* data set,
4. Image and deconvolve as usual
5. check the result. You can use `CLEAN` as argument to SHOW or VIEW: IMAGER will automatically fall back to `SKY` if needed (and vice versa).
6. Save it.

You are done. This works for mosaics as well as single fields.

But, again, if the result are crazy, do not blame the software. Rather read carefully the information below: UV_SHORT is simple to use, but can be tricky to use well !...

8.2 Principle

Let us note D the diameter of the single-dish antenna ($D = 30$m for the IRAM-30m telescope) used to produce the short-spacing information and d the diameter of the interferometer antennas ($d = 15$m for NOEMA). We already mentioned that a multiplicative interferometer filters out all the spatial frequencies smaller than $\sim d$ meters. When this information is needed to get reliable results, the source should also be observed with a single-dish antenna to produce the missing information. The single-dish antenna furnishes information about all spatial frequencies up to $\sim D$ meters (but this information is weighted by the single-dish beam shape, *i.e.* high frequencies are measured with a worse signal-to-noise ratio than low frequencies). To recover all the information at spatial frequencies smaller than d meters, the diameter of single-dish antenna must be larger or equal to the diameter of the interferometer antennae: $D \geq d$.

8.3 Algorithms to merge single-dish and interferometer information

The measurement equations of a single-dish and an interferometer are quite different from each other. Indeed, the measurement equation of a single-dish antenna is

$$I^{\rm sd}_{\rm meas} = B_{\rm sd} \star I_{\rm source} + N, \tag{12}$$

i.e. the measured intensity ($I^{\rm sd}_{\rm meas}$) is the convolution of the source intensity distribution ($I_{\rm source}$) by the single-dish beam ($B_{\rm sd}$) plus some thermal noise, while the measurement equation of an interferometer can be rewritten as

$$I^{\rm id}_{\rm meas} = B_{\rm dirty} \star \{B_{\rm primary}.I_{\rm source}\} + N, \tag{13}$$

i.e. the measured intensity ($I^{\rm id}_{\rm meas}$) is the convolution of the source intensity distribution times the primary beam ($B_{\rm primary}.I_{\rm source}$) by the dirty beam ($B_{\rm dirty}$) plus some thermal noise. $B_{\rm sd}$ has very similar properties than $B_{\rm primary}$ and very different properties than $B_{\rm dirty}$. In radioastronomy, $B_{\rm sd}$ and $B_{\rm primary}$ both have (approximately) Gaussian shapes. Moreover, the fact that we will use the single-dish information to produce the short-spacing information filtered out by the interferometer implies that $B_{\rm sd}$ and $B_{\rm primary}$ have similar full width at half maximum. Now, $B_{\rm dirty}$ is quite far from a Gaussian shape with the current generation of interferometer (in particular, it has large sidelobes) and the primary side lobe of $B_{\rm dirty}$ has a full width at half maximum close to the interferometer resolution, *i.e.* much smaller than the FWHM of $B_{\rm sd}$.

Merging both kinds of information obtained from such different measurement equations thus asks for a dedicated processing. There are mainly two families of short-spacing processing: the **hybridization** and the **pseudo-visibility** techniques.

8.3.1 Hybridization technique ("feathering")

In this family, most of the processing is done on the interferometric data alone. Indeed, the interferometric data is deconvolved and corrected for the primary beam contribution to obtain

$$I^{\rm id}_{\rm sky} = B_{\rm clean} \star I_{\rm source} + N', \tag{14}$$

where $B_{\rm clean}$ is a Gaussian of FWHM equal to the interferometer resolution and N' is some thermal noise corrected for the primary beam contribution. Two main facts are hidden in this formulation: 1) the field-of-view of the observation is obviously limited to the observed portion of the sky and 2) more importantly, the lack of short-spacings has not yet been overcome and a better formulation would be

$$I^{\rm id}_{\rm sky} = \text{Highpass-filter}\,\{B_{\rm clean} \star I_{\rm source}\} + N'. \tag{15}$$

The simplicity of equation 14 is thus slightly misleading but we will keep it for the sake of simplicity. The hybridization method consists in combining two images ($I^{\rm sd}_{\rm meas}$ and $I^{\rm id}_{\rm clean}$) in the uv plane.

1. Both images are first spatially regridded on the same fine grid.
2. The FFT of those two images are computed, and linearly combined by selecting the low spatial frequencies from FFT($I^{\rm sd}_{\rm meas}$) and the high spatial frequencies from FFT($I^{\rm id}_{\rm sky}$).

   $$\text{FFT}(uv) = f(uv)\text{FFT}(I^{\rm sd}_{\rm meas} + (1 - f(uv))\text{FFT}(I^{\rm id}_{\rm sky}$$

   The transition $f(uv)$between low and high spatial frequency is selected to use the best regions of the uv plane in both images.
3. The result is FFTed back to the image plane to produce a final,unique image, which takes into account both single-dish and interferometric information.

The method has the following free parameters: the transition radius and the detailed shape of that transition. To avoid discontinuity, the transition shape is chosen to be reasonably smooth. When the low resolution image is provided by a single-dish, the best signal-to-noise combination is obtained using a function $f(uv)$ that is Fourier transform of the single-dish beam. However, that is only optimal if the noise in this image is small enough and no other instrumental effect

(such as pointing errors or baseline ripples) affect the data. So, $f(uv)$ can also be chosen arbitrarily. The spatial frequency of transition is selected close to the smallest spatial frequency reliably measured by the interferometer (*e.g.* about 18 m for NOEMA), and/or the largest spatial frequency measured by the low resolution image (*e.g.* about 20 m for data taken with the IRAM 30-m telescope). For combining ACA and ALMA data, this would be 15 m, and of ACA and 12-m single dish, about 9 m.

8.3.2 Pseudo-visibility technique

General description In this family, the single-dish information is heavily processed before merging with the interferometric information. The basic idea is to produce from the single-dish observations pseudo-visibilities similar to the ones that would be produced by the interferometer if they were not filtered out.

1. The Single-Dish measurements are re-gridded and then FFTed into the *uv* plane.
2. The data are deconvolved of the single-dish beam ($B_{\rm sd}$) convolution by division by its Fourier Transform (truncated to the antenna diameter).
3. The data are FFTed back to the image plane and multiplied by the interferometer primary beam, $B_{\rm primary}$.
4. The result is FFTed again in the *uv* plane where the visibilities are sampled on a regular grid.
5. In the case of a mosaic, the two last operations are performed for each pointing center.

Using the properties of the Fourier transform, we can rewrite the measurement equation of an interferometer as

$$V(u,v) = \{\mathrm{FT}(B_{\rm primary}) \star \mathrm{FT}(I_{\rm source})\}\,(u,v) + N. \tag{16}$$

This equation means that the visibility measured by an interferometer at the spatial frequency (u,v) is the convolution of the Fourier transform of the source intensity distribution by the Fourier transform of the primary beam. Hence, to get pseudo-visibilities truly consistent with interferometric visibilities, we must be able to reliably compute the convolution by the Fourier transform of the primary beam. This implies that we can compute pseudo-visibilities only for spatial frequencies lower than *D-d*. The use of the IRAM-30m to produce the short-spacing information of the NOEMA is thus ideal as it enables to recover pseudo-visibilities up to 15 m (=30 m-15 m). Once the pseudo-visibilities have been computed, they are merged with the interferometric visibilities and standard imaging and deconvolution are then applied to the merged data set.

Single-dish vs interferometer weight In all cases involving short spacings, the relative weight of the single dish data to interferometer data is critical. Within the restrictions imposed by the noise level, this relative weight is a free parameter. It is all the more important that the Fourier transform of the *uv* plane density of weights is the dirty beam, a key parameter of the deconvolution. The general goal is to have a dirty beam as close as possible to a Gaussian. As the Fourier transform of a Gaussian is a Gaussian, we search for obtaining a *uv* plane density of weights as close as possible to a Gaussian. In general, the short spacing frequencies are small compared to the largest spatial frequency measured by an interferometer. This implies we can use the linear approximation of a Gaussian, *i.e.* a constant, in the range of frequencies used for the short spacing processing. We thus end up with the need to match (as far as possible) the

Single-Dish and interferometric densities of weights in the *uv* plane. In practice, we compute the density of weights from the single-dish in a *uv* circle of radius 1.25 d and we match it to the averaged density of weights from the interferometer in a *uv* ring between 1.25 and 2.5 d. Experience shows that this gives the right order of magnitude for the relative weight and that a large range of relative weight around this value gives very similar final results.

When processing IRAM-30m data to combine to NOEMA data, this criterion implies a large down-weighting of the IRAM-30m data which may make think that too much observing time was used at the IRAM-30m data. However, just using the above criterion to determine the observing time needed at the IRAM-30m would in general lead to very low signal-to-noise ratio of the single-dish map. In such a case, it is very difficult to detect problems which would translate in strong artifacts in the IRAM-30m + NOEMA combination. We recommend to ask for enough IRAM-30m time to get a *median* signal-to-noise ratio of 5 on the single-dish map. This ratio should be achieved for all velocity channels of interest (which may include the line wings).

8.3.3 Comparison

The simplicity of the hybridization technique is its main advantage. It is simple to understand and simple to implement. However, this method works badly in practice because it is truly difficult to obtain a reliable deconvolution of interferometric data alone when short-spacing information is important. An interferometer is a spatial pass-band filter, filtering in particular the zero spacing. This implies that the total flux in the dirty image is zero (*i.e.* as much negative as positive flux in the dirty image) but that the dirty beam integral is also zero (*i.e.* as much negative as positive sidelobes). Adding the short-spacing information (and in particular the zero spacing) through the pseudo-visibility method, we enforces the positivity of the dirty image total flux and of the dirty beam integral. It is well-known that trying to deconvolve a mosaic built only with interferometric data is quite difficult. It almost always requires the definition of support where the `CLEAN` algorithms can search for clean components with the clear risk to bias the final result. In contrast, adding the short-spacing information through pseudo-visibilities enables an almost straightforward `CLEAN` deconvolution *without* the need of any support.

For the sake of illustration, let us assume an intensity distribution made of a large scale structure (*e.g.* a smoothly varying intensity) superimposed with a small scale distribution both in emission and absorption. An interferometer will filter out the smooth distribution. If there is no additional zero spacing information, the smooth distribution is completely lost with the important consequence that the final deconvoled image will have positive (emission) and negative (absorption) structures. Trying to reproduce both negative and positive structures is one of the most difficult task for deconvolution algorithms. In addition, the presence of large negative structures create instabilities in the algorithms of the `CLEAN` family (because it is difficult to distinguish between negative absorption structures and negative sidelobes of emission structures). Only the definition of support around positive emission peaks may succeed to stabilize the `CLEAN` algorithms with the drawback of biasing the result.

Both kind of algorithms are implemented in IMAGER, under commands `FEATHER` for the hybridization and `UV_SHORT` for the pseudo-visibilities. However, we strongly recommend to use the pseudo-visibility algorithm. Pety et al. (2001a,b,c) showed through simulations that 1) the pseudo-visibility algorithm implemented in GILDAS enable extremely reliable results (fidelities of a few thousands) on ideal observations and 2) the accuracy of the wide-field imaging is limited by pointing errors, amplitude calibration errors and atmospheric phase noise (and not by the used algorithms), even for ALMA.

8.4 Hybridization technique and ALMA

A special case where Hybridization can be extremely useful is that of ALMA observations involving the main 12-m array, the ACA and single-dish data with the 12-m antennas. These can be combined in several ways.

For example, the deconvolved mosaic made (using the pseudo-visibility method) from ACA and the single-dish data can be used as a single-dish image, providing short spacings that complete (either as pseudo-visibilities, or by feathering) the 12-m data.

In most circumstances, an optimal result is obtained by hybridizing Cleaned images produced from the mosaics obtained by combining (using the pseudo-visibility method) ACA and Single-dish data as short spacings, with another mosaic obtained with the 12-m data and the Single-dish data (as zero spacing in this case) together.

The deconvolution of each mosaic is stabilized by the addition of the 12-m single-dish zero or short spacings, and these good mosaics are then merged in an optimal way by using the best region of the *uv* plane that they sample, typically using a transition radius of 15 to 18 m.

See command FEATHER for details.

8.5 The Zero spacing: an important subset

An important subset of the pseudo-visibility method is the production of only the zero spacing. Indeed, the zero spacing is just the total flux of the observed field-of-view. Hence, if the observed field-of-view is small enough to fit in the single-dish beam (this is in particular always the case if $D = d$), a single spectrum observed with the single-dish telescope in the direction of the interferometer phase center may be used as zero spacing, only a scaling from Kelvin to Jansky is needed. This is the poor man solution as only part of the short spacing information is recovered by this technique.

8.6 Short Spacings in practice: command UV_SHORT

Our algorithm to produce the short-spacing information is coded in the UV_SHORT command. UV_SHORT will **add** the short spacing information to the current *uv* table (read by command READ UV and optionally transformed by further UV_... processing commands).

UV_SHORT has a substantial number (17) of control variables, but with experience, they have been reduced to 5 significant ones, among which only 3 really matter in most cases but often can be used with their default values:

SHORT_SD_FACTOR The single-dish brightness unit to flux conversion factor. If set to zero, UV_SHORT will attempt to derive it from the information available in the single-dish data

SHORT_UV_TRUNC The longest baseline retained in the pseudo-visibilities. It defaults to the maximum theoretically possible, the single-dish diameter minus the interferometer diameter. Smaller values are allowed, and even recommended if the pointing quality of the single-dish data is moderate.

SHORT_SD_WEIGHT The relative weight scaling factor between the pseudo-visibilities and the interferometer visibilities.

The relative weight of these visibilities is derived by UV_SHORT in order to optimize the shape of the overall synthesized beam. SHORT_SD_WEIGHT is a scale factor to this optimum weight, which may need to differ from 1 in case of poor *uv* coverage in the interferometer data or noisy single-dish data (it should be lower than 1 in this case).

UV_SHORT ? will list these 3 major ones, and UV_SHORT ?? the 2 remaining main ones:

SHORT_TOLE The position tolerance in the single-dish map

SHORT_MIN_WEIGHT The minimum (relative) weight for a spectrum in the single-dish map to be included.

as well as four optional ones needed only if the original single-dish and *uv* data lacks the proper information (antenna diameter and beam sizes)

The UV_SHORT command starts from data in a the format produced by the CLASS command TABLE command, and read in IMAGER through command READ SINGLE. Basically, this is a GDF table containing one line per spectrum, the columns representing the lambda offset, beta offset, weight, and the spectrum intensities. [6]. This data must match spectrally the velocity sampling of the interferometric data. This can be obtained using the /RESAMPLING option of command TABLE in CLASS.

The READ SINGLE and UV_SHORT commands also support a 3-D data cube (as produced by e.g. command XY_MAP in CLASS) as input instead of a CLASS table. Again, the velocity axis must match that of the interferometric data.

UV_SHORT will automatically produce the Zero spacing from the single-dish data when the data does not allow other short spacings to be evaluated. The temporary image produced by UV_SHORT when starting from a CLASS Table is stored in the SHORT buffer, and can be written by command WRITE. This image can aslo be computed separately by command XY_SHORT.

Finally, as UV_SHORT **adds** the short spacing information, UV_SHORT /REMOVE allows to remove it (there is no direct "replace" possibility because the *uv* sampling may change).

8.7 Practical considerations

8.7.1 When are short-spacing information needed?

- If the source size is smaller than 1/3 the primary beam size, short-spacing information is superfluous.
- If the source size is between 1/3 and 1/2 the primary beam size of NOEMA antennas, a single spectra obtained at the IRAM-30m telescope in the direction of the source can be used to produce the zero spacing information with the UV_SHORT command. Indeed, the IRAM-30m diameter being twice the diameter of the NOEMA antenna, all the flux of the source will be measured by a single IRAM-30m spectrum only if the size of the source is smaller than 1/2 the primary beam size of NOEMA antennae.
- If the source size is larger than 1/2 the primary beam size of NOEMA antennas, short-spacing information under the form of an IRAM-30m map is almost always mandatory. The only exception could be wide-field imaging of a region made of unresolved or small (compared to the primary beam size) sources as it may happen when mapping close-by external galaxies for instance. However, adding short-spacing will anyway help the deconvolution.
- Short-spacing information is only useful if the brightness of the extended component is above the noise level. This requires a prior knowledge of the total flux in the imaged

[6] This format is subject to change: Please, refer to the TABLE documentation for up-to-date information

area to be determined. However, this information may be available from previous low-sensitivity single-dish observations. Checking this can avoid wasting a lot of telescope (and astronomer) time.

A generalization to ALMA (12 m antenna) and ACA (7 m antennas) is straightforward.

8.7.2 How to optimize single-dish observations?

One of the main difficulty of the short-spacing problematic is the need of observations from a single-dish telescope at least as big as the interferometer antennas[7]. In this respect, the IRAM-30m and NOEMA are very complementary. Nevertheless, when observing with the single-dish telescope, a few precautions are needed to avoid contaminating the interferometric data with possible artifacts of single-dish data.

- The field-of-view of the single-dish map must be twice the field-of-view covered by the mosaic. The only exception to this rule happens when the source intensity decreases to zero in a smaller field-of-view. Indeed, there is no point in observing an empty sky.
- The observing strategy must enforce Nyquist sampling (or better) of the source at the resolution of the single-dish telescope.
- A particular care should be taken of the pointing, tracking and amplitude calibration and baseline removal as those are critical issues in obtaining a high quality single-dish map to produce short-spacing information. For instance, data with too large tracking errors should be discarded.
- Among "baseline" issues, the presence of continuum sources is to be treated with care. Continuum is difficult to measure with single-dish telescopes, and a (linear or polynomial) spectral baseline is often fitted to avoid atmospheric contamination. In such cases, the combination should be made with interferometer data where the continuum has been removed, and added back later...
- We advise to make many On-The-Fly coverages of the observed field-of-view to get homogeneous observing conditions. Scanning in perpendicular directions is needed to decrease stripping.

Sometimes, single-dish telescope time is scarce and some of the above criteria can not be fulfilled. In those cases, you can still try to use your single-dish observations and our algorithm will try to make its best to get a sensible result. However, any artifact in the combination may directly come from wrong single-dish observations. In other words, do *not* blame the software unless you are sure of the quality of the quality of your single-dish (and interferometric) observations...

[7] If there were no pointing errors, a single-dish of the same size as the interferometer antennas would be strictly sufficient.

9 Self Calibration

9.1 Self-Calibration in a nutshell

```
1  read uv YourData
2  selfcal phase
3  selfcal summary
4  selfcal show
5  selfcal apply
6  uv_map
7  clean
6  write * YourSelf
```

1. Read your *uv* data
2. Self-calibrate the phase
3. Get a summary of the result
4. Show the phase correction between the last 2 iterations
5. Apply the self-calibration
6. Image as usual
7. Clean as usual
8. Save the result if it is worthwhile...

You are done. But, now, if the result is crazy, do not blame the software. Rather read carefully the information below: SELFCAL is simple to use, but there are some pitfalls...

9.2 Self-Calibration Principle

The self-calibration idea is based on the fact that the dominant error terms are antenna-based, while source information is baseline-based. With N antennas, one gets at any time $N(N-1)/2$ visibility measurements, but N amplitude gains, and only $N-1$ error terms for the morphology of the source (phase gains). The $N-1$ number is because only relative phases count. The absolute flux scale is a separate problem, and therefore also $N-1$ relative amplitude gains count.

The measured visibilities on baselines from antenna i to antenna j at time t are, from the simplified measurement equation:

$$V_{\rm obs}(i,j,t) = G(i,t)G^{\star}(j,t)V_{\rm true}(i,j) + Noise \tag{17}$$

where $G(i,t)$ is the complex (phase and amplitude) gain for the antenna i at time t. The true visibility $V_{\rm true}(i,j)$ only depends on the baseline (i,j), not on the time.

Given a source model $V_{mod}(i,j)$, one can derive the antenna gain products at time t, based on the system:

$$\frac{V_{\rm obs}(i,j,t)}{V_{\rm mod}(i,j)} = G(i,t)G^{\star}(j,t) \tag{18}$$

which is an over-constrained process, since there are $N(N-1)/2$ constraints for $N-1$ unknowns. Solving for this over-constrained problem is similar to deriving the amplitude and phase solution from a calibrator observation. In the calibrator case (i.e., an unresolved source like a distant

bright quasar), $V_{\rm mod(i,j)} = (1.0, 0.0)$ (constant amplitude, zero phase), so there is no risk of noise amplification in the process.

For any (not a point-like) source, $V_{\rm mod}$ must be guessed. Self-calibration will use your source to improve the calibration of the antenna-based (complex) gains as a function of time. The practice is to proceed iteratively, based on a preliminary deconvolution solution. Let $V_{\rm obs}(k)$ be the "observed" visibilities at iteration k, with $V_{\rm obs(k=0)} = V_{\rm obs}$ the raw calibrated visibilities. Some of the Clean components derived from $V_{obs}(k)$ are used to define "model' visibilities $V_{\rm mod}(k)$. Then, solving for the antenna gains, one obtains:

$$V_{\rm obs}(k+1) = \frac{V_{obs}(k)}{(G_i G_j^\star)} \tag{19}$$

The model is thus progressively refined, and in the end, satisfies better the initial constraints on the source shape and on the antenna gains as a function of time provided by the measurements. Note that the absolute phase (and hence the position) can be lost in the self-calibration process and it should not be used for absolute astrometry.

There are two types of self-calibration: phase and amplitude self-calibration. The amplitude gain is a more complex problem than the phase gain. Amplitude gains can (and often do) vary with time, but from the measurement equation, a scale factor in the amplitude gain can be exchanged by a scale factor on the source flux. It is thus customary to re-normalize the gains so that the source flux is conserved in the process. An alternate (perhaps not strictly equivalent) solution is to ensure that the time averaged product of the amplitude gains is 1. The two approaches differ by the averaging process.

For any typical source, $V_{\rm mod}$ is non zero and of magnitude smaller than 1 (using the total flux as a scale factor) since the source is partially resolved. So in computing $V_{obs}/V_{\rm mod}$, there is noise amplification. It may even be the case that V_{mod} is zero (case of an extended, over-resolved emission), and thus some (long) baselines will yield no direct constraint on the antenna gains $G(i)G^\star(j)$. But this should not matter too much for self-calibration, for two reasons. First, other (i.e., shorter) baselines may provide contraints on the gains. Second, if all $V_{\rm obs}$ for an antenna are close to zero, it implies $V_{\rm mod}$ must be close to zero too, so an error on the phase of those visibilities (as well as on its magnitude) is not so important.

Self-calibration is related to the "closure" relations. For any triplet of antennas, the phase of the triple product $V_{ij}V_{jk}V_{ki}$ is independent of the antenna errors, and thus is (within the noise) a bias free constraint on the source. Similarly, for any quadruplet of antennas, the amplitude of the ratio $(V_{ij}V_{kl})/(V_{ik}V_{jl})$ is independent of the antenna errors. But here, the noise amplification can be large because of the likelihood to have two small visibilities. For this reason, amplitude self-calibration requires in practice higher signal to noise ratios than phase calibration in the initial deconvolved data set used as a model.

Among the advantages of self-calibration, one may emphasize that antenna gains are derived at the correct time of the science object observation, while they must be interpolated in the classical calibration approach. Both atmospheric and electronic noises are supposed to vary with time, although with different timescales. Gains are also computed in the correct direction on the celestial sphere, while the calibrator-based approach introduces differences in the pointing direction with respect to the science object. The robustness of the approach increases with the number of baselines.

In order to implement self-calibration, it is however necessary that the signal to noise ratio be large enough (the process will require a sufficient bright source). Self-calibration can especially bring significant improvements to the calibration solution in the case of higher than expected

background noise, or in the presence of convolutional artifacts around objects, especially point sources.

9.3 Self-Calibration Implementation

The typical procedure for self-calibration consists in an iterative process based on the following steps:

1. UV_MAP /SELF + CLEAN + UV_RESTORE /SELF:
   from the classically calibrated (and preliminarily flagged) data, define an initial source model through a conservative (not too deep) first deconvolution : the model consists of a reasonably modest number of CLEAN components.
2. SOLVE: determine an estimate of the antenna gains (best fit to the observed visibilities)
3. APPLY: apply the derived gains to correct the observed data
4. STATISTIC: compare to the initial Image, and estimate the improvement through an adequate quality assessment (e.g., improvement of the dynamic range)
5. If necessary, re-build a new model from these corrected data, and iterate until the solution is satisfactory.

Phase-only self-calibration is less stringent on the signal-to-noise ratio (SNR) threshold than amplitude self-calibration, and it should therefore be attempted first. For phase self-calibration to work, the SNR values in the initial data should be at least of SNR>3 per antenna (in a solution interval shorter than the time for significant phase variations for all baselines to a single antenna). The SNR threshold in the initial image depends on the number of antennas and on the adopted time averaging. Depending on the complexity of the source (and the contribution from extended emission), all available baselines may not be considered in the process, and specific preliminary flagging could be necessary. Amplitude errors tend to be negligible for dynamic ranges below about 500. Amplitude self-calibration will thus be eventually attempted in a subsequent step.

Self Calibration is available in IMAGER through the command SELFCAL, which uses an iterative scheme driven by a script (gag_pro:p_selfcal.ima). From the above principle, the script controls
a) the number of iterations
b) the number of selected clean components at each iteration
c) the time scale of the solution, i.e. the integration time over which the gains are assumed to be constant

The parameters of the command SELFCAL are available as SELF_Names SIC variables. The script uses the commands UV_MAP, CLEAN, and SOLVE and APPLY from the CALIBRATE\ language. By default, a solution is searched for the phase calibration only (SELF_MODE=PHASE), and the number of iterations is 3 (SELF_LOOP). For each iteration:

- All components found by CLEAN are kept by default (SELF_NITER = 0), but for simple source structures, 10 components only may be enough (the maximum number NITER of CLEAN components to subtract is automatically guessed by the program in the default process, see CLEAN_NITER and other usual clean convergence control variables),
- the minimum flux density per pixel to be considered by CLEAN can be defined (SELF_MINFLUX = 0)

- The "integration" times (gain averaging) are fixed to a default value SELF_TIMES=45 s (minimum value for NOEMA, while it is only 6 s for ALMA). This can be adapted for each loop (in general, one should start with larger solution times, depending on the SNR values and try to decrease it in order to better sample the atmospheric fluctuations).

The number of iterations can be changed by resizing the SELF_TIMES, SELF_NITER, or SELF_MINFLUX arrays (the number of loops SELF_LOOP is then automatically recomputed). You should make sure that these 3 arrays have the same dimension. If any of the SELF_NITER and/or SELF_MINFLUX array are constant then their dimension is accordingly changed by SELFCAL each time one of these arrays is resized. For instance, the following command allows to define 5 loops with an integration time for solution of 45 sec:

```
let Self_times 45 45 45 45 45 /resize
```

SELF_NITER and SELF_MINFLUX are automatically enlarged to the same size if, and only if, they were already constant. By default, all channels are averaged to compute a "continuum" image, but the range of adequate channels can be specified through SELF_CHANNEL.

SELFCAL PHASE will compute a phase calibration, but will not apply it. One needs to call SELFCAL once more with the argument APPLY in order to apply the solution. The script really applies the solution if and only if the previously found solution can be considered as a good one (see SELF_STATUS argument value).

SELFCAL APPLY automatically saves the parameters and results in the *selfcal.last* file. By default, data are flagged if no sufficiently good solution is found. SELFCAL APPLY keep tracks of whether the solution has already been applied through SELF_APPLIED. SELFCAL APPLY will refuse to apply "bad" solutions: solutions are declared "bad" if the improvement in dynamic range and noise level is insufficient (i.e. below a precision level controlled by SELF_PRECISION). In this case, the SELF_STATUS variable is negative. In this case, the user can still decide to apply the solution directly using command APPLY, but the SELF_APPLIED variable will not be updated.

The merit criteria for the quality assessment of the computed solution are the final dynamic range, and the Clean map noise at each iteration. These quantities are stored in the SELF_DYNAMIC and SELF_RMSCLEAN variables (at each iteration). The dynamic is defined as the ratio of the peak flux density value to the noise in the clean map (automatically estimated with the command STATISTIC). The minimum signal to noise ratio value (for an antenna) for a valid solution is SELF_SNR=6 by default.

SELFCAL can be controlled through a widget, using command SELFCAL /WIDGET (see Figure 1)

It is possible to visualize the computed corrections with the command SELFCAL SHOW. The solution computed with the SOLVE command is written in a *'self_sname'.tab* file. By default, the difference between the last two iterations is displayed. For PHASE, the phase difference should be close to zero if the solution converged, and for AMPLI the amplitude values close to 1. It is also possible to show the difference between two specified iterations. In addition, the command SELFCAL SUMMARY will display the results of the process in terms of resulting noise and improved dynamic range. If the solution is satisfactory, the command SELFCAL SAVE can be used to save both the results and the parameters in the *selfcal.last* file. (SELFCAL APPLY performs an implicit SELFCAL SAVE.)

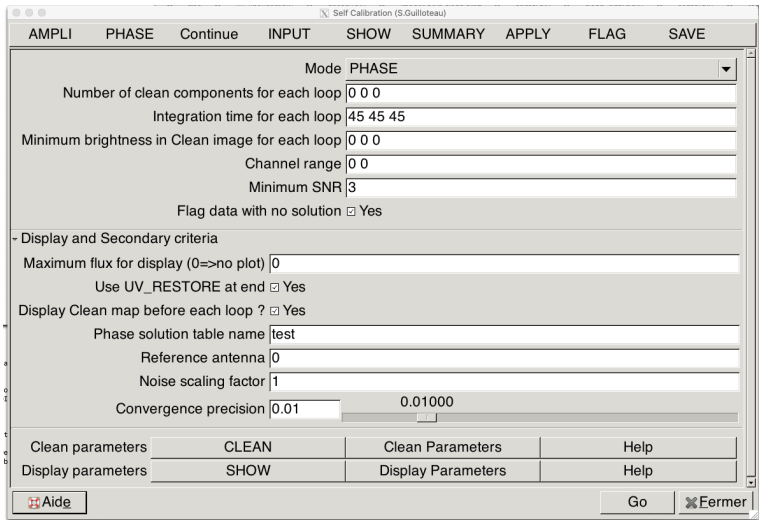

Figure 1: The Self-Calibration widget

9.4 Basic use

9.4.1 Timescale for averaging solution interval

The solution interval, or timescale used to average the solution for the gain variations, is the result of a tradeoff between the timescale of the true gain variations (e.g., changes in the atmospheric conditions or electronics) and the data averaging which is necessary to reach a minimum SNR value for the visibilities. In principle, this timescale should be smaller than the coherence time of the atmospheric fluctuations for the phase solution, typically one minute. It is in general longer for the amplitude gains, since here these are changes in the atmospheric transparency or antenna gains which matter. It is therefore recommended to start iterations with a not too short averaging time, and to decrease it in a second step if the self-calibration was successful. It is also recommended to use the same integration time for the last two iterations, in order to ease the interpretation of the results and of the `SELFCAL SHOW` display.

The current self-calibration method computes baseline-based gains (observed complex visibility divided by model prediction) for each visibility, and performs the time averaging on these baseline-based gains. Antenna based gains are derived from the time-averaged baseline-based gains. This is in principle an optimal method, since the model is not noisy: the linearity of the first step guarantees a noise decrease as square root of time.

9.4.2 Quality assessment and data flagging

Validation of the solution One of the difficulties of self-calibration is to evaluate whether it has improved the image or not. The self-calibration solution is biased towards the assumed model. If used with insufficient signal to noise, it will tend to produce a point source at the initial peak position, and the bias will be of order of the noise. This may be inappropriate. Currently, the validity of the self-calibration solution is based on the estimated signal to noise ratio for the gains at each time step. If that SNR is below a user-controlled threshold (by default, `SELF_SNR=6`), the corresponding data is flagged (default value: `SELF_FLAG=YES`) or kept WITHOUT self calibration (if `SELF_FLAG=NO`).

Flagging or not flagging ? The decision to flag or not results from a trade-off:

- SELF_FLAG = Yes : will result in no contamination by bad data, but may lead to lower angular resolution since long baselines may be flagged.
- SELF_FLAG = No: will avoid loosing all long baselines (where the SNR is lower)

Both options may be explored, and it is recommended to check afterwards the final angular resolution with and without flagging.

The following scheme is proposed to check the validity of the self-calibration solution:

- read the status using SELFCAL SUMMARY
- use SELFCAL SHOW to verify if the solution is converged
- if it looks good, but noise is still far from theoretical, try again to self-calibrate with a shorter integration time (SELF_TIMES).
- if it is not good, try to increase SELF_TIMES and find an optimum value. For ALMA data, typical values may be in the range $6 - 60$ s, and for NOEMA in the range $45 - 120$ s. Alternatively, you can also try to decrease SELF_SNR to lower values, but never less than 3.

From our experience, the number of loops SELF_NLOOP does not impact much the quality of the solution, and 2 to 3 iterations are usually sufficient.

Warning: the comparison with theoretical noise relies on a proper scaling of the weights of the UV data. This is fine for the IRAM array, but data exported from CASA is not always correct in this respect. UV_PREVIEW can warn you about potential issues in this respect. UV_REWEIGHT can also evaluate the scaling factor that should be applied to the weights to recover the apparent noise level. UV_PREVIEW requires a sufficient number of spectral channels for this purpose. By default, UV_REWEIGHT suffers from a similar restriction, and both commands may fail if the bandwidth is clobbered with spectral lines or has strong continuum. However, UV_REWEIGHT as a TIME mode where the noise is estimated from consecutive visibilities, and thus is not affected by this limitation.

The appropriate scaling factor can be specified in SELFCAL by variable SELF_SNOISE.

9.5 Advanced use

Amplitude self calibration

The amplitude calibration (SELFCAL AMPLI) is a secondary step in the self-calibration process. In general, it should only be attempted if the phase calibration was already excellent, i.e., once you obtained the best solution by adjusting the SELF_TIMES parameter for the SELFCAL PHASE command. If possible or needed, the amplitude self-calibration should use a *longer timescale* than the phase calibration (typically, SELF_TIMES = 120 s). SELFCAL automatically adjusts the gains so that their mean is 1, in order to avoid changing the flux scale. In practice, it is useless if the expected noise limited dynamic range is less than about 300.

Cases where Amplitude self calibration may be essential

If PHASE self-calibration does not sufficiently improve the image despite ample signal-to-noise, Amplitude self calibration may do it. This situation often occurs when the observations span different dates, so that the relative flux calibration between the separate dates is inconsistent. In this case, a AMPLI self-calibration may be of great help. For compact sources, the flux consistency scale across dates can be also checked and cured using command SCALE_FLUX.

Note however that the resulting improvement does not necessarily produce a higher fidelity image. It removes the inconsistencies, but the selected (average) flux scale has no guarantee to be good. Flux calibration should be independently checked if this situation occurs.

Support restriction, flux threshold

Support restriction in the CLEAN process may be needed to build a simpler model for very complex, extended sources only. Command MASK THRESHOLD can be useful in this respect. Similarly, limiting the flux per pixel in the model (see SELF_MINFLUX) may help, since noise peaks are then ignored. However, the later may fail if the source is too extended: low level brightness can be important for self-calibrating short baselines.

9.6 Transfering the solution to other *uv* data sets.

An important use of the Self-calibration is to compute a calibration solution using wide bandwidth data (*continuum* data, where signal-to-noise can often be maximized) and apply it to high spectra resolution (*line*) data.

The technique is quite simple. Solve for Self-calibration using the wide bandwidth data as usual, and save the results (gain tables) using the WRITE CGAINS command (eventually one for each of the PHASE or AMPLI steps of Self-calibration).

At any time, you can read back these gain tables using READ CGAINS, read an *uv* data set with READ UV and apply the gain solution using command APPLY. Since Self-calibration normally corrects for atmospheric errors, the derived 'phase" correction is in general here to compensate for a pathlength change, i.e. the corresponding phase correction should scale as Frequency. APPLY DELAY instructs IMAGER to take this effect into account.

9.7 Data re-weighting

The data weights are not changed during the self-calibration solution search.

However, when applying the gain changes, the SELFCAL APPLY command implementation makes a conscious choice the consequence of amplitude changes on the data weights. The weights are scaled down when the amplitude is increased (as thermal noise is also increased in proportion), but are untouched if the amplitude is decreased. In essence, IMAGER considers that self-calibration always degrades the noise.

Note that as a result of the different behaviour for the weights during the search and in the use of the solution, the last iterated image in the self-calibration search is not the image obtained after applying the solution. Even the angular resolution may change (hopefully only slightly).

This of course only applies for AMPLI self calibration, since phase changes do not affect the noise.

9.8 Data flagging

Self-calibration is intended to improve upon a basically good solution.

When corrections are too large, it may be wise for the user to flag data that suffer from excessive corrections. This is possible *a posteriori* (i.e. after a SELFCAL APPLY command) using command SELFCAL FLAG as below

```
selfcal flag [Threshold]
```

Threshold is the correction above which data should be flagged (in degrees for Phase or Delay, no units for Ampli). Like for SELFCAL APPLY, the current value of SELF_MODE is used to determine the mode.

Apart from minor verifications and messages upon the self calibration solution validity, the command is equivalent to command APPLY /FLAG

```
apply 'self_mode' 0 /flag [Threshold]
```

10 UV plane analysis

IMAGER is not only a tool to produce images. It also allows direct analysis by model fitting to the UV data.

A number of commands are directly related to these possibilities:

- READ UV
- READ MODEL
- SHOW UV_FIT
- SHOW UV
- UV_CIRCLE
- UV_CONTINUUM
- UV_DEPROJECT
- UV_FIT
- UV_RESIDUAL
- UV_RADIAL
- UV_SELECT

The overall intent of these commands is described below. Please refer to the corresponding HELP for more details about the command syntax, options and control values.

- UV_FIT is the primary fitting command. It allows to adjust simple, analytic models of source brightness to the current UV data. Up to 8 different source models can be combined in a single fit [8].

  The UV_FIT results can be displayed using command SHOW UV_FIT. The visibilities modeled by UV_FIT (or the fit residuals) can be displayed using command SHOW UV, either alone, or superimposed to the current UV data.

  UV_FIT works on spectral line data. To optimally fit continuum data, it is best to convert the spectrally resolved information into a *bandwidth synthesis continuum uv* table.

- UV_CONTINUUM converts a spectral line *uv* table into a *bandwidth synthesis continuum uv* table. UV_CONTINUUM requires some knowledge of the field of view to evaluate how many channels should be averaged together. This is done using the same parameters (MAP_FIELD, or the product of MAP_SIZE by MAP_CELL) and subroutines as for commands UV_STAT SETUP and UV_MAP.

  The *bandwidth synthesis continuum uv* table is peculiar in the sense that it has several baselines for the same pair of antennas at the same observing time: one baseline per independent frequency point, in order to preserve optimally the angular resolution that goes as λ/B. In practice, it is only useful for UV_FIT, but not suitable for self-calibration for example.

- SHOW UV will display the best fit visibilities (stored in the UV_MODEL data set) if (and only if) the UVSHOW%FIT variable is set to MODEL, POINT or CURVE. When UVSHOW%FIT variable is set to MODEL, the UV_MODEL data set is assumed to exist, and not re-computed. It is displayed as data points (not as a curve).

[8] this is probably too much, and would most likely lead to instability in the model fitting !..

- SHOW UV_FIT is intended to display the fit results (the best fit parameter values and their errors) as a function of channels (or velocity, or frequency). This is often a nice way to show the detection of a signal when the signal to noise is limited, but of course is useless for single channel data !
- READ MODEL can import the visibilities from an outside model, so that these can be overlaid to the current UV data by SHOW UV with UVSHOW%FIT variable set to MODEL.
- Command MODEL computes the visibilities (the UV_MODEL dataset) for the best fit parameters found by UV_FIT **or** for the selected Clean components, depending whether UV_FIT or CLEAN was used last.
- Similarly to command MODEL, UV_RESIDUAL computes the residual visibilities (the UV_RESIDUAL data set).
- Commands UV_CIRCLE UV_DEPROJECT UV_RADIAL can be useful in conjunction to UV_FIT, as they can provide a (deprojected, optionally azimutally averaged) visibility data set which can the be much more easily compared to simple, circularly symmetric models.

  Beware that the convention for *angle* are different in these commands (that use the PA of the axis) and UV_FIT that use the PA of the major axis: they thus differ by 90 degrees.
- Finally, command UV_SELECT allow to select which data set is displayed by SHOW UV among the UV data, the UV_MODEL or the UV_RESIDUAL.

Try @ gag_demo:demo-uvfit for some examples.

11 Continuum emission

IMAGER naturally handles many spectral channels. However, the astronomer may be interested in broad-band continuum emission. Extracting the properties of such a continuum emission may be a difficult issue in some cases. IMAGER offers a number of tools to do so, including the notion of a CONTINUUM image.

To first order (over a limited frequency coverage), continuum emission can be represented by a a flux (at some reference frequency) and a spectral index. Both flux and spectral index may vary spatially. Such a representation is often valid over a factor 2 or 3 in frequency. In general, the spatial distribution of these properties of the continuum emission may be widely different from those of the spectral lines. Furthermore, spectral lines can be easily optically thick at some velocities, and may hide the continuum emitted from behind, and thin at others. The continuum itself may sometimes have sufficient opacity to obscure line emission.

Thus a proper extraction of the continuum properties can formally only be done in the image plane, after imaging the combined continuum and spectral line emission, and cannot rely on a separation in the *uv* plane at the level of the visibilities.

In general, this cannot be done by simply imaging channel per channel. The reason for that is that continuum emission is in general fainter than the spectral line one, and often faint enough to have only limited signal to noise in a single channel. Deconvolution of such emission is noise limited. Yet, when averaging over many channels (PolyFix on NOEMA offers 2048 channels per wide band window, ALMA has up to 4096 per spectral window), the S/N changes by a large amount, and the undeconvolved sidelobes from the continuum emission appear.

11.1 Continuum imaging

To avoid this, it is better to deconvolve the continuum emission by averaging as many channels as possible. Line emission should be filtered as much as possible at this level, since otherwise it contaminates the information which is sought. A further issue is that the effective resolution of the array goes as λ/B. With a wide enough frequency coverage, this is a significant change over the bandwidth that must be accounted for. It actually helps to obtain a better *uv* coverage, a process called *bandwidth synthesis*.

In IMAGER, the whole process can be done in four commands
`uv_preview`
`uv_filter`
`uv_map /cont`
`clean` (and optionally `uv_restore`)

- Command `UV_FILTER`, used after `UV_PREVIEW`, will remove all regions of the spectrum that are contaminated (above the noise level) by spectral lines.
- Command `UV_MAP /CONT` will then image in bandwidth synthesis the remaining continuum emission. After that, `CLEAN` and `UV_RESTORE` can deconvolve and properly restore the image.

In command `UV_MAP /CONT`, the user can specify the spectral index of the emission to optimize the signal to noise by giving the best weight to each spectral channel.

This part is only a first step in a proper analysis. The next one is to image the remaining signal in spectral lines, and ultimately add back the deconvolved continuum image. To help the user in this, the image obtained as above is automatically saved as the `CONTINUUM` SIC image variable.

11.2 Split Continuum and Spectral line imaging

To image the spectral line, we simply remove in the *uv* plane the continuum defined before and image the resulting spectral line data:
uv_baseline
uv_map
clean (and optionally uv_restore)

The last step is to add back to this image the deconvolved continuum one:

```
define image my_clean * real /like clean /global
map_combine my_clean ADD clean continuum
```

where the last command is equivalent to the following loop:

```
let my_clean% clean% !  Set its header
for i 1 to clean%dim[3]
let my_clean[i] clean[i]+continuum
next
```

Because the Fourier Transform is a linear operation, we end up with a combined data set that properly includes all the emission, line and continuum, deconvolved in a (well almost...) [9] optimal way.

Further analysis of the spatial variations of the continuum emission (including variations of its spectral index) can be done on this combined image that contains all the information, see Sec.11.4

Note that we used in this example commands uv_filter and uv_baseline with their default behaviour provided by uv_preview: the user could specify more adapted values using the command options and/or control variables if needed.

11.3 Combined Continuum and Spectral line imaging

An alternative to the method described in Sec.11.2 is to use the Continuum image, or more precisely the Continuum Clean Components, as a starting point for the deconvolution of the continuum+line data. The first step is to save the Continuum Clean Components:

```
uv_preview
uv_filter
uv_map /cont
clean
write cct Continuum.cct
```

Re-imaging the data set with the continuum included can be done by uv_filter /reset

```
uv_map
read cct Continuum.cct
clean /restart
```

uv_restore (as an optional step).

UV_FILTER /RESET reset the weights of the line emission channels to their proper values. Reading again the Clean Component list is needed because UV_MAP destroys this list. The Clean Component list of the continuum data is then used as a starting value for every spectral channel by the CLEAN /RESTART.

[9] The combination is only optimal if the spectral index has no spatial variations. A better handling of spatially variable spectral index is under study.

11.4 The CONTINUUM image

The CONTINUUM image can also be produced from a (deconvolved) spectral line data cube by the MAP_CONTINUUM command. Compared to the method presented in Sec.11.1, this has the advantage of being able to handle spatially variable spectral index, and also spatially variable spectral line contamination. It has the drawback of limited S/N for the deconvolution, as already mentionned, unless the spectral line data cube also contains the continuum emission using one of the two methods presented in Sec.11.2-11.3.

Also, the CONTINUUM image can be read from a GILDAS data file using command READ CONTINUUM. And of course, it can be written to GILDAS data file by command WRITE CONTINUUM.

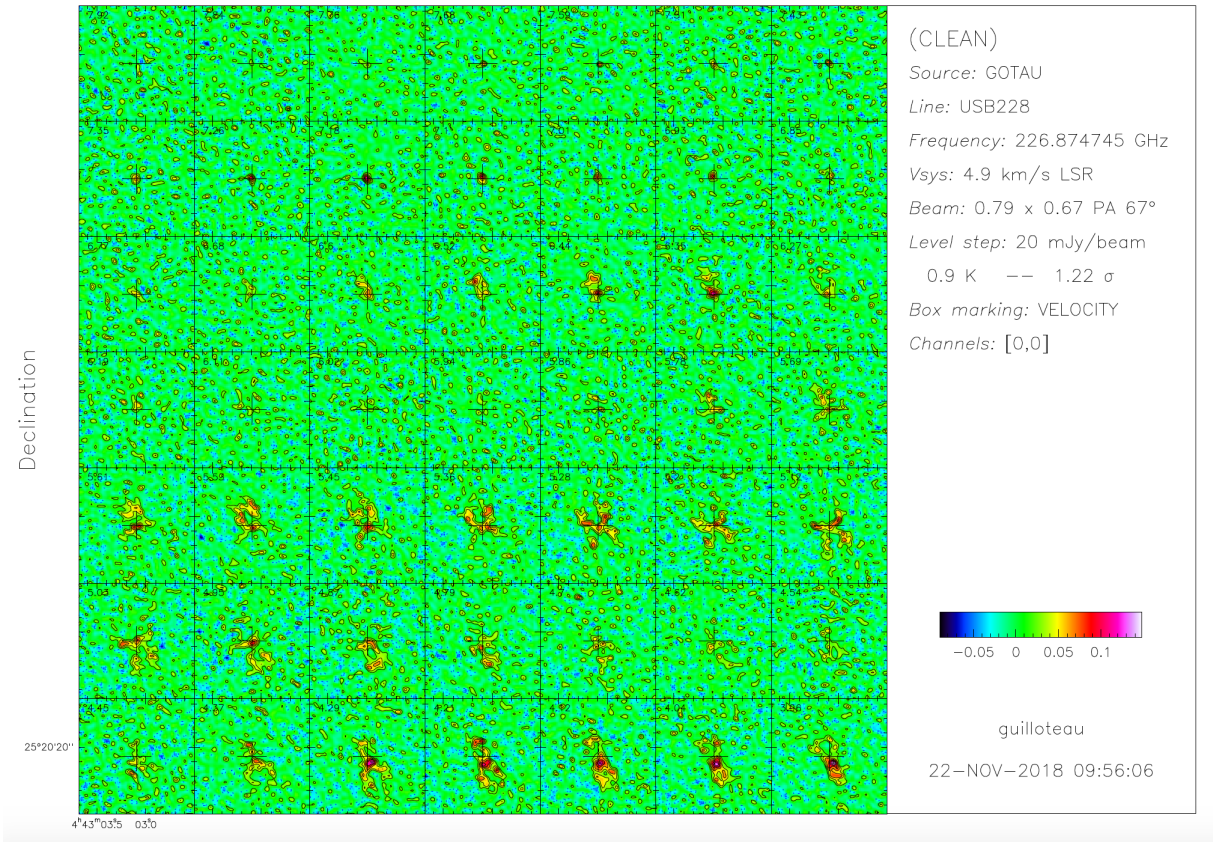

Figure 2: The SHOW CLEAN output.

12 Visual Checks and Image Displays

The ultimate (and often sole) way to evaluate the data quality and suitability for scientific analysis is to visualize it. IMAGER offer simple, yet powerfull, tools to do so.

Most visualization (or related) commands are grouped in language DISPLAY\ (which may be used alone to provided a simple stand-alone data cube visualisation tool.).

12.1 The VIEWER program

The VIEWER program is a subset of IMAGER that only contains the DISPLAY\ language. It can be used to provide visualisation of any GILDAS data cube. Besides visualisation commands that are described in the sub-sections, VIEWER includes the CATALOG (and its related FIND command) that controls the behaviour of some displays, as well as generic support commands like STATISTIC

12.1.1 The SHOW command

In general, command SHOW allows a per-plane display of any SIC 3-D image variable, with contours overlaid on bitmap for each plane: e.g.
SHOW DIRTY 30 -30 will display contour and bitmaps of each channel of the DIRTY image, starting for channel 30 and ending 30 channels before the last one. See Fig.2 for an example. Command SHOW /SIDE (and also command SHOW when the SHOW_SIDE variable is set to YES) will call the cursor, so that the pixel values at the cursor position are displayed in the <VSIDE panel.

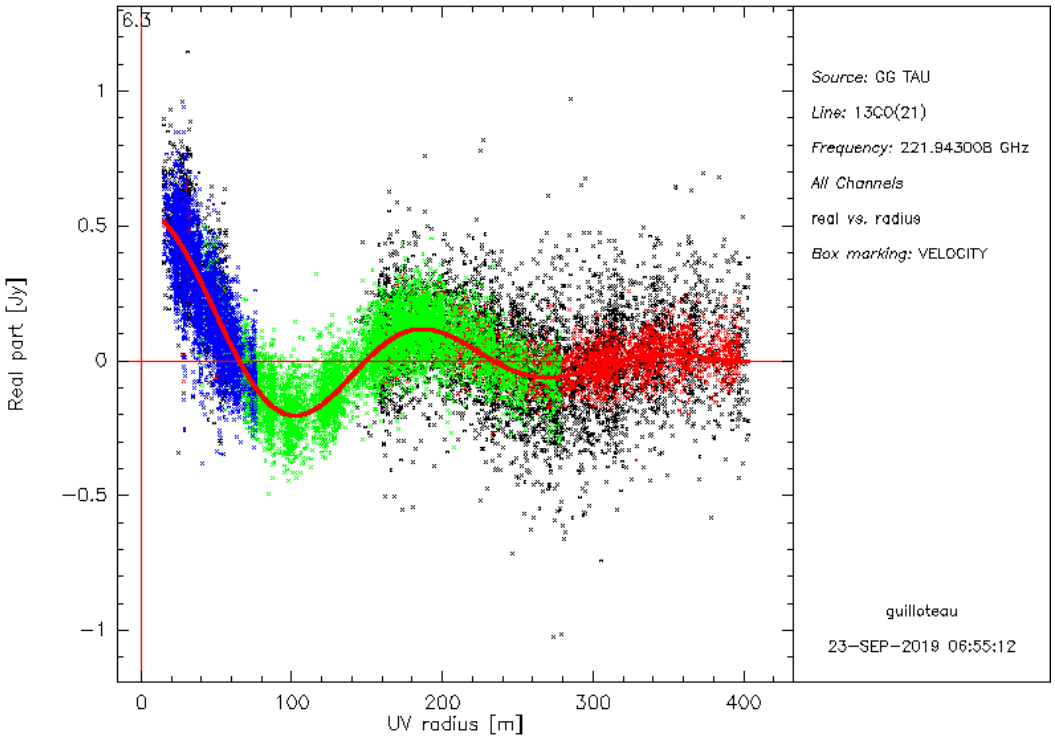

Figure 3: The SHOW UV output with results from UV_FIT command overlaid.

A direct use on Gildas 3-D image or UV data files is also possible: SHOW `Filename.ext` will directly display the file if possible. It also works for simple FITS files in which the data array is in the HDU.

For *uv* data, SHOW UV can plot visibility values such as amplitude as a function of time, baseline length, etc..., again on a per channel basis. Fit results can be overlaid, as shown Fig.3

SHOW can also display more specific issues:

- SHOW CCT will display the cumulative flux as a function of number of clean components (Fig.4).
- SHOW COVERAGE will display the *uv* coverage (it assumes there is only one, and not one per channel, because the display time is long, see Fig.5)
- SHOW SELFCAL behaves as SELFCAL SHOW
- SHOW FIELDS displays the fields of a Mosaic.
- SHOW NOISE displays the histogram of the intensity distribution for each channel, estimating the noise by fitting a Gaussian in these histograms (see Fig.6).
- SHOW UV_FIT displays the UV_FIT results for spectral line data.

SHOW recognizes many other keywords. See HELP SHOW for further details.

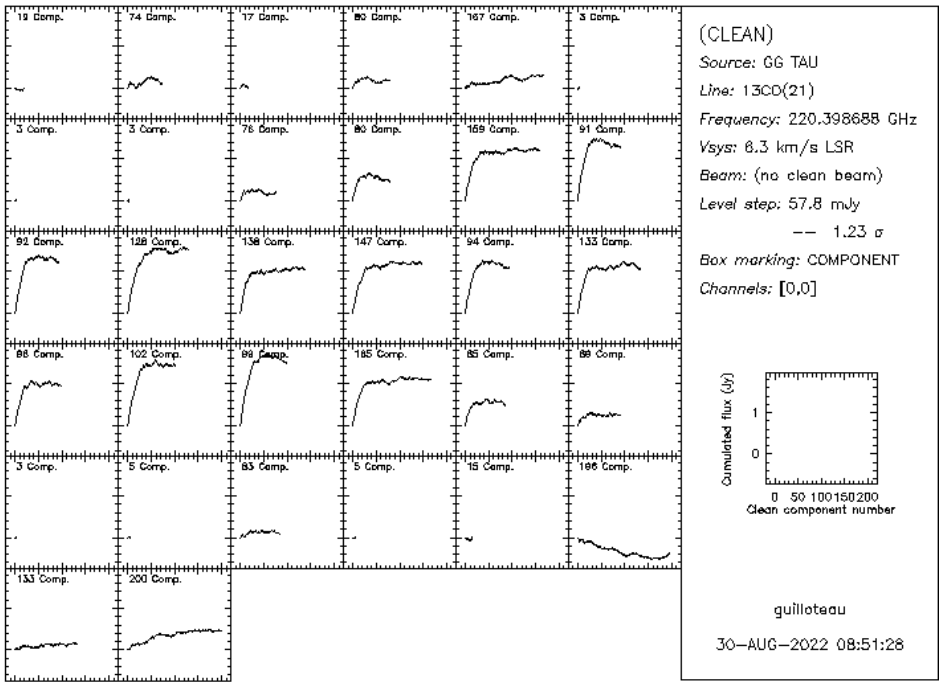

Figure 4: The SHOW CCT output.

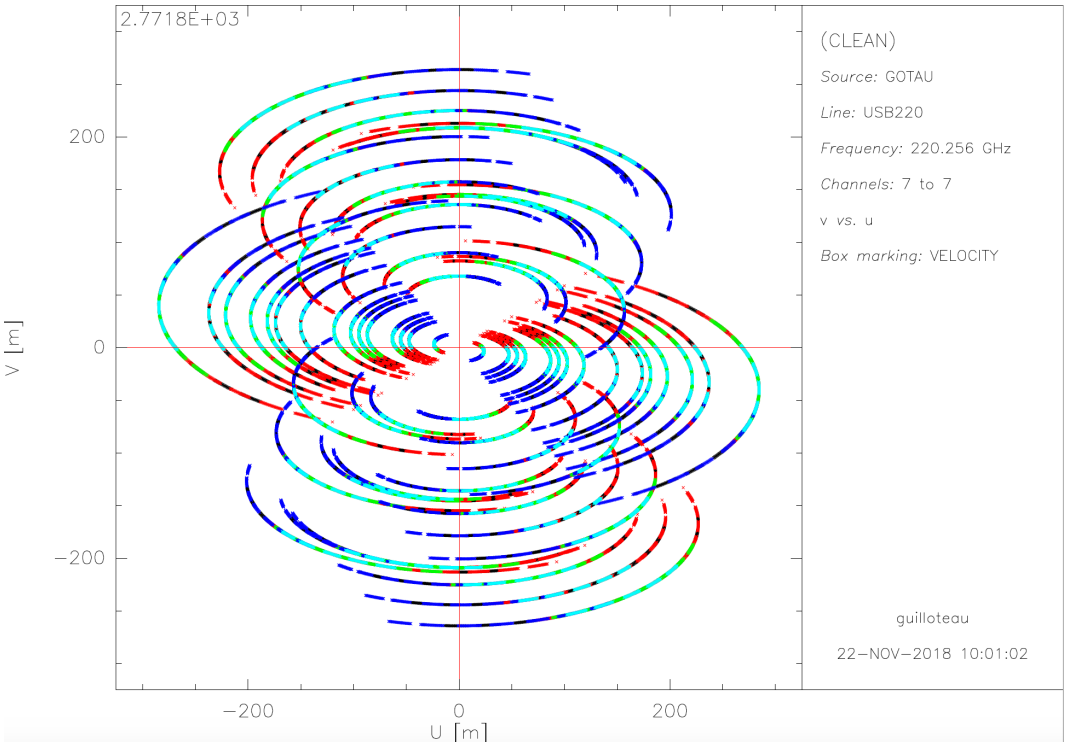

Figure 5: The SHOW COVERAGE output. Colors indicate different dates.

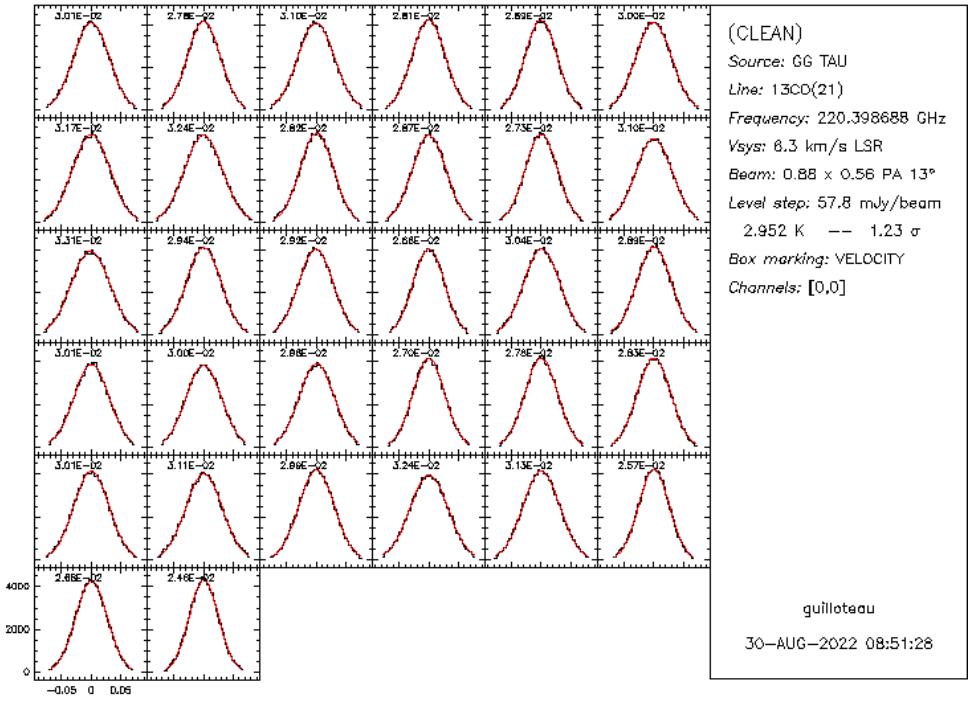

Figure 6: The SHOW NOISE output.

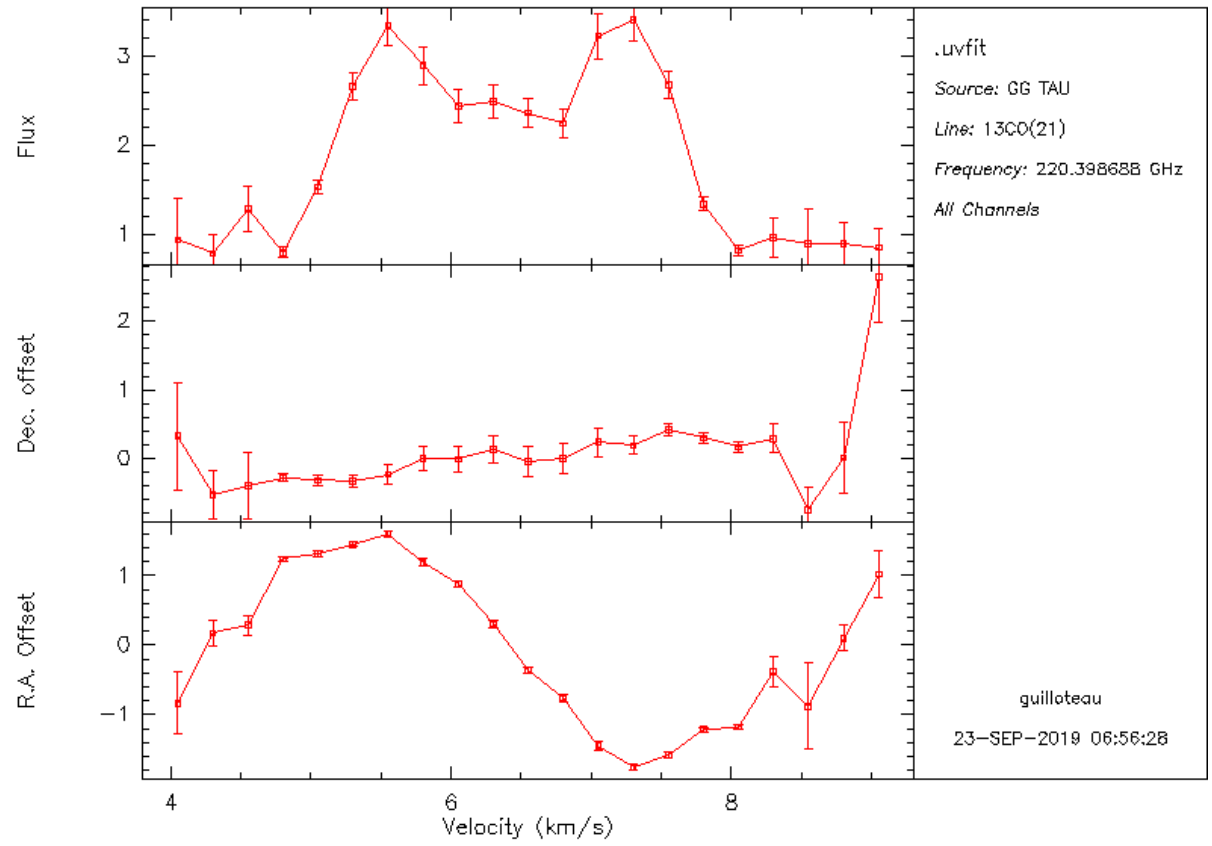

Figure 7: The SHOW UV_FIT output.

12.1.2 the VIEW command

The VIEW command is a powerful alternative to SHOW, the later being inefficient when the number of spectral line channels is large.

VIEW provides a 4 panel display for 3-D data cubes, with the current channel bitmap, the integrated area bitmap, the current spectrum and the integrated intensity spectrum. The spectra can be displayed with 2 simultaneous frequency windows, a broad and a zoomed one, allowing browsing through a large number of channels. Spectral line identification (from the line catalog specified using command CATALOG) can be added by typing L when the cursor is on one of the 2 broad-band spectra.

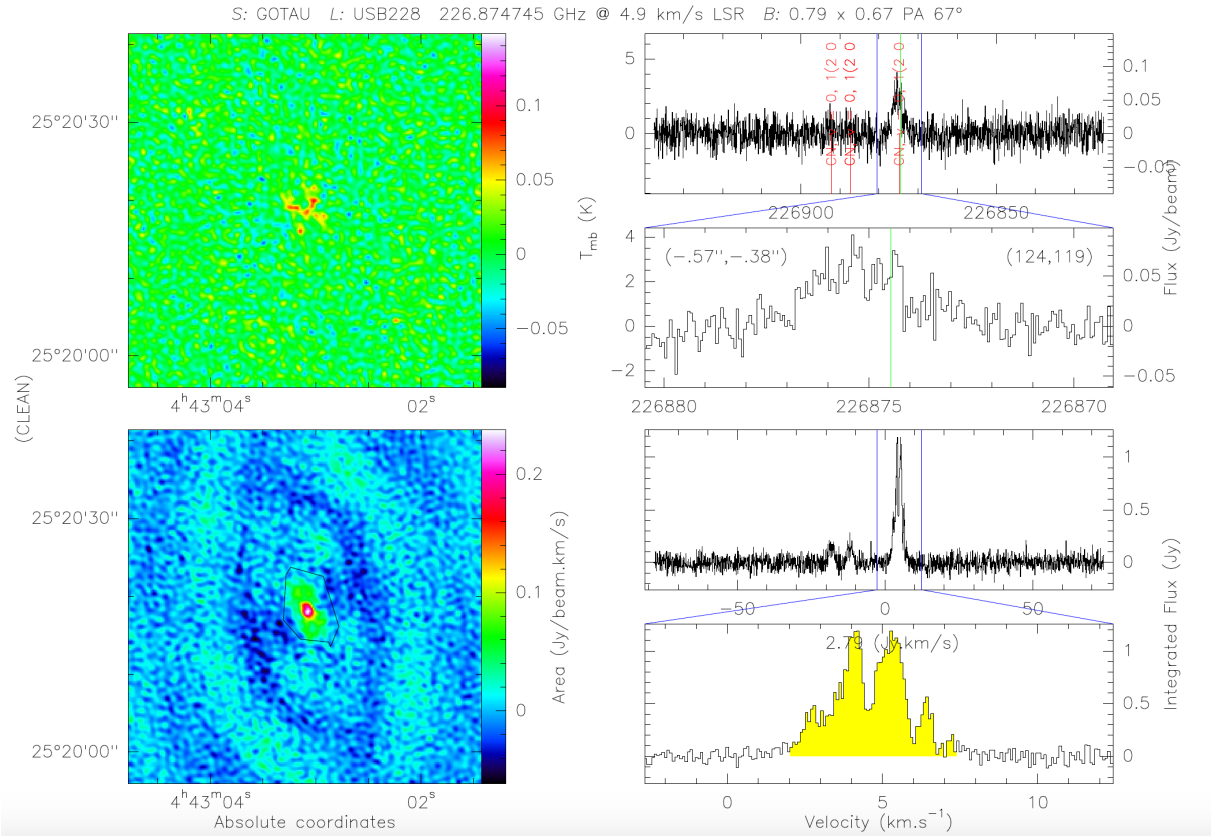

Figure 8: The VIEW output.

VIEW CCT will display the cumulative flux of Clean components for all channels in just one panel, instead of a per-channel panel for SHOW CCT

Similarly, VIEW NOISE will plot all pixel intensity distributions from every plane in a single panel, contrarary to SHOW NOISE.

VIEW /OVERLAY allows to overlay (in contours) the plane of a second data cube. The main use is to overlay the continuum, or a plane of the current data cube to help revealing velocity structures.

VIEW will fallback to SHOW whenever it has no specific support for a given action.

Like SHOW, VIEW /NOPAUSE will not loop interactively to explore the data cube, but just display the view with its current parameters (channel, velocity range, support). The same behaviour can be obtained by setting DO_LOOP to NO.

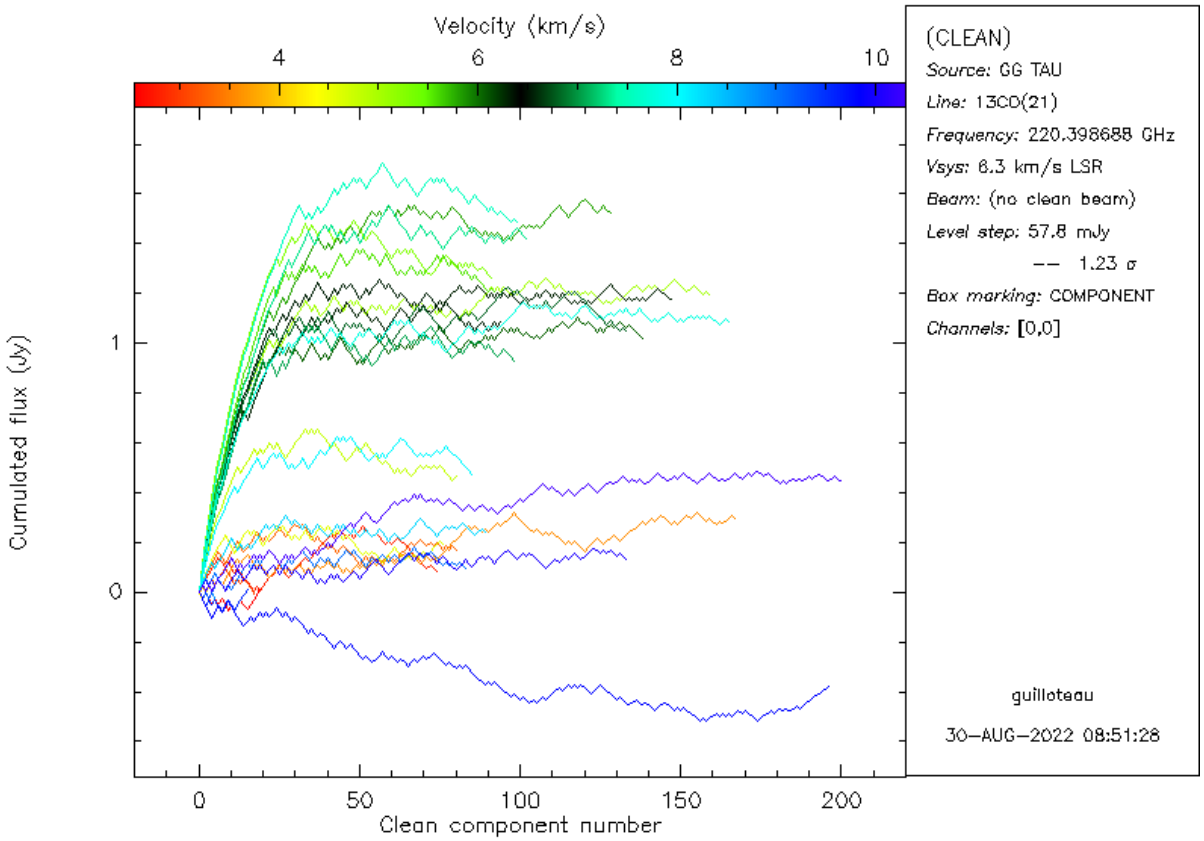

Figure 9: The VIEW CCT output.

12.1.3 the INSPECT_3D command

Another alternative to SHOW or VIEW is the INSPECT_3D command, which presents cuts along the 3 main directions of the data cube (xy, xv, vy). Like SHOW /SIDE and VIEW, INSPECT_3D has a <VSIDE display panel where data values at current position are displayed.

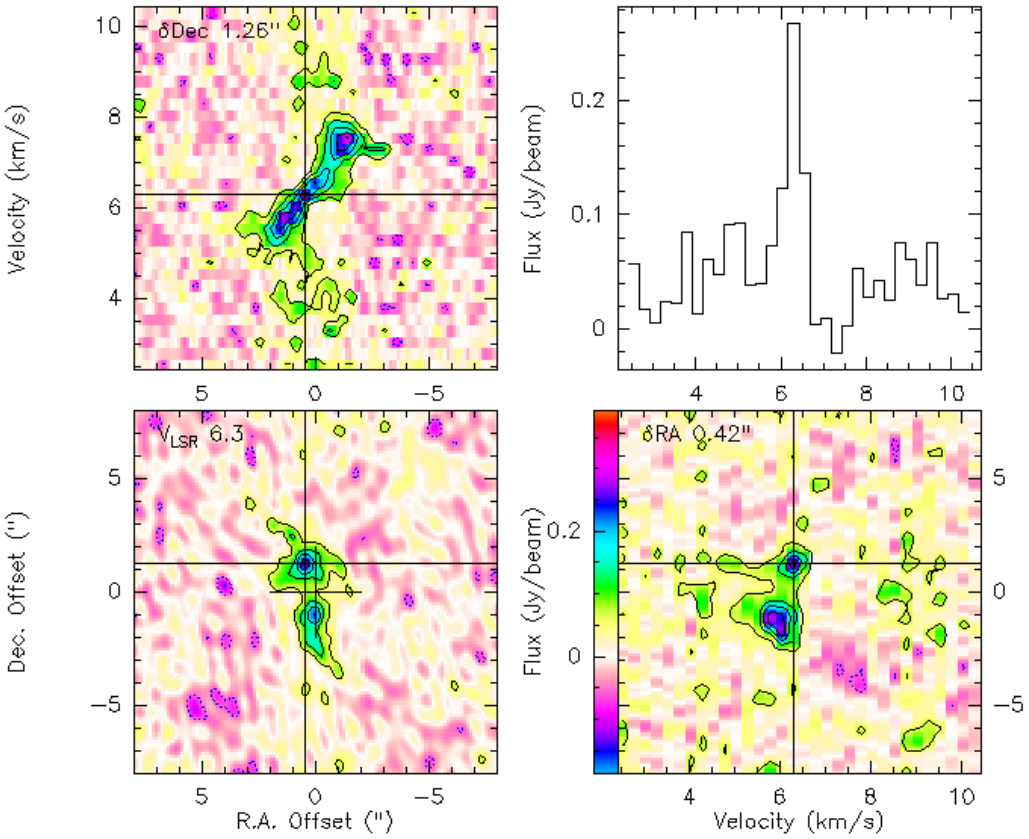

Figure 10: An INSPECT_3D output. Cursor controls the coordinates of the 3 orthogonal cuts and spectrum.

12.1.4 the EXPLORE command

The latest alternative to SHOW, VIEW or INSPECT_3D is the EXPLORE command, which presents spectra extracted from a datacube around an image of one velocity channel or of the velocity integrated area.

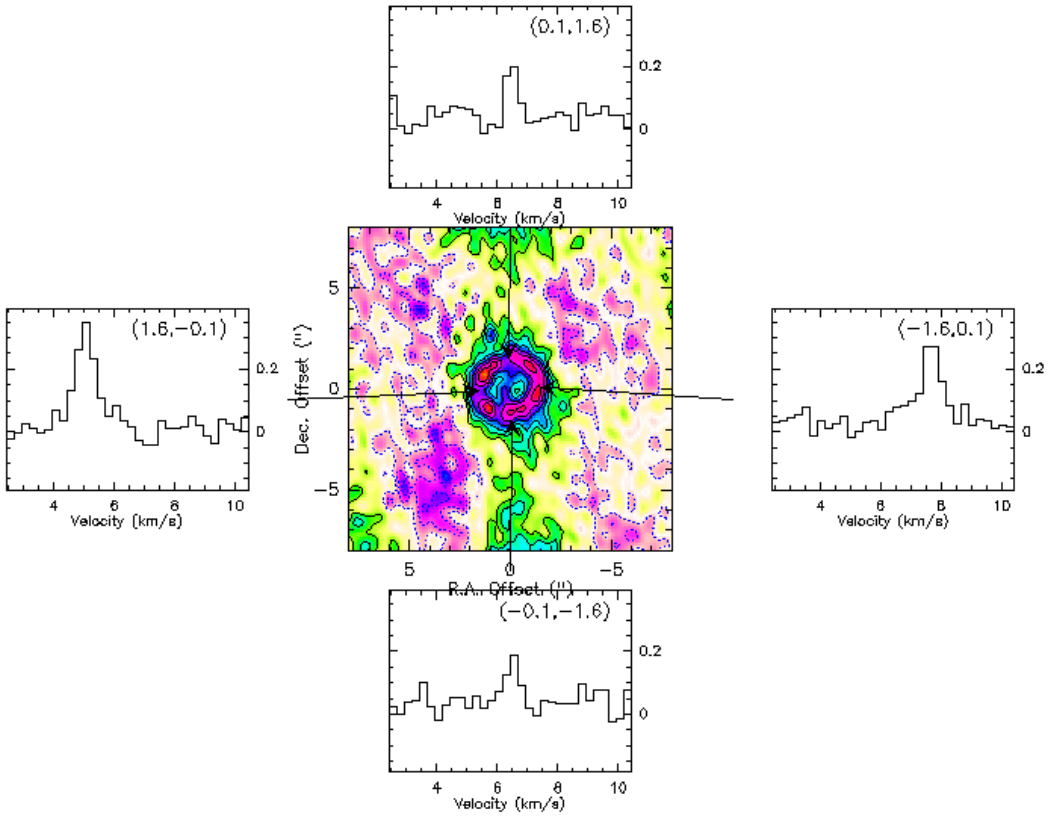

Figure 11: A sample EXPLORE output. Up to 8 spectra can be displayed around the central image.

12.2 Specific IMAGER visualisation tools

In addition to the generic tools offered in the DISPLAY\ language, IMAGER contains commands that are more specific to the UV data handling or imaging process, or to advanced analysis commands. The former category includes UV_PREVIEW, as well as SELFCAL SHOW, while an example of advanced specific display command is KEPLER SHOW

Like SELFCAL, some other commands (e.g. KEPLER SHOW) have a SHOW mode. In general, SHOW will then recognize the reverse syntax, SHOW KEPLER in the above example.

Use HELP SHOW for more details, as the Help is in general more up-to-date than the compiled documentation which is more intended to document general features.

12.2.1 The UV_PREVIEW command

With large datasets, imaging can be long. IMAGER offers a simple, fast pre-imaging viewer through command UV_PREVIEW.

This command will display the spectra obtained towards the phase center at several spatial scales (the default is for 4 tapers). It will also attempt to detect spectral features, by analyzing for each spectrum the noise statistics and the outliers. It performs automatic line identifications, using database(s) in the LINEDB\ or ASTRO data format selected by command CATALOG. Potential spectral lines in the band are displayed in blue, and detected ones in red. UV_PREVIEW also warns the user about improper scaling of the data weights. The line emission region is indicated in grey.

The result of this automatic signal detection and line identification is saved in a SIC structure named PREVIEW%, that is automatically used by commands UV_BASELINE /CHANNELS and UV_FILTER /CHANNELS to respectively remove the continuum and filter the line emission to produce a continuum data set. The detected line frequencies are stored in PREVIEW%FOUND%FREQ and their names in PREVIEW%FOUND%LINES. This can be used to reference the velocity scale of the data to one of the detected lines, by using command SPECIFY FREQUENCY 'PREVIEW%FOUND%FREQ[1]' for example before further processing.

An example is shown in Fig12

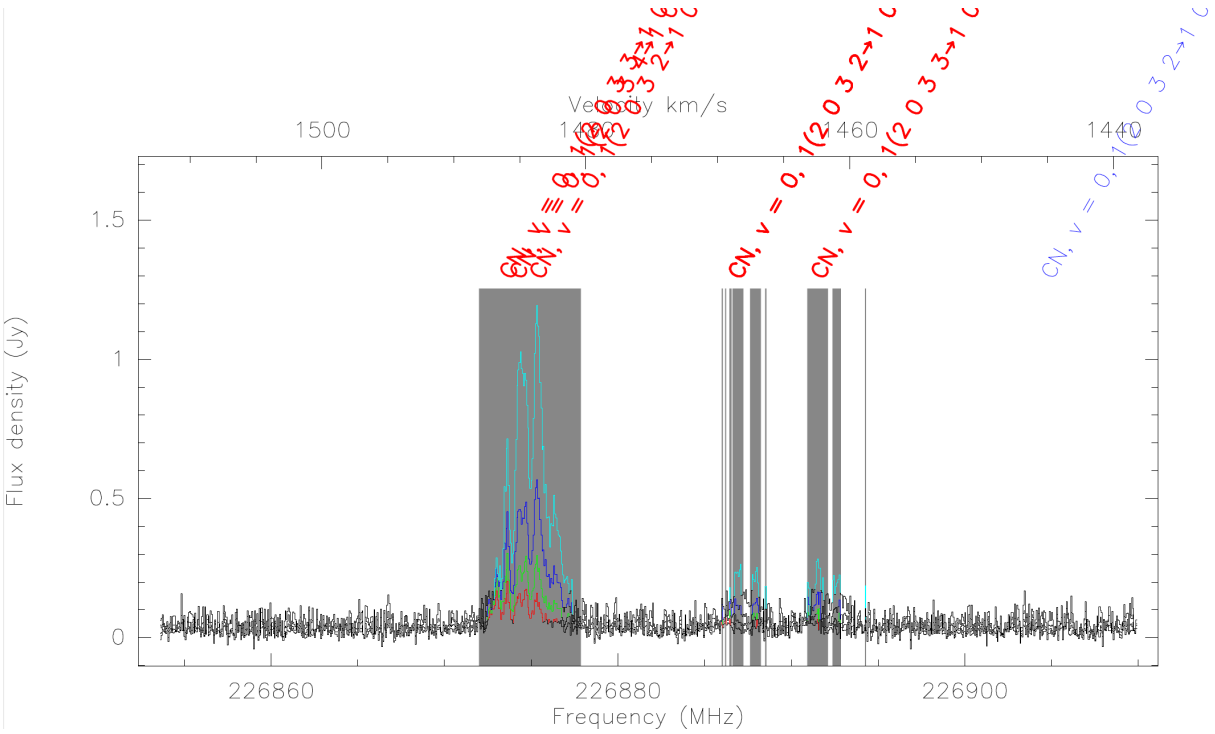

Figure 12: The UV_PREVIEW output

12.2.2 the SELFCAL SHOW command

Verifying the convergence of a self-calibration is important. Figure 13 shows an example, while Fig.14 shows the total phase correction between the original data and the last iteration.

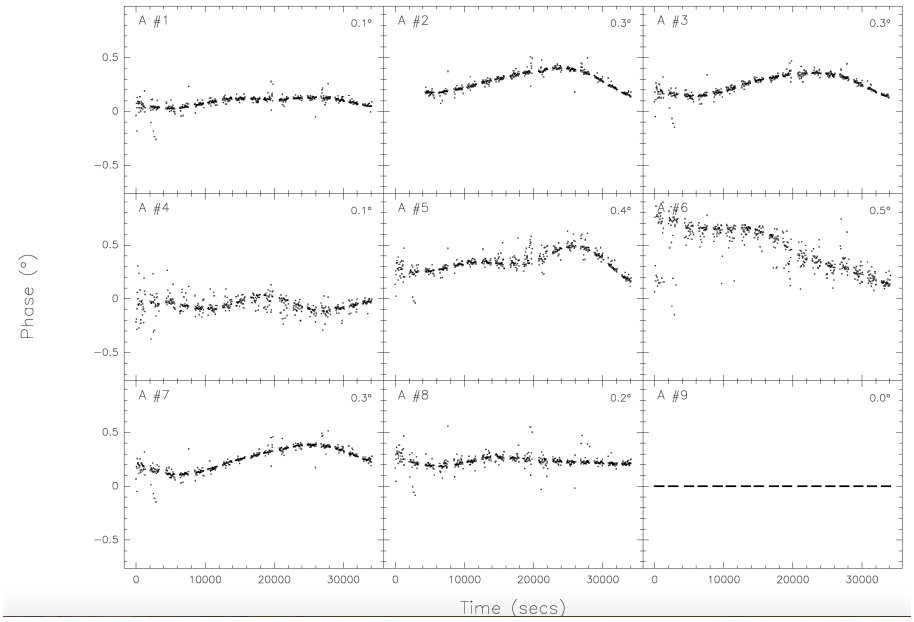

Figure 13: The SELFCAL SHOW output after a phase calibration, showing the convergence of the corrections between the last 2 iterations.

The displayed range can be controlled by variables SELF_PRANGE[2] for the Phase, SELF_ARANGE[2] for the Amplitude, and SELF_TRANGE[2] for the Time. Error bars are displayed if ERROR_BARS is YES, as shown in Fig.15 for amplitude.

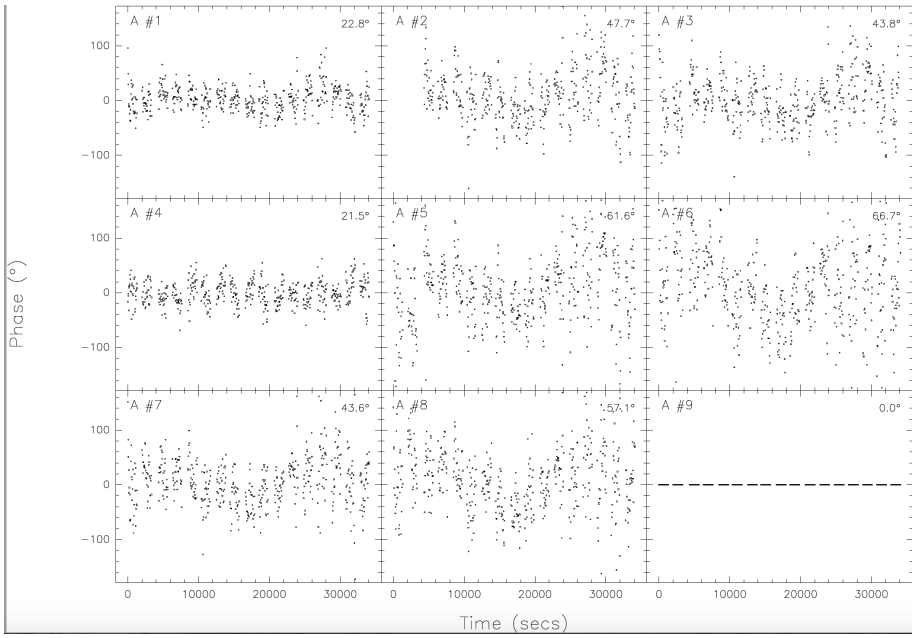

Figure 14: The SELFCAL SHOW 4 output after a phase calibration, showing the difference between iteration 4 and the original data.

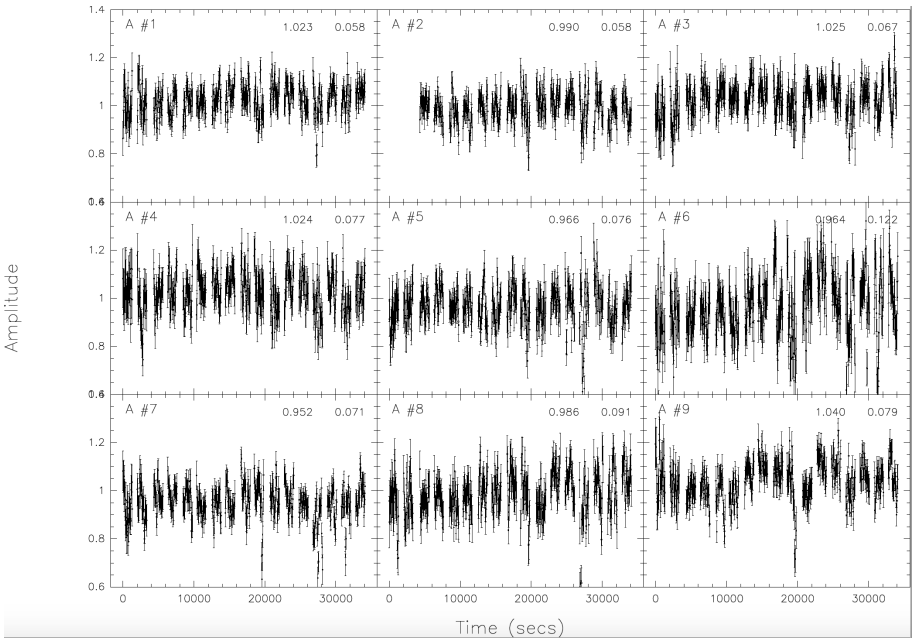

Figure 15: The SELFCAL SHOW 4 output after an amplitude self calibration, showing the difference between iteration 4 and the original data, with the error bars.

13 The Imager PIPELINE tools

With ALMA or NOEMA, you may end up with an observational data set that contains several sources, each of them observed with a number of spectral windows of different spectral resolutions, covering many spectral lines.

The bookkeeping of such data sets can be intricate. To help the users to focus on the science, we have developped a pipeline (namely a suite of scripts named `all-*.ima`) that automates the whole imaging process in this case. This pipeline can be run through the `PIPELINE` command, or through a control panel launched by the `PIPELINE /WIDGET` command.

The pipeline control parameters are available in the `ALL%` global structure, and can be checked using the `PIPELINE ?` command.

The pipeline is intended to handle all spectral sub-bands for a single receiver tuning of the interferometer, with the data (possibly with several observing dates) for each sub-band and each source stored in a separate file (in UVFITS or GILDAS UV format).

The principle is to gather all UV tables in a sub-directory (named `./RAW/` by default), and to store the results of the various processing steps (Self Calibration, Spectral line extraction, Imaging) in different sub-directories (respectively named `./SELF/, ./TABLES/, ./MAPS/` by default).

The behaviour of the Imaging step depends on the pipeline mode, `ALL%MODE`, that can be controlled by the `PIPELINE /MODE` option. It also depends on whether a `CATALOG` is used or not, and on the `ALL%RANGE[1:3]` variable that controls the velocity span and resolution.

- CONTINUUM Mode:
  In this mode no data cube is produced. Only continuum images are created, with no attempt to remove any contaminating line emission.

- SURVEY Mode
  In this mode, there is also no spectral line emission filtering. Full data cubes are produced, at the spectral resolution specified by `ALL%RANGE[3]`. No continuum image is produced: continuum can be later extracted through the `MAP_CONTINUUM` command.

- SPLIT Mode

  - Without a spectral line catalog, the Imaging step will make no attempt to identify spectral lines, and thus no attempt to image each line separately. Instead, each "high spectral resolution" UV table will be imaged completely. An automatic estimate of the continuum level is made, and is imaged separately from the continuum-subtracted data. At the end of the process, the deconvolved "continuum" image is added back to the spectral image."'Low spectral resolution" data only produce a continuum image. This mode is appropriate for e.g. low spectral resolution data, very wide lines, and/or nearly confusion limited data with many spectral lines.
  - With a line catalog, a user-specified velocity range is imaged separately around each line in the catalog that falls into the observed frequency coverage. Continuum and Line are separated, but no attempt is made to add back the continuum data at end.

- ALL Mode
  This mode is similar to the SPLIT mode, except that no attempt is made to produce continuum-free line data cubes. The line data cubes also include the continuum data. This mode is appropriate when the continuum emission has spatial variations of its spectral index.

Naming conventions apply to identify which data set covers which spectral line.

13.1 Preparing the data

For NOEMA, the UV data can be created from the `.IPB, .hpb` raw data files using the CLIC script `@ all-tables`. The resulting UV tables (`.uvt` files) should be placed in your working directory.

For ALMA, UVFITS files should be created from the original Measurement Set (`.ms` directory) using the `casagildas()` Python tool in CASA, and placed in your working directory.

Once your working directory is loaded with the `.uvt` or `.uvfits` files, you can start using the `all-widget` script, usually through the `PIPELINE /WIDGET` command.

It is recommended to separate data from different sources into different directory structures, to simplify further processing.

13.2 The PIPELINE Widget

This section describes the widget that can be used to control the image processing in a systematic way for an ensemble of UV tables. The (selected) UV tables must come from observations of the same source **and** correspond to a single frequency setup, so that all spectral windows share the same *uv* coverage and observing dates and times.

13.2.1 Getting started: PIPELINE /WIDGET

`PIPELINE /WIDGET` creates a Widget that is used to customize the process and launch the various steps. It also creates (through a call to `@ all-create`) the `ALL%` structure and its components.

13.2.2 ORGANIZE step

The `ORGANIZE` button will move the initial files (`.uvfits` and `.uvt` files) into an appropriate sub-directory structure. By default, `.uvfits` files will be in `./UVFITS/` (name controlled by `all%uvfits`) sub-directory, while `.uvt` files will be in `./RAW/` sub-directory (name controlled by `all%raw`).

If absent, the `.uvt` files will be created from the `.uvfits` ones, using default parameters (i.e. assuming unpolarized emission).

13.2.3 FIND step

The `FIND` button will explore the directory containing the initial UV tables (`./RAW/` by default) to identify *Wide Band* UV tables, i.e. those that cover enough bandwidth to provide enough sensitivity for self-calibration on continuum flux.

This information will be used later to derive the self-calibration solutions from some UV tables, and apply it to others, typically narrow band spectral line data.

13.2.4 SELECT step

The `SELECT` button will restrict the subsequent work on the files matching the `"File Filter"` field value.

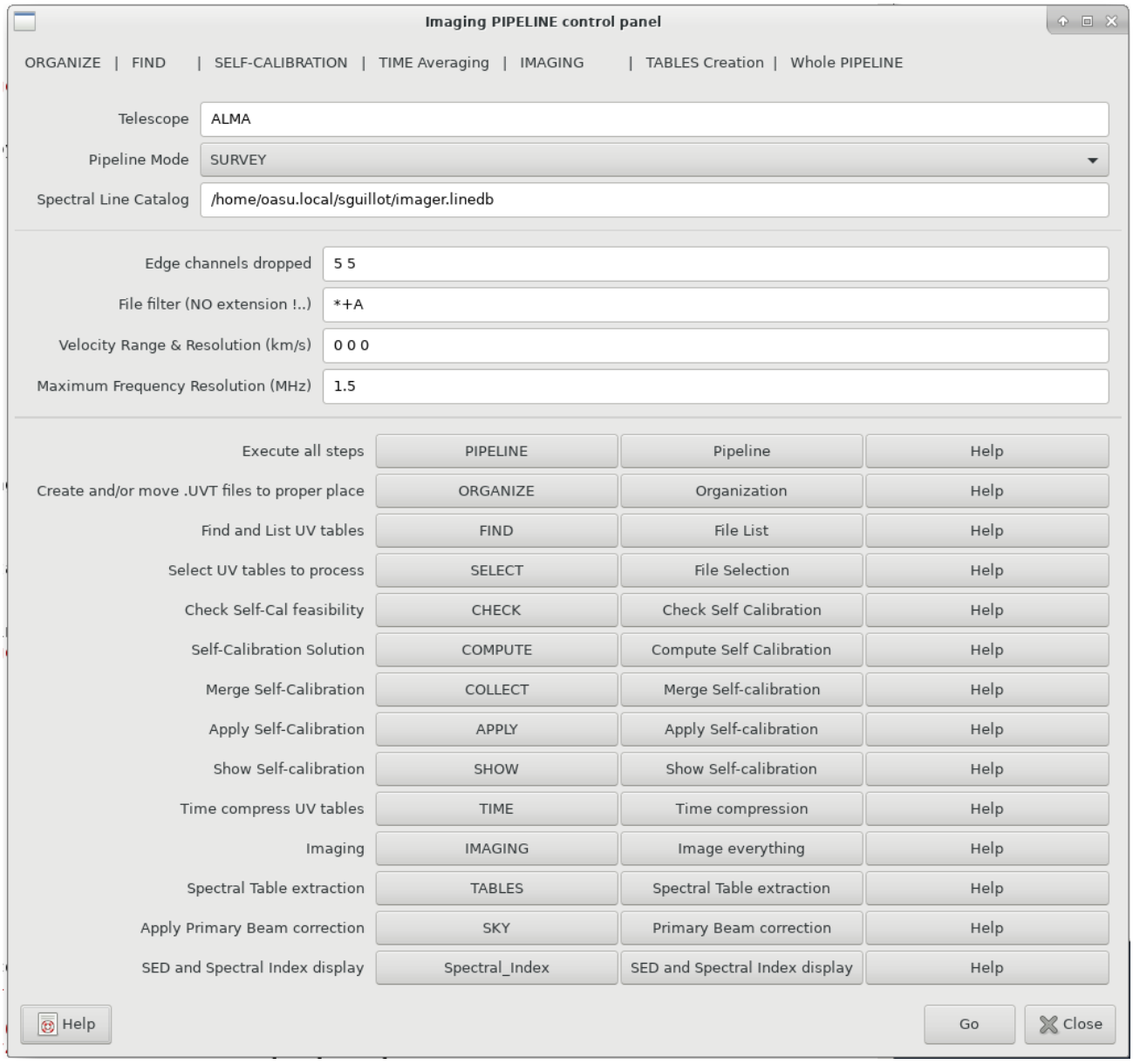

Figure 16: The Pipeline control widget. The two first lines of the widget cannot be modified directly. The `Telescope` information is derived by button `ORGANIZE` from the data set. The `Spectral Line Catalog` is defined through command CATALOG.

13.2.5 `CHECK` step

The `CHECK` button will use the list of *Wide Band* UV tables to check whether Self calibration may be needed and possible. The method is currently approximate and will be improved in future releases.

13.2.6 `SELF` step

The `SELF` button will use the list of *Wide Band* UV tables to compute a (phase and amplitude) Self calibration solution, and apply it to all UV tables. It will identify which UV tables correspond

to a given Wide Band one, so that the proper self calibration solution is applied.

The self calibrated UV tables are placed in another sub-directory (`./SELF/` by default, controlled by `all%self`).

A prefix (controlled by `all%prefix_self`) can be added to the file names to remind the user that they have been self-calibrated.

Caveats: CURRENT LIMITATIONS
- No provision is made to use spectral line flux when the continuum flux is too low.
- Thus, if the Self-calibration solution is not good, you should not apply it. The `COLLECT` step may help at this stage.

13.2.7 `SHOW` step

This step will display the phase and amplitude correction that have been computed in the Self Calibration step. Optionally, these plots can be saved on Hardcopies (`.pdf` files for Pdf, `.eps` for EPS).

13.2.8 `COLLECT` step

Most NOEMA data include several wide bands (4 with PolyFix in its standard mode), and an independent Self-calibration solution is computed for each of them. The `COLLECT` button will merge all these solutions into a single one, assuming the phase errors are due to pathlength changes.

This improves the S/N of the self-calibration solution, and can be beneficial when the S/N is the limiting factor. On the other hand, if the data has very high S/N, this step may result in lower dynamic range than the application of the separate self-calibration solutions.

13.2.9 `APPLY` step

The `APPLY` button will apply the Self-calibration solution found in the previous steps (`SELF` and optionally `COLLECT`) to all UV tables. The selection of the solution to be applied is based on matching frequencies between the UV table and the solutions.

If the `COLLECT` step has been performed, one may still use the individual solutions by setting variable `ALL%COMBINE` to `NO`.

13.2.10 `TIME` step

The `TIME` button will time average the self calibrated UV tables to save space and speed up further processing.

The integration time can be specified, or left to 0. In this case, the data will be time-averaged to an appropriate sampling time given the longest baselines and field of view.

13.2.11 `IMAGING` step

The `IMAGING` button will scan all spectral windows to identify whether they can be used to produce a Continuum or spectral line image.

Low resolution spectral windows, identified as those whose resolution is coarser than `ALL%MINFRES` will be used to produce continuum images, by filtering any detected spectral signature in the data before.

High resolution spectral windows, identified as those whose resolution is better than ALL%MINFRES, data cubes will be produced.

In SPLIT Mode, if a Line Catalog is present, these high spectral resolution windows will be scanned for line identifications. For each spectral line in the current catalog(s) (defined by command CATALOG) that fall in a spectral window, a continuum-free UV table will be created, covering the velocity range specified by the user (by variable ALL%RANGE) around the line frequency. The naming convention is the following:

```
original-molecule-I-X
```

where
- original is the name of the initial high resolution spectral window
- molecule is the name of the spectral line in the catalog
- I is a sequence number, incremented each time a new line is found from the same original UV table.
- X is a character code, equal to D if the line is suspected to be detected, and to U if not. When several lines are too close together, the D status may be incorrectly affected, but this is just a naming convention, not a strong result.

In addition an

```
original-C
```

file that contains (presumably) line-free emission only is created for each original UV table.

The imaging results are stored in a specific sub-directory (./MAPS/ by default, controlled by all%maps). Only the Clean Component Table (.cct) and Clean image (.lmv-clean) files are written.

When no Line catalog is present, only 3 images are produced for each UV tables: starting froml the original name

```
original-C       original-U    and   original-A
```

The ...-C contains (an estimate of) line-free emission, averaged over the bandwidth of the UV data. The ...-U is the continuum subtracted image, while in ...-A the pseudo-continuum image obtained in ...-C has been added back. The ...-A thus contains all the signal, and can be used to provide a better estimate of the continuum level using e.g. the MAP_CONTINUUM command.

Spectral resampling can be performed by appropriate setting of ALL%RANGE. If non zero, ALL%RANGE[3] indicates the desired spectral resolution. Furthermore, in the CATALOG case, ALL%RANGE[1:2], if non zero, specify the Min and Max velocities imaged around each spectral line.

In ALL mode, the behaviour is similar, but only the -C and -A products are created.

In CONTINUUM mode, line emission is assumed to be negligible, and only the -C continuum images are produced.

13.2.12 TABLES step

The TABLES button performs the same scanning and identification process as the IMAGING button, but stores the resulting UV Tables (instead of the .cct and .lmv-clean files for the IMAGING button) in a sub-directory, named ./TABLES/ by default, controlled by all%tables.

This step is optional, and only needed if the user intends to perform some UV data analysis (like UV plane fitting, line stacking, continuum spectral index determination, direct modeling, etc...).

Extraction of the velocity range specified by ALL%RANGE[1:2] is performed if a CATALOG is provided by the user. Spectral resampling according to ALL%RANGE[3] is performed if and only if ALL%RESAMPLE is set to YES. Without no CATALOG, the full spectral coverage is preserved.

Except in CONTINUUM mode, line-free continuum UV tables are also produced.

13.2.13 SKY step

The SKY button will apply primary beam correction to the deconvolved images. This button has no effect for Mosaics, as sky brightness distributions are directly produced in this case.

13.3 The non interactive PIPELINE

The whole series of actions available through the Pipeline Widget can be performed in series through the PIPELINE command.

Command PIPELINE * will perform all the actions, without any user interaction. The PIPELINE button in the Pipeline widget will do the same (and also the GO button, as usual in GILDAS widgets).

A specific step can be performed by command PIPELINE StepName. The available step sequence is ORGANIZE, FIND, SELECT, COMPUTE, COLLECT, APPLY, SHOW, TIME, TABLES, IMAGE, SHOW and ultimately SAVE to remember the parameters used in the processing. PIPELINE LAST will repeat the last step (presumably after you changed a control variable !...) while PIPELINE NEXT will execute the next one.

13.4 Mosaics and other limitations.

Mosaics: The pipeline handles mosaics, but with several limitations. First, the data has to fit in memory. This is often the case for NOEMA data, but may not be true for ALMA data. Second, there is no provision to add short spacings: if this is needed, it should be done for each *uv* table prior to using the pipeline. Last, Self-Calibration is ignored for Mosaics, although there could be cases where sufficient S/N is available for this.

UV table consistency: Also, the PIPELINE is intended for simultaneous observations with a single tuning and single source. Although multiple sources can be handled by using the appropriate file filter, it is recommended to process them in separate sub-directories.

Non simulaneous observations can lead to puzzling inconsistencies, as the self-calibration from one date might be applied to the wrong setup, flagging that data!

14 Really Huge Problems

IMAGER basically assumes everything fits into memory. This is in general quite fine for NOEMA data. However, for ALMA data, if you are working with a too small computer (such as my laptop, which is otherwise fine), you may be lacking physical memory (RAM, Random Access Memory), and IMAGER may become really inefficient by using Virtual Memory instead.

To avoid time losses, IMAGER prevents reading UV data whose size exceeds the available RAM, and warns the user if it exceeds half of the RAM. To treat these cases, IMAGER provides instead a number of tools that can work sequentially on the data set, instead of loading it in memory all at once.

1. Working by subsets:
   the READ /RANGE command allows to select an ensemble of channels from a UV data. If this ensemble is small enough, IMAGER can work. At the end the WRITE /APPEND and WRITE /REPLACE command will allow to put these channels at their proper places in a deconvolved data cube.

2. Working on UV data files:
   Some operations on UV data, like time averaging, separation of line from continuum emission, or self-calibration, and of course, spectral resampling, are best done using all valid channels to avoid loosing sensitivity. To allow IMAGER to do them even for large data files, most UV-related commands have a /FILE option which instructs the command to work from the corresponding data file, instead of the UV buffers. This includes UV_PREVIEW /FILE, UV_BASELINE /FILE, UV_FILTER /FILE and especially the UV_SPLIT /FILE commands. Time averaging can be performed by UV_TIME /FILE, and prior sorting by time order can be done by UV_SORT /FILE. Spectral range extraction is possible by UV_EXTRACT /FILE, spectral resampling by UV_COMPRESS /FILE, UV_RESAMPLE /FILE, and UV_HANNING /FILE.

By using the above commands, all operations can be done in a quasi sequential way, avoiding to load in memory whole data sets.

Two commands have no equivalent using the IMAGER buffers, and work only on files. Their /FILE option is used to provide an homogeneous syntax, but must be present for the command line to be valid.

1. the UV_MERGE /FILE
   that allows to merge together an arbitrary number of UV tables, in spectral line (with spectral resampling) or continuum (with flux scaling according to a spectral index) modes.

2. the UV_SPLIT /FILE
   that combines the capabilities of UV_BASELINE and UV_FILTER in a single command, since both operations require the same parameters and provide complementary informations.

The UV_SORT /FILE command also has a different behaviour than its memory only version UV_SORT. While the latter creates a transposed version of the UV table for internal use (it **cannot** be saved), the former keeps the normal organisation with the visibility axis first.

An example of use of such facilities is the @ image-mosaic script that splits the uv_map step into chunks that fit in the computer memory available to the user.

15 Polarization Handling

IMAGER recognizes and handles polarization at different levels. Although support for polarization is basically complete for continuum data, it remains **experimental, and is continuously improving**.

Contact the IMAGER authors if you need to analyze Polarized data, using Mail to: `imager-hotline@services.cnrs.fr`

15.1 Data Handling

Importing data

When importing data, the `fits_to_uvt` script assumes by default the data is unpolarized and produces the pseudo-polarisation state "None" from the UVFITS file, by a properly weighted combination of the two parallel hand states if more than one state is present. The weighting is based on the relative noise in both polarization states.

Full polarization information can be preserved by adding the `/STOKES` option to the `@ fits_to_uvt` command.

The `READ UV` command will read data with any polarization state(s).

Processing data

On the contrary, practically all IMAGER commands cannot handle more than 1 polarization state. So far, only two commands fully support polarized data: `UV_TIME` and `STOKES`.

- `STOKES` is the primary command that allows to derive or extract a UV data with only one Stokes parameter from a UV data set with several polarization states. IMAGER can then process the individual Stokes parameters separately.
- For convenience (because polarized data is obviously in general bigger), the `UV_TIME` command can be used for time averaging prior to use of the `STOKES` command.
- Most other commands will flatly refuse to handle data with more than 1 polarization state (e.g. `UV_FILTER`, `UV_RESAMPLE`, etc...).
- For debugging purpose, some commands like `UV_PREVIEW`, `UV_MAP` or `UV_STAT` will operate with more than 1 polarization state, but will not produce meaningful results (only a subset of the data may be treated).

Some imaging strategies cannot be used directly on polarized data. Also, the automatic definition of supports (`MASK /THRESHOLD`) only makes sense on Stokes I or parallel hand polarization states, not on Stokes Q,U,V or cross-hand states. The same applies to spectral line identifications.

PIPELINE processing The PIPELINE tools does not automatically handle polarization data. However, through judicious use of the `PIPELINE SELECT` and `PIPELINE FIND` commands with ad-hoc filters, imaging, including self-calibration, is possible through custom (though simple) scripts.

The script `self-polar.ima` just does that, and follows the naming conventions used by the `MAP_POLAR` command. It contains the following commands

```
CATALOG               ! No catalog, no Line emission
! "Split the X+Y (XX,YY,XY,YX) UVFITS file into I,Q,U,V Stokes UV tables"
@ stokes-split &1
```

```
! Setup the pipeline
IMAGER\PIPELINE ?
IMAGER\PIPELINE organize        !  " Move UV tables to RAW/"
sic mkdir UVFITS
SIC\SYSTEM " mv *.uvfits UVFITS/"
SIC\SYSTEM " mv RAW/&1.uvt .."  !   "Remove the X+Y UV table from RAW/"
!
! Select the I image for self-calibration
let all%filter *-I
IMAGER\PIPELINE select
IMAGER\PIPELINE compute
!
! Apply self-calibration solution to I,Q,U,V data
let all%filter *
IMAGER\PIPELINE find
IMAGER\PIPELINE select
IMAGER\PIPELINE collect
IMAGER\PIPELINE apply
! Image every thing consistently
IMAGER\PIPELINE image
PIPELINE SAVE
```

Analysing data

The MAP_POLAR command allows to derive images of the Polarized intensity and/or fraction and Polarization angle, as well as display of the polarization vectors onto a background image. So far, the command only works for a single plane image.

The MAP_POLAR command works through data files that follow conventional naming rules to identify them. The implicit rule is that Stokes S (where S is any of I,Q,U,V) is stored in a file of name 'NameBegin'"-N"'NameEnd', e.g. if 'NameBegin' is My_Data

```
My_Data-I+C.uvt,  My_Data-U+C.lmv-clean, etc..
```

15.2 The STOKES command

The STOKES command operates on the current UV buffer, or on files if the /FILE option is present. It allows to extract a UV data with visibilities for one output (pseudo-)Stokes parameter from UV data with visibilities with 1,2 or 4 (pseudo-)Stokes parameters. Besides the standard Stokes parameters I, Q, U, V, RR, LL, RL, LR (Left and Right circular), XX, YY, XY and YX (X and Y linear) which are defined in the Sky frame[10], command STOKES recognizes pseudo-Stokes values NONE, HH, VV, HV and VH which are the linear polarization states in the frame of the antennas (Horizontal and Vertical pure states).

Conversion from the H-V pseudo-Stokes polarization states to any standard Stokes parameter is made by the STOKES command by applying the rotation due to the parallactic angle. For this, the UV data set must contain the PARA_ANGLE extra column. If it is not present, it can be added

[10] ultimately, a similar DD, SS, DS and SD pseudo-Stokes polarization set may be added for circular polarization. D is the first letter for Dexter, the Latin name for Right, while S is the first letter of Senester, the Latin name for Left.

to the data set by command UV_ADD /FILE. That command can also insert the Doppler correction as an extra column.

Script stokes-split can be used to split a full polarization UV table in 4 UV tables, one for each IQUV Stokes parameter, following the standard naming convention expected for final image use by MAP_POLAR.

16 Advanced Data Analysis

Besides basic interferometric imaging, IMAGER offers a number of advanced tools that can be used to extract more information from the available data than just by visual inspection of the deconvolved images.

These tools are particularly useful for weak signal detection. Some of them are generic (e.g. the Matched Filtering), others specific of some astrophysical situations (e.g. the KEPLER tool)

16.1 Matched Filtering

Weak signal detection is always an issue in most astrophysical cases (e.g. high redshift galaxies, faint lines in proto-planetary disks, to cite some).

For spectral lines, IMAGER offers the UV_DETECT command, that uses an image model as an optimal filter to enhance weak line detection. This is a generic tool, that can use an observed data cube (from a strong line, presumably) as the image model, or the result of a simulation.

16.2 Line Stacking

Another aspect is spectral line stacking, when several lines are assumed to follow the same spatial distribution. This can be done using command UV_MERGE /MODE STACK. The stacked UV table result can be imaged and analyzed as usual. Like UV_DETECT, UV_MERGE does not make any assumption on the source shape.

16.3 Rotationally symmetric (thin) structures: Keplerian disks

The KEPLER command re-aligns in velocity the spectra as a function of radius (distance from the rotation center), by correcting from the projection effect and radial velocity law. This allows to stack spectra coming from the same radius, and thus to derive radial profiles.

The result is a Position-Velocity image, and can be displayed conveniently with the SHOW KEPLER command.

The KEPLER command assumes by default that the velocity law is Keplerian. However, arbitrary analytic laws can be used (see KEPLER /VELOCITY, so that the tool can also be used for example for spectral lines in galaxies.

16.4 Image Handling and Comparison

The MAP_* suite of commands (MAP_COMBINE, MAP_COMPRESS, MAP_INTEGRATE MAP_REPROJECT, MAP_RESAMPLE, MAP_SMOOTH and MAP_CONTINUUM) allow basic operations on Data Cubes.

Command COMBINE simplifies Data cube combinations, by combining individual steps that are otherwise available in MAP_REPROJECT, MAP_RESAMPLE, MAP_COMBINE, while command VIEW /OVERLAY allows visual comparison of two data cubes.

17 DISPLAY Language Internal Help

17.1 Language

```
DISPLAY\ Language summary
```

The DISPLAY\ Language contains commands to display data cubes and results in a flexible, interactively controlled or scripted way. Available commands:

```
CATALOG       : Define the spectral line catalog to identify transitions
COLOR         : Control the color LUT to highlight the Zero level
FIND          ! Search for spectral lines in the specified range
LOAD          : Read a file in the DATA area for further display
EXPLORE       : Explore a data cube: make a plot of spectra around a 2-D map
EXTRACT       : Extract a subset from a data cube
INSPECT_3D    : Visualize any data cube by cuts along 3 main directions
POPUP         : Enlarge one panel of a display
SET           : Set some IMAGER or GreG parameter
SHOW          : Display (in a plane by plane plot) any data cube or
                other results from the IMAGER program.
SLICE         : Extract a Slice of the specified data cube
SPECTRUM      : Compute integrated spectrum from data cube
STATISTIC     : Compute Statistics on specified data
VIEW          : Show (in a comprehensive plot) any data cube
```

Commands EXPLORE, INSPECT_3D, SHOW and VIEW remember and share the same input data for their display (e.g. SHOW CLEAN, followed by EXPLORE will work as EXPLORE CLEAN).

17.2 CATALOG

```
[DISPLAY\]CATALOG [FileName [.. [FileNameN]]
```

Define or list the current catalog(s) for spectral line identification. A catalog is a file in the "Astro" GILDAS format, as used in programs ASTRO or CLASS in the GILDAS suite, or in the LINEDB data format. Without argument, or with argument ?, the command will just list the names of the current catalog(s).

There can be only ONE Astro catalog connected at a time, and it is exclusive of any LINEDB data bases. On the opposite, there may be several LINDEB data bases specified by the CATALOG command.

At initialization, IMAGER search for $HOME/imager.linedb as default catalog, then gag_data:imager.linedb if not found. If the specified catalog does not exist, no spectral line identification will be done.

Direct use of the LINEDB\USE command is also possible to define the

LINEDB catalog(s), though this is not recommended (possible conflicts with an Astro catalog).

17.2.1 CATALOG Default

A default catalog can be created (in the local directory) by executing once

```
@ gag_data:imager-linedb.sic
```

This may take a while, and even get stuck occasionally because it performs network access to remote databases. You can later copy this catalog to your $HOME. See HELP LINEDB\ to add or remove species from such catalogs.

17.2.2 CATALOG Astro

The ASTRO catalog format is one of the two following ones:

A comma separated value file, including the frequency in MHz, the species name, and the transition name, e.g.:

```
150E3; MOLE; TRANSITION;
```

A list-directed file, with the same information, e.g.:

```
150E3 'MOLE' 'TRANSITION'
```

Lines beginning with an exclamation mark are considered as comments. The first line **MUST** be a comment line.

A default catalog (with no implied warranty) is given in gag_data:molecules.dat.

17.2.3 CATALOG Linedb

The LINEDB catalog format is a more flexible facility, which can also holds additional information. On-line spectral databases (JPL and CDMS) are directly useable as LINEDB catalogs, although access can be long in this case.

Scripts named gag_data:*-linedb.sic are availables as examples to construct local data bases from the on-line ones. Beware that the on-line databases sometimes change their naming conventions, so that identifying a given species by name can be tricky.

In 2020, access to CDMS was unreliable. It has improved since. Access to JPL is operational, but frequencies are not as up-to-date as in CDMS.

Local files in the JPL ".cat" format can be used, but as of Jan-2020, the Einstein coefficients are wrong. They also require a "partfunc.cat" in the following format (given here for HN-15-C as an example)
species HN-15-C 28006
temperatures 300. 225. 150. 75. 37.5 18.75 9.375
qpart 141.06 105.9 70.797 35.514 17.23 9.135 4.746
The tag number (28006 here) must match that of the ".cat" file.

17.2.4 CATALOG LINEDB%ENERGY

Maximum energy of the upper state of the selected transitions, in K. This applies only to LINEDB data bases.

17.3 COLOR

[DISPLAY\]COLOR HueRange [HueZero HueExtent]

Select a color Look-up-Table so that the color at value Zero is given by HueZero (in range [0,360[), and the overall HUE extent is HueExtent (which can be larger than 360). HueRange (in the same units) indicate a part of the LUT which becomes progressively whiter (less saturation, HueRange < 0) or darker (less intensity, HueRange > 0) around the colour HueZero.

Such LUTs allow to outline the Signal-containing regions compared to the noise level, either in bright (HueRange < 0) or dark (HueRange > 0) modes.

17.4 EXPLORE

[DISPLAY\]EXPLORE [DataCube] [/ADD Cx Cy [Box]] [/NOPAUSE]

Create a plot with a 2-D map in the middle, and up to 8 spectra around it.

DataCube is the name of an internal 3-D buffer, SIC variable or data file. The 2-D map will be either a channel of this 3-D data set, or a velocity integrated map obtained from it. It will displayed at the center of the plot. If not present, argument DataCube keeps its previous value.

Spectra can be displayed around, with arrows showing their position in the map, either interactively (when the /NOPAUSE option is not present), or manually through the /ADD option. In interactive mode, the user can press S to extract and display the spectrum from any place in the 2-D map.

EXPLORE ? will list the control variables for this command. The usual SIZE and CENTER control the 2-D map size, RANGE controls the velocity range, and SCALE the spectral flux range.

17.4.1 EXPLORE /ADD

[DISPLAY\]EXPLORE /ADD Cx Cy [Box]

Add the spectrum coming from positin Cx Cy (in arcsec) in the specified Box number. The Box number indicates the placement as in a numeric keypad layout. Box 5 (the center) is reserved for the 2-D map.

If Box is not present, the spectrum is placed in the most plausible position according to the coordinates (Cx,Cy).

17.4.2 EXPLORE /NOPAUSE

[DISPLAY\]EXPLORE [DataCube] /NOPAUSE

Just display the 2-D map and any already existing spectra, but do not call the cursor.

17.4.3 EXPLORE Variables:

Like SHOW, VIEW and INSPECT_3D, EXPLORE is controlled by variables. Most of them are among those controlling SHOW:
SPAN, RANGE, SIZE, CENTER, SCALE, SPACING, DO_BIT DO_GREY, DO_CONTOUR, DO_MASK.

In addition, the SFLUX variables controls the displayed flux scale in the map.

17.4.4 CENTER

REAL CENTER[2]

Specify the relative coordinates of the center of the displayed region,

in ANGLE_UNIT defined by SET ANGLE_UNIT command.

17.4.5 DO_BIT

LOGICAL DO_BIT

Show a bitmap color plot (YES) or do not (NO) for each panel. The bitmap range is controlled by variable SCALE.

17.4.6 DO_CONTOUR

LOGICAL DO_CONTOUR

Draw (YES) or do not (NO) draw the contour level, which are controlled by SPACING and SPACE_TYPE. This is provided for compatibility with Mapping and Greg.

17.4.7 DO_GREY

LOGICAL DO_GREY

Do (YES) or do not (NO) use grey-scale contouring. Note that this is independent of DO_CONTOUR.

17.4.8 DO_MASK

LOGICAL DO_MASK

Hide part of the plot (contour or bitmaps, and even axis ticks) that lie in the upper left corner of each panel to have a clean display of the value specified by MARK.

17.4.9 RANGE

REAL RANGE[2]

Specify the range to be displayed for the 3rd axis, as well as the displayed range of the integrated spectrum.

RANGE should not be confused with SPAN, which specifies the displayed velocity range in the two PV diagram panels.

2 SFLUX

```
Character*32 SFLUX
```

Specify the flux (intensity) range for the spectra. SFLUX can be * or 0 to indicate a fully automatic scale, or indicate a Min Max, each of which possible * to indicate a automatic limit.

17.4.10 SIZE

```
REAL SIZE[2]
```

Specify the displayed are size, in ANGLE_UNIT (see SET ANGLE_UNIT command).

17.4.11 SCALE

```
REAL SCALE[2]
```

SCALE indicates the Colour Scale Range, range of image units which are displayed in the color bitmap. SCALE = 0 implies an automatic evaluation of this range, based on the Min and Max of the data cube.

17.4.12 SPACE_TYPE

```
CHARACTER SPACE_TYPE
```

```
 Define the type of spacing for the contour levels
   LIN       Linear spacings. The Zero contour is always omitted.
   LOG       Exponential spacing
   NONE      No contour levels at all (same as DO_CONTOUR = NO)
   USER      Keep the contour levels specified by command LEVELS
For LOG case, see HELP LEVELS EXPO in the GreG internal help.
```

As a backward compatibility, a negative spacing is a convention to use the current contour levels, although using SPACE_TYPE = USER should be preferred.

17.4.13 SPACING

```
CHARACTER*16 SPACING[2]
```

SPACING indicates the step (and implicitely the contour levels) for contouring and/or greyscale. SPACING[1] control the values, and SPACING[2] the Unit of these values. It can be UNITS (default data cube units), NOISE or SIGMA (noise level), Jy (which also stands for Jy/Beam) and its sub-units mJy (or milliJy) and microJy, or K and its sub-units mK (or

milliK) or microK.

Jy (and its sub-units) is taken as equivalent to Jy/Beam, and automatic conversion between brightness unit (K) and flux unit (Jy/Beam) is made when the data cube has a specified angular resolution and frequency.

SPACING[1] = 0 means an automatic guess of the contours from the Min and Max of the data cube.

The type of spacing is controlled by variable SPACE_TYPE.

17.4.14 SPAN

REAL SPAN[2]

For EXPLORE, SPAN indicates the limits of plot for the spectra.

17.5 EXTRACT

[DISPLAY\]EXTRACT VarName [blc1 blc2 blc3 trc1 trc2 trc3]

Extract a subset of the image variable VarName and put it in the EXTRACTED image variable.

Without argument, the region covered by the subset is defined by the SIZE, CENTER and RANGE values. In this case, the EXTRACTED variable dimensions are automatically derived from this values, and EXTRACTED can be bigger than the actual cube defined by VarName. Areas not covered by the initial VarName are filled out with the Blanking value of the VarName, or by zeros if VarName has no blanking.

If the arguments blc1 to trc3 are specified, the subset corresponds to VarName[blc1:trc1,blc2:trc2,blc3:trc3] of the image variable VarName. The BLCi and TRCi values must correspond to a valid subset.

17.6 FIND

[DISPLAY\]FIND Fmin Fmax [/SPECIES Name]

Select spectral lines in the range Fmin to Fmax (in MHz) and return them in the LINES% structure.

LINES%SPECIES contains the name of the species for each line. LINES%FREQUENCY holds its frequency in MHz. LINES%LINES contains the full line names, including the transition. LINES%PLOT contains the

string to be used in graphic display, with codes for superscripts and subscrips when needed.

The sign of LINES%N encodes the number of selected lines and type of catalog which was used.

The rest of the structure depends on the catalog type.

If the catalog is in the ASTRO format, LINES%N is negative and contains the opposite of the number of lines.

If the catalog is in the LINEDB format (see HELP LINEDB\ for details), LINE%N is positive and contains the number of selected lines. Quantum numbers, energy levels and line strengths are also available in this case, allowing a more precise selection.

17.6.1 FIND /SPECIES

[DISPLAY\]FIND Fmin Fmax /SPECIES Name

Only select lines from species that match the specified Name. Only one Name can be specified, and the match is case sensitive. There is (limited) support for wildcard so far.

17.7 INSPECT_3D

[DISPLAY\]INSPECT_3D [Argument [FirstPlane [LastPlane]]]

where
Argument
is the the name of a 3-D internal buffer to be plotted (BEAM, CCT, CLEAN, DIRTY, FIELDS, MASK, etc..), or any Sic Image variable, or a file name. If not present, Argument falls back to the last argument used in EXPLORE, INSPECT_3D, SHOW, or VIEW commands.

Simultaneously display cuts along each of main cube directions, as well as the current spectrum. Coordinates are controlled by the cursor position in the plots.

17.7.1 INSPECT_3D History

INSPECT_3D is a command similar to GO 3VIEW in Mapping, but with a simpler syntax and the ability to display program buffers as well as files.

17.7.2 INSPECT_3D Variables:

Like SHOW, VIEW and EXPLORE, INSPECT_3D is controlled by variables, selected among those controlling SHOW:
SPAN, RANGE, SIZE, CENTER and CROSS.

17.7.3 SPAN

```
REAL SPAN[2]
```

SPAN indicates the limits of 2-D plot along the velocity axis, while RANGE indicates the 3rd axis range that is displayed in the integrated spectrum. The spatial dimension (if any is displayed) is controlled by SIZE and CENTER.

If SPAN is 0, it defaults to RANGE. If RANGE is zero, all planes/channels are displayed.

17.8 LOAD

[DISPLAY\]LOAD File [/FREQUENCY NewFreq] [/PLANES Start End] [/RANGE Start End Type]

Read the specified File (GILDAS or FITS data cube) into the DATA buffer, for further display.

17.8.1 LOAD /FREQUENCY

```
[DISPLAY\]LOAD File /FREQUENCY RestFreq [/RANGE Min Max Type]
```

Read the file into the DATA area and reset the velocity scale to the corresponding rest frequencies. Velocities specified in the /RANGE Min Max VELOCITY option would then refer to this new frequency.

17.8.2 LOAD /PLANES

```
[DISPLAY\]LOAD File /PLANES Min Max
```

Read only the last axis "planes" between the First and Last implied by Min and Max. For an LMV-ordered cube, this would be equivalent to

```
LOAD File /RANGE Min Max CHANNEL
```

However, for transposed data cubes, VML or LVM-ordered for example, this

is not true at all when a Frequency/Velocity axis is present: /RANGE would specify a subset of that axis, while /PLANES always indicate a subset of the 3rd axis of the cube.

Min and Max indicate offsets from Plane 1 and the number of planes Nplane. Thus Max can be negative: it then indicates Last = Nplane-Max. Also Min=0 and Max=0 implies loading all the channels.

17.8.3 LOAD /RANGE

[DISPLAY\]LOAD File /RANGE Min Max Type

Load only the channels between the First and Last defined by Min Max and Type. Type can be CHANNEL, VELOCITY or FREQUENCY. If the Data cube has no Frequency/Velocity axis, only CHANNEL is allowed.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies loading all the channels.

The /RANGE option is not (yet) allowed for FITS files.

17.9 POPUP

DISPLAY\POPUP [Number]

POPUP allows to "pop up" a single plane display out of a multi-box plot. he action driven by POPUP depends on the content of the whole plot.

After a SHOW command, POPUP will call the cursor, and any key stroke (except E to EXIT the loop) will display the corresponding sub-plot in a separate window. The sub-plot may contain an single plane display, a spectrum, a cumulative flux, the phase or amplitude correction from a SELFCAL, etc...

POPUP Number will popup the specified panel number (counted from Top Left Corner down to Bottom Right corner).

POPUP will have no action if the display was not created by SHOW.

17.10 SET

[DISPLAY\]SET KeyWord [Value [...]] [/DEFAULT]

Set an IMAGER or GREG internal value. The only current possible keyword for IMAGER is ANGLE_UNIT. Use HELP GREG\SET for help on the Greg keywords. This command has no impact on axes that are in Angles.

17.10.1 SET ANGLE_UNIT

[DISPLAY\]SET ANGLE_UNIT [Degree|Minute|Second|Radian|ABSOLUTE|RELATIVE]

List or Specify the angular unit and mode used for box labelling. The mode can be ABSOLUTE to use sexagesimal notation in Hour - Degree for Right Ascension and Declination, or RELATIVE to label with offsets in the current angle unit.

SET ANGLE_UNIT
will list the current and fallback units (if any)

SET ANGLE_UNIT ABSOLUTE|RELATIVE
will toggle between ABSOLUTE labels and relative ones in the current angular unit. ABSOLUTE labeling is not possible if there is a rotation angle in the images: only relative labels are used in this case, whatever the mode specified by SET ANGLE_UNIT.

SET ANGLE_UNIT Degree|Minute|Second|Radian
will set the current angle unit to the specified value, and enforce use of offset labels in the specified unit.

17.10.2 SET OtherKeyword

[DISPLAY\]SET OtherKeyword [Value ...]

If the SET argument is not recognized, the command falls back to the GREG\SET command. See HELP GREG\SET for possibilities.

17.11 SHOW

[DISPLAY\]SHOW Argument [Arg1 [Arg2]] [/SIDE]

where
Argument
is a Keyword (e.g. COVERAGE, NOISE, UV_FIT or MOMENTS), the name of an internal buffer to be plotted (BEAM, CCT, CLEAN, DIRTY, etc..), a 2-D or 3-D SIC image variable or datacube filename (in Gildas or simple FITS image format).

[DISPLAY\]SHOW ?

will list the names of recognized keywords and buffers.

Except for UV data where they have a different meaning,
Arg1 and Arg2
are optional arguments to restrict the range of channels to be plotted. They default to FIRST and LAST variable values respectively, and Arg2 defaults to Arg1 if only Arg1 is specified.

The UV data in the internal buffer is the one loaded by command READ UV File and optionally resampled by UV_RESAMPLE, UV_COMPRESS, etc..., or transformed by UV_CONTINUUM. Flagged UV data will appear in a different color.

Command VIEW offers a different style of display for cubes, NOISE and CCT. Command INSPECT_3D offers another style for cubes.

17.11.1 SHOW /SIDE

[DISPLAY\]SHOW Argument [FirstPlane [LastPlane]] /SIDE

Add the <VSIDE display of coordinates and values like in the VIEW command, and loop on the Graphic cursor to explore the data.

This will be the default mode if variable SHOW_SIDE is set to YES.

17.11.2 SHOW History

SHOW derives from the GO PLOT script available in GreG and Mapping. GO PLOT (or its variants GO BIT, GO NICE and GO MAP) and GO UVSHOW offered similar features to those of SHOW, but take data from files or SIC image variables, depending on variables NAME and TYPE.

17.11.3 SHOW Keywords:

SHOW recognizes the following keywords

BEAM	Synthesized beam(s)
CCT	Clean Cumulative Flux
COMPOSITE	Moments + Flux
COVERAGE	UV Coverage
FLUX	Integrated spectra over defined regions
KEPLER	Radial profile and PV diagram of a Keplerian disk
MOMENTS	0th,1st,2nd moments + peak value
NOISE	per channel noise
PRIMARY	Primary beam(s)
PV	Position-Velocity plot

```
  SED         Spectral Energy Distribution
  SELFCAL     Self-Calibration corrections
  SPECTRA     Spectra on a 2-D map-like grid
  SOURCES     Clean component positions
  UV          UV data
  UV_FIT      Results of the UV_FIT command

SHOW calls the appropriate procedures for each action
  p_show_map       for data cubes
  p_show_beam      for BEAM and PRIMARY cubes
  p_show_cct       for CCT (Clean Component Tables)
  p_show_composite for Composite display (Moments + Flux)
  p_show_flux      for FLUX
  p_show_moments   for MOMENTS
  p_show_noise     for NOISE
  p_lmv_map        for PV
  p_show_sed       for SED
  p_show_selfcal   for SELFCAL
  p_show_tcc       for SOURCES (Clean Component Positions)
  p_spectra        for SPECTRA
  p_uvshow_sub     for UV_DATA data
  p_plotfit        for UV_FIT data (UV_FIT results)
```

17.11.4 BEAM

```
SHOW BEAM [PLANE|FIELD Kn] [First Last]
```

Show some sub-cube of the (up to 4-D) dirty Beam.

The 2nd argument indicates whether SHOW shows all Fields for a given frequency plane (PLANE keyword), or all Frequency-dependent beams for a given Field. The 2nd and 3rd arguments are optional if the beam is only 3-D, which occurs for a single field or a frequency-indepedent beam in Mosaics (narrow band approximation).

17.11.5 CCT

```
SHOW CCT
```

Display the Cumulative Clean Flux as function of component number. Pretty useful to see if CLEAN has reasonably converged.

17.11.6 COMPOSITE

```
SHOW COMPOSITE
```

Display the 3 "moments" images derived from a datacube by command MOMENTS, and the integrated line flux computed by command FLUX. The contour spacing for each image is controlled by its own variable:
AREA_SPACING for the integrated area M_AREA
VELO_SPACING for the Velocity field M_VELO (in km/s)
WIDTH_SPACING for the Width field M_WIDTH (in km/s)

As for other image displays, CENTER and SIZE control the center and size of the image (in arcsec). Limits for the Flux spectrum are computed automatically.

17.11.7 CONTINUUM

SHOW CONTINUUM

Display the "continuum" map, i.e. the map obtained by UV_MAP /CONTINUUM and CLEAN.

17.11.8 COVERAGE

SHOW COVERAGE [Ant [Date]]

Displays the UV coverage. Ant and Date are optional arguments indicating which Antenna is to be highlighted, and for which date. Date is a sequential number from 1 to the number of observing dates.

For spectral line UV data, SHOW COVERAGE will only show one UV coverage if FIRST and LAST are set to zero. It will show one per channel otherwise.

17.11.9 FIELDS

SHOW FIELDS

Display the positions of the fields in a Mosaic, by a circle at the half-power primary beam size.

17.11.10 FLUX

SHOW FLUX

Displays the integrated spectra in all regions used by command FLUX.

17.11.11 KEPLER

```
SHOW KEPLER [ALl|BOth|PV|PRofile|SPectrum]
```

Displays the results of the KEPLER command, which can be

PV	Position (radius) Velocity plot
PRofile	Peak intensity radial profile
SPectrum	Integrated spectrum
BOth	Radial Profile and Integrated spectrum
ALl	All of the above

Other combinations are allowed using the KEPLER_SHOW%PROF, KEPLER_SHOW%PV and KEPLER_SHOW%SPEC variables.

Display ranges are controlled by variables KEPLER_SHOW%X, where X = V for the velocity range, X = T for the temperature profile range, X = R for the radius range, and X = F for the integrated flux range.

17.11.12 MOMENTS

```
SHOW MOMENTS
```

Display the 4 "moments" images derived from a datacube by command MOMENTS. The contour spacing for each image is controlled by its own variable:

AREA_SPACING	for the integrated area M_AREA
PEAK_SPACING	for the peak brightness value M_PEAK (in K)
VELO_SPACING	for the Velocity field M_VELO (in km/s)
WIDTH_SPACING	for the Width field M_WIDTH (in km/s)

As for other image displays, CENTER and SIZE control the center and size of the image (in arcsec).

17.11.13 NOISE

```
SHOW NOISE [CLEAN|DIRTY]
```

Compute and display the noise statistic of the specified data cube, using a Gaussian fit to the histogram of pixel intensities for each channel.

17.11.14 PRIMARY

```
SHOW PRIMARY [PLANE|FIELD Kn] [First Last]
```

Show some sub-cube of the (4-D) Primary Beams.

The 2nd argument indicates whether SHOW shows all Fields for a given frequency plane (PLANE keyword), or all Frequency-dependent beams for a given Field (FIELD keyword). The 2nd and 3rd arguments are optional if the beam is only 3-D, which occurs for a single field or a frequency-indepedent beam in Mosaics (narrow band approximation).

17.11.15 PV

SHOW PV [[CLEAN|DIRTY|SKY] Axis Offset]

Extract from the indicated data cube a Position-Velocity image at the specified Offset (in arcsec) along the specified Axis (X or Y), and display it.

The data cube name is optional: the last specified one is used by default.

will simply redisplay (with different controls, see SHOW PV ? to have a list) the last extracted Position-Velocity image.

17.11.16 SED

SHOW SED [Frequency [FileName]]

Display the SED, computed by the "SED" widget from images (using the FLUX command) or from UV Tables (by fitting a simple model using UV_FIT), and fit a simple power law model through it.

Frequency is the reference frequency in GHz at which the flux is computed.

FileName is the data file produced by the "SED" widget.

If Frequency or Filename are not specified, their last values are reused.

17.11.17 SELFCAL

See HELP SELFCAL SHOW, and/or use SHOW SELFCAL ?

17.11.18 SNR

```
SHOW SNR
```

Display the Signal to Noise ratio map of a Mosaic.

17.11.19 SOURCES

```
SHOW SOURCES [First [Last]]
```

Display the "point sources" found by CLEAN at their positions, using a marker with area proportional to their flux. Positive sources are in black, negative ones are in red.

The displayed zone is controlled by variables SIZE and CENTER, as for in SHOW CLEAN or SHOW DIRTY. However, when SIZE is 0, the displayed zone is set to the minimum required to display all point sources detected by CLEAN. It may thus differ from the CLEAN map size.

17.11.20 SPECTRA

```
SHOW SPECTRA Clean|Dirty|Sky|ImageVar
```

Display individual spectra in a Map-like display (one per box, placed at their respective positions) from a 3-D SIC variable, assumed to be in the LMV order.

Note: the second argument may be omitted to re-display the same dataset with different control parameters. SHOW SPECTRA is however not fully protected against changes in the data set between two invocations, so this should only be used for consecutive commands.

17.11.21 UV_DATA

```
SHOW UV_DATA [Ant [Date]]
SHOW UV [Ant [Date]]
```

Displays the UV data. Items to be displayed are controlled by XTYPE and YTYPE variables. Ant and Date are optional arguments indicating which Antenna is to be highlighted, and for which date. Date is a sequential number from 1 to the number of observing dates.

Structure uvshow% controls some additional options, such as coloring of various dates, of data flagging, etc...

17.11.22 UV_FIT

```
SHOW UV_FIT [? | RESET]
```

Display the UV_FIT results. The display is controlled by the UVF% structure.

SHOW UV_FIT ? will list the current values of these control parameters.

SHOW UV_FIT RESET will reset all control variables (UVF% structure) to their default values.

The number and order of the fitted functions to be plotted are controlled by UVFIT%NF and UVFIT%ORDER

The number of parameters to be plotted along X and Y axes are controlled by UVFIT%NX, UVFIT%NY. The kinds and ranges of those parameters are coded in the UVFIT%XTYPE and UVFIT%YTYPE strings (e.g. kind min max). Possible kinds are: CHANNEL VELO FREQ RMS RA DEC FLUX WIDTH MAJOR MINOR ANGLE and RATIO. A * instead of min and/or max implies the use of the default minimum and maximum values of the plotted parameter.

17.11.23 Variables:

```
SHOW Image control variables
```

The following variables control the SHOW display.

Display the header	DO_HEADER	[YES]	
Display a color wedge	DO_WEDGE	[YES]	
Plot bitmaps	DO_BIT	[YES]	
Add the clean beam	DO_NICE	[NO]	
Add the UV coverage	DO_COVERAGE	[NO]	
Add the dirty beam	DO_DIRTY	[NO]	
Display the labels	DO_LABEL	[YES]	
Display contour levels	DO_CONTOUR	[YES]	
grey-scale contours	DO_GREY	[NO]	
Better marks	DO_MASK	[NO]	(Hide image behind cha
Channel range	RANGE	[0 0]	in 3rd axis unit, (0,0)
Channel label	MARK	[velocity]	[VELOCITY, FREQUENCY or
Displayed field size	SIZE	[0 0]	in ANGLE_UNIT
Field center	CENTER	[0 0]	offset in ANGLE_UNIT
Cross size at center	CROSS	[2]	in arcsec, 0 ==> none
Fields layout	DO_FIELDS	[NO]	(for Mosaics)
Colour Scale range	SCALE	[0 0]	(0,0) ==> all dynamic

```
Contour levels          SPACING     [ 0 Units]      (0) =+> Guess
Spacing type            SPACE_TYPE  [ AUTO]  [LIN, LOG or USER]
Side Display            SHOW_SIDE  [ NO ]
```

17.11.24 BOX_LIMITS

```
CHARACTER*60 BOX_LIMITS
```

If not void, this character string must indicate the 4 limits of each panel display, for example

```
LET BOX_LIMITS "* * -5 10"
```

If void, variables SIZE and CENTER are used instead.

17.11.25 DO_BIT

```
LOGICAL DO_BIT
```

Show a bitmap color plot (YES) or do not (NO) for each panel. The bitmap range is controlled by variable SCALE.

17.11.26 DO_CONTOUR

```
LOGICAL DO_CONTOUR
```

Draw (YES) or do not (NO) draw the contour level, which are controlled by SPACING and SPACE_TYPE. This is provided for compatibility with Mapping and Greg.

17.11.27 DO_COVERAGE

```
LOGICAL DO_COVERAGE
```

For SHOW DIRTY or SHOW CLEAN, DO_COVERAGE indicates if in addition to the channel maps, the UV coverage should be displayed in another panel.

17.11.28 DO_DIRTY

```
LOGICAL DO_DIRTY
```

For SHOW DIRTY or SHOW CLEAN, DO_DIRTY indicates if in addition to the channel maps, the dirty beam should be displayed in another panel.

17.11.29 DO_FIELDS

LOGICAL DO_FIELDS

Plot the 50 % contour levels of the individual pointings in a Mosaic on top of the maps.

17.11.30 DO_GREY

LOGICAL DO_GREY

Do (YES) or do not (NO) use grey-scale contouring. Note that this is independent of DO_CONTOUR.

17.11.31 DO_HEADER

LOGICAL DO_HEADER

DO_HEADER indicates whether a panel displaying a summary information about the displayed data is drawn or not.

17.11.32 DO_LABEL

LOGICAL DO_LABEL

DO_LABEL indicates if the axes should be labelled.

17.11.33 DO_MASK

LOGICAL DO_MASK

Hide part of the plot (contour or bitmaps, and even axis ticks) that lie in the upper left corner of each panel to have a clean display of the value specified by MARK.

17.11.34 DO_NICE

LOGICAL DO_NICE

DO_NICE indicates if the angular resolution of the data ("beam") should be displayed in the lower left corner of each panel.

17.11.35 DO_WEDGE

LOGICAL DO_WEDGE

DO_WEDGE indicates whether a color wedge of the bitmaps is drawn or not. It appears in the Header panel, so DO_WEDGE is active only if DO_HEADER and DO_BIT are set.

17.11.36 CENTER

REAL CENTER[2]

Specify the relative coordinates of the center of the displayed region, in ANGLE_UNIT (as specified by SET ANGLE_UNIT).

17.11.37 CROSS

REAL CROSS

Indicate the size (in arcsec) of a cross to be drawn at the map projection center, as a reference marker for relative positions.

17.11.38 MARK

CHARACTER*12 MARK

Specify the type of labelling for each panel. It can be VELOCITY, FREQUENCY or CHANNEL. Case is not significant.

17.11.39 RANGE

REAL RANGE[2]

Specify the range (in 3rd axis unit) of displayed channels. This provides a facility similar to FIRST and LAST variables, but in a more user-oriented unit.

RANGE should not be confused with SPAN, which specifies the displayed velocity range when one of the 2 first axis is of type Velocity, e.g. for SHOW PV

17.11.40 SCALE

REAL SCALE[2]

SCALE indicates the Colour Scale Range, range of image units which are displayed in the color bitmap. SCALE = 0 implies an automatic evaluation of this range, based on the Min and Max of the data cube.

17.11.41 SIZE

REAL SIZE[2]

Specify the displayed are size, in ANGLE_UNIT.

17.11.42 SPACE_TYPE

CHARACTER SPACE_TYPE

Define the type of spacing for the contour levels
LIN Linear spacings. The Zero contour is always omitted.
LOG Exponential spacing
NONE No contour levels at all (same as DO_CONTOUR = NO)
USER Keep the contour levels specified by command LEVELS
For LOG case, see HELP LEVELS EXPO in the GreG internal help.

As a backward compatibility, a negative spacing is a convention to use the current contour levels, although using SPACE_TYPE = USER should be preferred.

17.11.43 SPACING

CHARACTER*16 SPACING[2]

SPACING indicates the step (and implicitely the contour levels) for contouring and/or greyscale. SPACING[1] control the values, and SPACING[2] the Unit of these values. It can be UNITS (default data cube units), NOISE or SIGMA (noise level), Jy (which also stands for Jy/Beam) and its sub-units mJy (or milliJy) and microJy, or K and its sub-units mK (or milliK) or microK.

Jy (and its sub-units) is taken as equivalent to Jy/Beam, and automatic conversion between brightness unit (K) and flux unit (Jy/Beam) is made when the data cube has a specified angular resolution and frequency.

SPACING[1] = 0 means an automatic guess of the contours from the Min and Max of the data cube.

The type of spacing is controlled by variable SPACE_TYPE.

17.11.44 SPAN

```
REAL SPAN[2]
```

SPAN indicates the limits of plot along the velocity axis, when one of the plotted dimensions is Velocity. In particular, SPAN is used for SHOW PV. The spatial dimension (if any is displayed) is controlled by SIZE and CENTER.

Note that SPAN differs from RANGE: RANGE controls the range of planes displayed for a 3-D cube.

17.11.45 SHOW_SIDE

```
LOGICAL SHOW_SIDE
```

Controls whether the Side Display is active by default or not.
Option /SIDE can be used to overcome this default value.

17.12 SLICE

```
[DISPLAY\]SLICE VarName Start1 Start2 End1 End2 UNIT
```

Cut a slice through the 3-D image variable VarName with the specified edges and put it in the SLICED image variable. UNIT is a keyword indicating in which units the edges are specified. It can be PIXELS, DEGREE, MINUTE, SECOND, RADIAN, of ABSOLUTE. For ABSOLUTE, Start1 and End1 are assumed to be in hours and Start2 End2 in degrees, with sexagesimal notation allowed.

```
[DISPLAY\]SLICE VarName StartX StartY Unit ANGLE AngleValue
```

This alternate syntax allows to specify a point and an orientarion of the slice. The coordinates StartX StartY are in the angular unit specified by Unit (which can be ABSOLUTE) AngleValue is in Degree. The Slice length will be automatically computed to intersect the whole cube. This syntax can be more convenient to compare parallel slices, or slice intersecting at a common point that can be specified by StartX StartY.

17.13 SPECTRUM

```
[DISPLAY\]SPECTRUM OutSpec InCube [Mask]
  [/CORNER Imin Imax Jmin Jmax] [/PLANE First Last]  /MEAN  or  /SUM
```

Compute a Spectrum, stored in variable OutSpec from an input data cube stored in variable InCube, by summing (if /SUM is present) or averaging (if /MEAN is present) over the region specified by the current GreG polygon, or the 2-D or 3-D mask specified by Mask, truncated to the specified corner if any.

OutSpec[1] contains the "velocities" (values along the 3rd axis of the InCube data cube). OutSpec[2] contains the spectrum values.

17.13.1 SPECTRUM /CORNER

[DISPLAY\]SPECTRUM OutSpec InCube [Mask] /CORNER Imin Imax Jmin Jmax [/PLANE First Last] /MEAN or /SUM

Restricts the computation to the specified 2-D region of the data cube. One of the mutually exclusive options /MEAN or /SUM must be present.

17.13.2 SPECTRUM /MEAN

[DISPLAY\]SPECTRUM OutSpec InCube [Mask] /MEAN [/CORNER Imin Imax Jmin Jmax] [/PLANE First Last]

Indicates that we are computing a Mean spectrum.
The spectrum is the average spectrum accounting for all valid pixels defined by the selection.

/MEAN is exclusive of /SUM.

17.13.3 SPECTRUM /PLANE

[DISPLAY\]SPECTRUM OutSpec InCube [Mask] /PLANE First Last [/CORNER Imin Imax Jmin Jmax] /MEAN or /SUM

Restricts the spectrum to the specified channels of the data cube. One of the mutually exclusive options /MEAN or /SUM must be present.

17.13.4 SPECTRUM /SUM

[DISPLAY\]SPECTRUM OutSpec InCube [Mask] /SUM [/CORNER Imin Imax Jmin Jmax] [/PLANE First Last]

Indicates that we are computing an integrated spectrum. The result corresponds to the integral of the signal over the valid area (sum of

values times the pixel size in user units).

/SUM is typically used to compute integrated flux from Jy/Beam units. /SUM is exclusive of /MEAN.

17.14 STATISTIC

[DISPLAY\]STATISTIC [DataCube [FirstPlane [LastPlane]] [/EDGE Margin] [/NOISE VarNoise]

Compute some basic statistics (min, max, mean, rms,...) on the specified DataCube buffer (default: CLEAN) and planes (default: variables FIRST and LAST). The current polygon or mask (see commands SUPPORT and MASK) is used to define the area on which the pixels are to be taken into account, except when /EDGE 0 is mentionned.

In addition to Min and Max, the command prints 3 noise numbers:
- the rms inside the current support (whose value depends on the source strength and structure)
- the rms outside of the support (a good estimate of the effective noise), returned in variable STAT_NOISE.
- the theoretical thermal noise (that would be the expected noise unless the data is dynamic range limited, or unless the weights were not set properly)

Caution: Statistics on primary-beam corrected images (SKY) cannot be properly computed in this way, because the noise is non uniform.

17.14.1 STATISTIC /NOISE

[DISPLAY\]STATISTIC [DataCube [FirstPlane [LastPlane]] [/EDGE Margin] /NOISE VarNoise

Return the rms noise in the specified VarNoise SIC real scalar variable. The rms noise is also available in variable STAT_NOISE, and if the image is the CLEAN image, in CLEAN%NOISE as well for further convenience and use in scripts.

17.14.2 STATISTIC /EDGE

[DISPLAY\]STATISTIC [DataCube [FirstPlane [LastPlane]] /EDGE Margin [/NOISE VarNoise]

Define the size of the "edges" that are avoided in the statistic estimate. Margin is a number in range [0,0.5[. Default is 1/16.

Margin 0 means considering the whole image, ignoring mask and supports.

17.15 VIEW

[DISPLAY\]VIEW Argument [FirstPlane [LastPlane]] [/NOPAUSE] [/OVERLAY]

where (in the most general case, see details for specific Keywords)
is the the name of an internal buffer to be plotted (BEAM, CCT, CLEAN, DIRTY, FIELDS, MASK, etc..), or any Sic Image variable, or a file name.

[DISPLAY\]VIEW ?

will list the names of recognized keywords. If Variable is not one of the recognized keywords, but an existing Image variable, this image will be displayed.

FirstPlane and LastPlane
are optional arguments to restrict the range of channels to be plotted (default: planes between variables FIRST and LAST). If only FirstPlane is specified, SHOW only that plane.

VIEW is also controlled by a set of variables, available in the View% global structure. VIEW shows 4 (or 6) panels, representing the Integrated area, the current channel, and the Integrated spectrum and the current spectrul (in possibly 2 different scales). An interactive control of the View is possible using the cursor, unless the /NOPAUSE option is given.

The spectral panels can display line identifications based on the current CATALOG. If a variable REDSHIFT exists, the frequency scale is corrected for the source Redshift, so that line identification remains possible.

17.15.1 VIEW /NOPAUSE

[DISPLAY\]VIEW Argument [FirstPlane [LastPlane]] /NOPAUSE

Just display the panels as currently defined, without looping with the Graphic cursor for interactive view.

This will be the default mode if variable DO_LOOP is set to NO.

17.15.2 VIEW /OVERLAY

```
    [DISPLAY\]VIEW Argument [FirstPlane [LastPlane]] [/NOPAUSE]
  /OVERLAY [Variable [Channel]]
```

Overlay (in contours) the specified Channel of the specified SIC 2 or 3-D Variable. The default Variable is CONTINUUM, the continuum image. If Variable is a 3-D SIC Variable, Channel indicates which plane is overlaid (default: last one).

17.15.3 VIEW Keys

```
Position-independent actions:
  Press E key: EXIT loop
  Press H key: HELP display
  Press P key: PRINT plot in "ha" subdirectory
  Press Q key: QUIT loop
  Press X key: EXTRACT on disk current zoomed region
  Press N key: Toggle  Narrow - Wide mode

Cursor on images:
  Left  clic: Display spectrum at pointed position
  Right clic: Define a polygon
  Press B key: BACK to full field of view
  Press C key: COORDINATES toggled from absolute to relative and back
  Press M key: Display MASK if any
  Press S key: SLICE definition (velocity-position)
  Press U key: UNKILL pointed pixel
  Press Z key: ZOOM defined spatial region
  Press V key: Display map coordinates at current position
  Press W key: WRITE Integrated Area image
Cursor on spectra:
  Left clic in Selected   Spectrum: Display selected velocity channel
  Left clic in Integrated Spectrum: Define velocity range
  Press B key: BACK to full velocity range
  Press C key: COORDINATES toggled from freq/velo to channels and back
  Press L key: LABEL spectral Lines
  Press M key: MOVIE
  Press Z key: ZOOM defined velocity region
  Press W key: WRITE Integrated (and current) Spectrum
Cursor outside plots:
  Press B key: BACK to full velocity range AND full field of view
```

Returned values are found in view%current structure. The ZOOM action produces a sub-cube named EXTRACTED. The SLICE action produces a 2-D data set named SLICED. As any other SIC Image variable, EXTRACTED and

SLICED can be written on disk using command WRITE.

17.15.4 VIEW Scripts

The VIEW plot is done by procedure p_view_cct for Clean Components CCT and p_view_map for data cubes.

17.15.5 VIEW Variables

VIEW uses the SHOW control variables SIZE, CENTER, RANGE and CROSS. In addition, it has its own control variables

```
view%expand       0.8 Character and Tick expansion factor
view%contour      NO  Contour the current channel map
view%movie        0   Elapsed time in seconds for a movie (0 means guess)
view%side         YES Display values in side Window
view%status%rima  NO  Relative coordinates in Images
view%status%rspe  NO  Relative coordinates in Spectra
```

17.15.6 VIEW Keywords:

```
    VIEW recognizes the following keywords
  BEAM        Synthesized beam(s)
  CCT         Clean Cumulative Flux
  CLEAN, DIRTY, SKY   Usual sky maps
  NOISE       per channel noise
  PRIMARY     Primary beams
or any other 3-D SIC variable.
```

VIEW falls back to SHOW for un-recognized keywords, such as MOMENTS COMPOSITE, UV, etc...

17.15.7 BEAM

```
VIEW BEAM [PLANE|FIELD Kn]
```

View some sub-cube of the (up to 4-D) dirty Beam.

The 2nd argument indicates whether VIEW shows dirty beams of all Fields for a given frequency plane (PLANE keyword), or all Frequency-dependent beams for a given Field.

The 2nd and 3rd arguments are optional if the beam is only 2-D or 3-D,

which occurs for a single field or a frequency-independent beam in Mosaics (narrow band approximation).

17.15.8 PRIMARY

```
VIEW PRIMARY [PLANE|FIELD Kn]
```

View some sub-cube of the (up to 4-D) primary Beams.

The 2nd argument indicates whether VIEW shows all Fields for a given frequency plane (PLANE keyword), or all Frequency-dependent beams for a given Field.

The 2nd and 3rd arguments are optional if the beam is only 2-D or 3-D, which occurs for a single field or a frequency-independent beam in Mosaics (narrow band approximation).

18 CLEAN Language Internal Help

18.1 Language

```
CLEAN\ Language Summary
```

The CLEAN\ Language contains commands related to UV data handling, UV to DataCubes conversion and DataCube processing.

```
ALMA          : Joint deconvolution of ALMA and ACA dirty images
BUFFERS       : List and Display the known buffer status
CCT_CLEAN     : Compute CLEAN image from Clean Component Table, without resi
CCT_CONVERT   : Convert an image into a Clean Component Table
CCT_MERGE     : Merge two Clean Component Tables
CLEAN         : Deconvolve a dirty image using the current METHOD
DISCARD       : Discard one of the active data buffers
DUMP          : Dump the control parameters of the deconvolution algorithms
FIT           : Fit the dirty beam
MAP_COMBINE   : Combine two data Cubes
MAP_COMPRESS  : Average a Cube by several channels
MAP_INTEGRATE : Compute a Moment 0 Map
MAP_REPROJECT : Spatial resampling of a Cube
MAP_RESAMPLE  : Resample a Cube on a different Velocity/Frequency scale
MAP_SMOOTH    : Smooth by N channels
```

```
MOSAIC        : Toggle the mosaic mode
MX            : Iteratively image and deconvolve a dirty image
PRIMARY       : Apply primary beam correction
READ          : Read the input files in internal buffers
SPECIFY       : Change the Frequency / Velocity scale or the Telescope
SUPPORT       : Define the support used to search clean components
UV_BASELINE   : Subtract a continuum baseline from a Line UV data
UV_CHECK      : Check UV data for null visibilities or per channel flags.
UV_COMPRESS   : Compress a Line UV data into another Line UV data
UV_CONTINUUM  : Compress a Line UV data into a Continuum UV data
UV_EXTRACT    : Extract a range from
UV_FILTER     : Filter out (line) channels
UV_FLAG       : Interactively flag UV data
UV_MAP        : Build the dirty image and beam from a UV table
UV_RESAMPLE   : Resample (in Velocity) the UV data
UV_RESIDUAL   : Subtract Clean Component from the UV Data
UV_RESTORE    : Restore a Clean image from UV data and Clean Components
UV_REWEIGHT   : Evaluate UV weights from visibility statistics
UV_SHIFT      : Shift a UV table to common phase center
UV_SMOOTH     : Smooth spectrally a Line UV data
UV_SPLIT      : Split a UV table into Line and Continuum
UV_STAT       : Gives beam sizes and noise properties as a function of
                  tapering or robust weighting parameter
UV_TIME       : Time average the current UV data
UV_TRIM       : Suppress useless information from UV data
UV_TRUNCATE   : Truncate the baseline range of the UV data
WRITE         : Save internal buffers or Image variables onto output files
```

18.2 ALMA

[CLEAN\]ALMA [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY] [/NOISE] [/METHOD]

Joint deconvolution methods specific to ALMA+ACA observations.

The ALMA simulator can be accessed either by typing "@ alma" at the prompt or from the MAPPING main menu.

18.3 BUFFERS

[CLEAN\]BUFFERS

List the status (sizes) of known buffers. Active buffers can be nullified by using the DISCARD command.

18.3.1 BUFFERS AGAINS

AGAINS is a buffer containing the results of a self-calibration with command SELFCAL. It is a special table in the following format. Each line contains a different time stamp. Column 1 is the Time (in seconds), Col 2 the Date (in Days, with an arbitrary origin), Col 3 the number of antennas, Col 4 the antenna reference number, and then, for each antennas i, Cols (2+3*i:2+3*i) contains the Amplitude correction factor, the Phase correction (in degree), and the Signal-to-noise of the correction.

18.3.2 BUFFERS BEAM

BEAM is a 2,3 or 4-D image buffer containing the synthesized beam computed by UV_MAP (or by READ BEAM).

18.3.3 BUFFERS CCT

CCT is a special 3-D array containing the Clean Component Table. The first dimension contains (x,y,f), the flux (in Jy) at offset x,y (in radians). The second dimension handles the number of channels. The last dimension handles the Clean iteration number.

18.3.4 BUFFERS CGAINS

CGAINS is a pseudo-UV table containing the complex gain correction resulting from a SELFCAL command (or read by READ CGAINS). It is analogous to a single-channel UV table, but interpreted by IMAGER as gain corrections to apply to the UV data by command APPLY.

18.3.5 BUFFERS CLEAN

CLEAN is the deconvolved 2 or 3-D image produced by UV_MAP (or read by READ CLEAN). It is not corrected for primary beam attenuation.

For Mosaics, command CLEAN produces a SKY brightness map, corrected from primary beam attenuations.

18.3.6 BUFFERS CLIPPED

CLIPPED is a 2-D table containing the Clipped spectra produced by command UV_PREVIEW. Col 1 contains the Velocity, Cols 2:N+1 the spectra for the N tapers used in UV_PREVIEW, Col N+2 the natural weights.

18.3.7 BUFFERS CONTINUUM

CONTINUUM contains the pseudo-UV table used (and created) by command UV_MAP /CONTINUUM to construct the continuum image. See command UV_CONTINUUM for details.

18.3.8 BUFFERS DIRTY

DIRTY contains the dirty image produced by UV_MAP. Its interpretation is different for a Single field than for a Mosaic.

18.3.9 BUFFERS EXTRACTED

EXTRACTED contains the 3-D data cube produced by command EXTRACT

18.3.10 BUFFERS FIELDS

FIELDS is an intermediate buffer created by SHOW PRIMARY. It is a transposed version of the PRIMARY buffer, in order (l,m,f) where f is the field number.

Note: there is a confusion with the FIELDS structure created by READ UV on Mosaic UV tables.

18.3.11 BUFFERS MASK

MASK contains the Mask computed by command MASK (or read by READ MASK). It may be used (or not) for CLEAN, depending on the selected type of Support (see SUPPORT /MASK and MASK USE commands).

18.3.12 BUFFERS M_AREA

M_AREA is a 2-D image containing the integrated area computed by command MOMENTS (CASA 0-th moment)

18.3.13 BUFFERS M_PEAK

M_PEAK is a 2-D image containing the peak flux computed by command MOMENTS (CASA 8-th moment)

18.3.14 BUFFERS M_VELO

M_VELO is a 2-D image containing the mean velocity computed by command MOMENTS. Its value depends on the method used in command MOMENTS: it can be the 1-st moment, or a fit of the peak velocity.

18.3.15 BUFFERS M_WIDTH

M_WIDTH is a 2-D image containing the velocity dispersion computed by command MOMENTS. It is equivalent to the Full Width at Half Maximum is the line shape is Gaussian.

18.3.16 BUFFERS PRIMARY

PRIMARY is the primary beam, computed by command PRIMARY or read by command READ PRIMARY. It may be 2-D (for single fields) or 3-D (for Mosaics). It the later case, it is in the order (f,l,m) where f is the field number.

18.3.17 BUFFERS RESIDUAL

RESIDUAL is the image of the residuals from the Clean deconvolution.

18.3.18 BUFFERS SHORT

SHORT is the Short-Spacing image, computed by commands UV_SHORT or XY_SHORT, or read by command READ SINGLE.

18.3.19 BUFFERS SINGLE

SINGLE is the Single-Dish data read by command READ SINGLE. It may be a Class-table or a 3-D data cube.

18.3.20 BUFFERS SKY

SKY is a 3-D image buffer containing the current estimate of the Sky brightness distribution. It is provided by command PRIMARY from the CLEAN and PRIMARY buffers for single-field data, but by command CLEAN for Mosaics. Noise is not uniform in SKY image.

18.3.21 BUFFERS SLICED

SLICED is 2-D buffer produced by the SLICE command.

18.3.22 BUFFERS SPECTRUM

SPECTRUM is a 2-D table containing the spectra produced by command UV_PREVIEW. Col 1 contains the Velocity, Cols 2:N+1 the spectra for the N tapers used in UV_PREVIEW, Col N+2

18.3.23 BUFFERS UV

UV is the current UV data. It is initially set by command READ UV, but modified by many UV_like commands (resampling, calibration, etc...)

18.3.24 BUFFERS UV_FIT

UV_FIT handles the results of the UV_FIT command. It is a special format 2-D table. See HELP UV_FIT for details.

18.3.25 BUFFERS UVCONT

UVCONT handles the intermediate (pseudo-)UV table created and used by command UV_MAP /CONT for Multi-Frequency-Synthesis. It is a valid 2-D UV table, but which have more than 1 visibility for the same time and baseline pair, with different (frequency-scaled) (u,v) coordinates. Such UV tables can be properly imaged, but not properly self-calibrated.

18.3.26 BUFFERS UVRADIAL

UVRADIAL handles an azimutally averaged version of the UV data. It is produced by command UV_RADIAL.

18.3.27 BUFFERS WEIGHT

WEIGHT is the equivalent of PRIMARY for Mosaics. It is a 2 or 3-D image buffer containing the relative response given by all pointings of the Mosaic coverage.

18.4 CCT_CLEAN

[CLEAN\]CCT_CLEAN [Niter]

Convert the CCT Clean Component Table into the CLEAN image. The resid-

uals are not added at this stage, contrary to what is done by the CLEAN command.

Niter is the last iteration to be retained (to be implemented). If not present, all components are considered.

18.5 CCT_CONVERT

[CLEAN\]CCT_CONVERT [Threshold]

Convert the CLEAN image into the CCT Clean Component Table.

Threshold is the minimum (absolute value of) flux per pixel retained. Default is 0

18.6 CCT_MERGE

[CLEAN\]CCT_MERGE Out In1 In2

Combine two input Clean Component Tables.

Out can be a file name or an existing Image variable. The distinction is made by the existence of a "." in the name
If it is a file, it is created like the In1 Variable
If it is an Image variable, it must match the number of channels of the In1 Variable, and
be large enough to handle the total number of components.

In1 and In2 can be file names or existing Image variables. They must match in terms of number of channels.

18.7 CLEAN

[CLEAN\]CLEAN [FirstPlane [LastPlane]] [/RESTART [File]][/PLOT Clean|Residu]
[/FLUX Fmin Fmax] [/QUERY] [/NITER NiterList] [/ARES AresList]

Deconvolve a Mosaic or Single-field using the current METHOD (in SIC variable METHOD). See INPUT CLEAN for the other SIC variables controlling the deconvolution process. Supported methods are CLARK, HOGBOM, MRC, MULTISCALE and SDI. Use HELP CLEAN Method and/or HELP CLEAN METHODNAME for further details on each algorithm.

Clean the specified plane interval (default: planes between variables FIRST and LAST). If only FirstPlane is specified, Clean only that plane.

This command allows a per-plane definition of the convergence criteria CLEAN_NITER and CLEAN_ARES.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is possible, a sensible value of each parameter will automatically be computed from the context if the value of the corresponding variable is set to its default value, i.e. zero value and empty string. A few variables are initialized to "reasonable" values.

[CLEAN\]CLEAN ?
Will list all main CLEAN_* variables controlling the CLEAN parameters for the current METHOD.

HELP CLEAN Variables will give a more complete list.

18.7.1 CLEAN /FLUX

[CLEAN\]CLEAN [FirstPlane[LastPlane]] /FLUX Fmin Fmax [/PLOT Clean|Residu] [/QUERY] [/NITER NiterList] [/ARES AresList]

Display the cumulative Clean flux as Clean progresses. This option is inactive in Parallel mode.

18.7.2 CLEAN /PLOT

[CLEAN\]CLEAN [FirstPlane [LastPlane]] /PLOT Clean|Residu [/FLUX Fmin Fmax] [/QUERY] [/NITER NiterList] [/ARES AresList]

Display the iterated Clean or Residual image for Cleaning methods which have major cycles (CLARK or SDI). This option is inactive in Parallel mode.

18.7.3 CLEAN /QUERY

[CLEAN\]CLEAN [FirstPlane [LastPlane]] /PLOT Clean|Residu /QUERY [/FLUX Fmin Fmax] [/NITER NiterList] [/ARES AresList]

*** Obsolescent ***

Prompt for continuation when a Major cycle is complete. This option is inactive in Parallel mode.

As a replacement of this facility, we suggest you to fragment your CLEAN in several steps, using the /RESTART option and the VIEW CLEAN (or RESIDUAl) command after each step.

18.7.4 CLEAN /NITER

[CLEAN\]CLEAN [FirstPlane [LastPlane]] /NITER NiterList [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY] [/ARES AresList]

Use a per-plane value for the number of iterations, instead of the global NITER variable. NiterList should be a 1-D integer array of dimension the number of channels.

This option is only available through the CLEAN command, not through the specific command of each method.

18.7.5 CLEAN /ARES

[CLEAN\]CLEAN [FirstPlane [LastPlane]] /ARES AresList [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY] [/NITER NiterList]

Use a per-plane value for the absolute residual used to stop cleaning, instead of the global ARES variable. AresList should be a 1-D real array of dimension the number of channels.

18.7.6 CLEAN /RESTART

[CLEAN\]CLEAN [FirstPlane [LastPlane]] /RESTART [File] [/PLOT Clean|Residu]
[/FLUX Fmin Fmax] [/QUERY] [/NITER NiterList] [/ARES AresList]

Instruct CLEAN to start from the specified Clean Component Table stored in the specified File, or those currently in memory if no File is given.

This can be used to re-start a Clean that was not deep enough. It can be used to split the Clean operation in several steps, possibly changing the support at each step, or even the Clean METHOD.

For spectral data, the starting model should have the same number of channels, or have only 1. The later capability can be used to start from the Clean Components derived from a Continuum-only data set, so that deconvolution proceeds to find out the line-only structure. This is an alternative to separate deconvolution of Continuum and continuum-subtracted data followed by addition of the Clean images.

18.7.7 CLEAN Methods:

CLARK	A minor/major cycle cleaning using FFTs for speed Fast, but less stable than others.
HOGBOM	The basic deconvolution method, working component per component. Slow but robust
MRC	A dual-scale multi-resolution Clean working on a smooth and a difference map. Fast, but with no notions of Clean components
MULTISCALE	A 3-scales multi-resolution-Clean working on 3 smoothed versions of the map. Slow, but very stable and adapted to sources with extended structures.
SDI	A major-cycle only method. Good for extended structures but not very performant overall.

18.7.8 CLARK

CLEAN [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY]

with METHOD = CLARK

A Major-Minor cycles CLEAN method, originally developed by B.Clark, in which clean components are selected using a limited beam patch, and deconvolved through Fourier transform at each major cycle. In mosaic mode (See command MOSAIC), a mosaic clean is performed. Rings and/or stripes may appear on extended sources. Faster than the HOGBOM method for single fields but maybe slower for mosaics. The strategy to search for CLEAN components in CLARK method does not work properly when the secondary side lobes are too large (e.g. larger than 0.3), or in case of high phase noise.

18.7.9 HOGBOM

CLEAN [FirstPlane [LastPlane]] [/FLUX Fmin Fmax] [/RESTART [File]]

with METHOD = CLEAN

The simplest CLEAN algorithm, originally developed by Hogbom. In mosaic mode (See command MOSAIC), a mosaic clean is performed. Rings and or stripes may appear on extended sources. It is slower than CLARK for a single field but maybe faster for a mosaic. It is extremely robust. Cleaning can be interrupted by pressing C at any time.

18.7.10 MRC

CLEAN [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin

Fmax] [/QUERY]

with Method = MRC

Perform a Multi-Resolution CLEAN on the current dirty image. MRC does not support mosaics for theoretical reasons.

If option /PLOT is given, a display of the CLEAN or RESIDUAL map will be shown at each major cycle, depending on the argument (default: Residual). The user will be prompted for continuation when the /QUERY option is present. The cumulative, already cleaned flux is displayed in real time in an additional window while cleaning goes on when the /FLUX option is present. Parameters of the /FLUX option are then used to give the flux limits for this display. A summary plot with the Difference, Smooth, and total CLEANed maps is also displayed.

18.7.11 MULTI

CLEAN [FirstPlane [LastPlane]] [/FLUX Fmin Fmax] [/RESTART [File]]

with METHOD = MULTI

Perform a 3-scales Hogbom-like deconvolution. A each minor cycle, the the scale with the highest signal to noise is used to define the Clean Components to retain. The algorithm is thus very stable.

The cumulative, already cleaned flux is displayed in real time in an additional window while cleaning goes on when the /FLUX option is present. Parameters of the /FLUX option are then used to give the flux limits for this display.

MULTI also works on mosaics. The smoothing ratio can be controlled by variable CLEAN_SMOOTH.

18.7.12 SDI

CLEAN [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY] [/RESTART [File]]

with METHOD = SDI

Perform a Steer-Dewdney-Ito CLEAN. This clean method selects an ensemble of clean components and remove them at once using FFTs. It works best for extended sources and UV coverages with short spacings. In such a case, it may avoid the "ringing" features which appear using the CLARK

or HOGBOM techniques. In mosaic mode (see command MOSAIC), a mosaic clean is performed.

SDI is very sensitive to the selected support (see MASK and SUPPORT commands), especially when the dirty beam sidelobes are significant.

18.7.13 Variables:

```
    Basic CLEAN parameters
CLEAN_GAIN       [       ] Loop gain
CLEAN_STOP       [       ] Stopping criterium (string)
CLEAN_NITER      [       ] Maximum number of clean components
CLEAN_FRES       [      %] Maximum value of residual (Fraction of peak)
CLEAN_ARES       [Jy/Beam] Maximum value of residual (Absolute)
CLEAN_POSITIVE   [       ] Minimum number of positive components at start
CLEAN_NKEEP      [       ] Min number of components before convergence

Clean beam user specification
BEAM_SIZE       [ 0 0 0 ] Major, Minor (in arcsec) and PA (in Degree)

Method dependent CLEAN parameters
CLEAN_INFLATE    [      ] Maximum Inflation factor for UV_RESTORE (MULTISCAL
CLEAN_NCYCLE     [      ] Max number of Major Cycles (SDI & CLARK methods)
CLEAN_NGOAL      [      ] Max number of comp. in Cycles (ALMA method)
CLEAN_RATIO      [      ] Smoothing factor (MRC default 0: guess, otherwise
CLEAN_SEARCH     [      ] Threshold to search Clean Comp. in a Mosaic (def 0
CLEAN_SIDELOBE   [      ] Min threshold to fit the synthesized beam
CLEAN_SMOOTH     [      ] Smoothing ratio: MRC (def 2 or 4)  and MULTISCALE
CLEAN_SPEEDY     [      ] Speed-up factor (CLARK)
CLEAN_TRUNCATE   [      ] Threshold for restoring a Mosaic (def 0.2)
CLEAN_WORRY      [      ] Worry factor (CLARK and MULTISCALE)

Old (seldom used) names like in MAPPING
BLC              [  pixel] Bottom left corner of cleaning box
TRC              [  pixel] Top right corner of cleaning box
BEAM_PATCH       [  pixel] Size of cleaning beam ** not clear **


Restoration control CLEAN parameters
CLEAN_RESIDUAL   [-1,0,1]  Control how residuals are added or not - For debu
```

18.7.14 BEAM_SIZE

BEAM_SIZE is a Real array of size 3 containing the user-specified values for the Clean beam size, with BEAM_SIZE[1] the Major axis, BEAM_SIZE[2]

the Minor axis (both in arcsec) and BEAM_SIZE[3] the Position angle (in degrees, North towards East).

If BEAM_SIZE[1] and BEAM_SIZE[2] are 0, the CLEAN or FIT commands will determine the Clean beam size (available in variable BEAM_FITTED similar to BEAM_SIZE) by adjusting the dirty beam main peak.

If only one of BEAM_SIZE[1] and BEAM_SIZE[2] is non zero, it will be used as the Clean beam size for a circular beam.

If both are non zero, BEAM_SIZE totally specifies the Clean beam size. BEAM_FITTED is thus set to the BEAM_SIZE values when used.

18.7.15 CLEAN_ARES

This is the minimal flux in the dirty map that the program will consider as significant. Alternatively, the threshold can be specified as a fraction of the peak flux using CLEAN_FRES. Once this level has been reached the program stops subtracting, and starts the restoration phase. The unit for this parameter is the map unit (typically Jy/Beam). The parameter should usually be of the order of magnitude of the expected noise in the clean map.

If 0, CLEAN_FRES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is ARES.

This may be overseded by the /ARES option which imposes a limit per plane through an array of values.

18.7.16 CLEAN_FRES

This is the minimal fraction of the peak flux in the dirty map that the program will consider as significant. Alternatively, an absolute threshold can be specified using CLEAN_ARES. Once this level has been reached the program stops subtracting, and starts the restoration phase. This parameter is normalized to 1 (neither in % nor in db). It should usually be of the order of magnitude of the inverse of the expected dynamic range of the intensity.

If 0, CLEAN_ARES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will

be used for CLEAN_ARES.

Short form is FRES.

18.7.17 CLEAN_GAIN

This is the gain of the subtraction loop. It should typically be chosen in the range 0.05 and 0.3. Higher values give faster convergence, while lower values give a better restitution of the extended structure. A sensible default is 0.2.

Short form is GAIN.

18.7.18 CLEAN_INFLATE

Maximum Inflation factor for UV_RESTORE (MULTISCALE method). If the number of true (i.e. pixel based) Clean components found by MULTISCALE is larger than CLEAN_INFLATE times the number of compressed (i.e. those with the smoothing factor information) components, expansion of the compressed components will not be possible, and UV_RESTORE or CLEAN /RESTART will not be useable.

The default is 121, large enough to handles all expansion, but it might exceed the available memory. Smaller values are often acceptable. Better solutions might be found in the future, and this parameter suppressed. Apart from memory usage, this number has no consequence on the algorithm.

18.7.19 CLEAN_NCYCLE

Maximum number of Major Cycles for the SDI and CLARK methods.

18.7.20 CLEAN_NGOAL

Number of clean components to be selected in a Cycle in the ALMA heterogeneous array cleaning method.

18.7.21 CLEAN_NITER

This is the maximum number of components the program will accept to subtract. Once it has been reached, the program starts the restoration phase.

If 0, the program will guess a number, based on the image size and maximum signal-to-noise ratio, and specified residual level CLEAN_ARES and/or CLEAN_FRES.

Short form is NITER.

This may be overseded by the /NITER option which imposes a limit per plane through an array of values.

18.7.22 CLEAN_NKEEP

This is an integer specifying the minimum number of Clean components before testing if Cleaning has converged. The convergence is criterium is a comparison of the cumulative flux evolution separated by CLEAN_NKEEP components. If th

IF CLEAN_NKEEP is 0, CLEAN will ignore this convergence criterium, and continue clean until the CLEAN_NITER, CLEAN_ARES or CLEAN_FRES criteria indicate to stop.

With CLEAN_NKEEP > 0, CLEAN will explore the stability of the total clean flux over the last CLEAN_NKEEP iterations. For a positive (resp. negative) source, if the Clean flux becomes smaller (resp. larger) than the Clean flux CLEAN_NKEEP iterations earlier, CLEAN will stop.

Using CLEAN_NKEEP about 70 is a reasonable value. Some special cases (faint extended sources) may require larger values of CLEAN_NKEEP.

18.7.23 CLEAN_POSITIVE

The minimum number of positive components before negative ones are selected.

18.7.24 CLEAN_RESIDUAL

An integer code indicating what the Clean image will contain. Its usage is reserved for tests or specific algorithm using the basic CLEAN as an engine.

-1 The CLEAN image contains only the Clean components, not convolved with the Clean beam.
0 Normal behaviour: Convolve Clean components with Clean beam and add the residuals.
1 Convolve Clean components, but do not add the residuals.

18.7.25 CLEAN_SEARCH

Fraction of peak response of the primary beams coverage beyond which no Clean component is searched in a Mosaic deconvolution.

The default is 0.2.

18.7.26 CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (BEAM_FITTED variable) when 0. The default is 0.35.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

18.7.27 CLEAN_SMOOTH

Smoothing factor between different scales in the MRC and MULTISCALE methods. The default is 2 or 4 for MRC depending on image size. It is sqrt(3) for MULTISCALE and may be set to larger values if needed.

18.7.28 CLEAN_SPEEDY

Speed-up factor for the CLARK major cycles. The default is 1.0. Larger values may be used, but at the expense of possible instabilities of the algorithm.

18.7.29 CLEAN_STOP

CLEAN_STOP is a 2-elements character string that defines the Clean

```
stopping criterium.

If CLEAN_STOP is not empty, it supersedes the older CLEAN_ARES,
CLEAN_FRES and CLEAN_NITER variables (and their even older ARES, FRES,
NITER historical equivalent), and even the CLEAN_NKEEP variable, provid-
ing a more flexible syntax to specify the stopping criterium.

The general syntax is
  LET CLEAN_STOP Value [Unit]
  LET CLEAN_STOP   1 Sigma     Indicates stopping at 1 times the noise level
  LET CLEAN_STOP   10 mJy      Indicates stopping at 0.01 Jy/beam
  LET CLEAN_STOP   1  %        Indicates stopping at 0.01 of the peak value
  LET CLEAN_STOP   100 Iterations  Set the max number of iterations to 100
  LET CLEAN_STOP   20 Stop     Indicates convergence is tested over 20 itera
If  no  unit is specified, the last specified unit is used instead. Sub-
units (mJy, Jy, milliJy, microJy, etc...)  are allowed for convenience.

CLEAN_STOP overwrites the previously specified  CLEAN_ARES,  CLEAN_FRES,
CLEAN_NITER and CLEAN_NKEEP values when (and only when) CLEAN is execut-
ed.


2 CLEAN_TRUNCATE

  Fraction of peak response of the primary beams  coverage  under  which
the Sky brightness image is blanked in a Mosaic deconvolution.

  It is also used as a default truncation value in the PRIMARY command.

The default is 0.2.
```

18.7.30 CLEAN_WORRY

```
  Worry  factor  in the MULTISCALE method for convergence. It propagates
the S/N from one iteration to the other, so that if this  S/N  degrades,
the  method  stops.  Default  is 0 (no propagation, and hence no test on
S/N).  The value should be < 1.0 in all cases.
```

18.7.31 METHOD

```
  Method used for the deconvolution. Can be HOGBOM, MULTI, MRC,
  SDI or CLARK.
```

18.7.32 Old_Names:

Some of the CLEAN parameters have kept their old names: BLC, TRC and BEAM_PATCH (which are seldom used)

Others have equivalent short names: ARES, FRES, GAIN, NITER for which the CLEAN_ prefix may be omitted.

MAJOR, MINOR, ANGLE (which were used to control the Clean beam size) have been replaced by BEAM_SIZE[3]

18.7.33 BLC

These are the (pixel) coordinates of the Bottom Left Corner of the cleaning box. The default (0,0) means the bottom left quarter (Nx/4,Ny/4). If a SUPPORT is defined, BLC is ignored.

18.7.34 TRC

These are the (pixel) coordinates of the Top Right Corner of the cleaning box. The default (0,0) means the top right quarter (3*Nx/4,3*Ny/4). If a SUPPORT is defined, TRC is ignored.

18.7.35 MAJOR

This was the major axis (FWHP) of the Gaussian restoring beam. Replaced by BEAM_SIZE[1] instead.

18.7.36 MINOR

This was the minor axis (FWHP) of the Gaussian restoring beam. Replaced by BEAM_SIZE[2] instead.

18.7.37 ANGLE

This was the position angle (from North towards East, i.e. anticlockwise) of the major axis of the Gaussian restoring beam. Replaced by

BEAM_SIZE[3] instead.

18.7.38 BEAM_PATCH

The dirty beam patch to be used for the minor cycles in CLARK and MRC method. It should be large enough to avoid doing too many major cycles, but has practically no influence on the result. This size should be specified in pixel units. Reasonable values are between N/8 and N/4, where N is the number of map pixels in the same dimension. If set to N, the CLARK algorithm becomes identical to the HOGBOM algorithm.

18.8 DISCARD

[CLEAN\]DISCARD BufferName

Discard (i.e. remove from the memory and the list of known variables) the specified buffer.

Only the most important buffers (e.g. DIRTY, CLEAN, etc...) can be discarded. The current list is:
CCT, CLEAN, CONTINUUM, DIRTY, MASK, PRIMARY, RESIDUAL, SINGLEDISH, SKY, UVCONT, UVSELF, UV_MODEL

18.9 DUMP

[CLEAN\]DUMP [U]

Dump on screen the control parameters of the different CLEAN deconvolution algorithms, mainly for debugging purpose. "DUMP U" dumps the parameters as input by the user while "DUMP" dumps the parameters really used and/or modified by the deconvolution algorithm.

18.10 FIT

[CLEAN\]FIT [Plane] | [CHANNEL First Last] [FIELD First Last] [/THRESHOLD Value] [/JVM_FACTOR [NoCircle]]

Fit the dirty beam to obtain the clean beam parameters.

For single fields, "FIT Plane" can be used to fit just one of the possible Frequency dependent beam planes. Note that the number of beam plane

can be different from that of image channels.

For mosaics, dirty beams can be up to 4-D, depending on Frequency and Field number. Keywords CHANNEL (or equivalently PLANE) and FIELD allow to restrict the fit to the specified range of Channels or Field numbers. All ranges are considered

Note that an automatic FIT is performed by all CLEAN algorithms when needed. The appropriate Frequency plane corresponding to each image channel is used. For mosaics, CLEAN takes the average of the fitted beam parameters on all fields.

User specified values can be used instead (see HELP FIT FixedValues).

18.10.1 FIT FixedValues

Clean beam parameters can also be specified by the users, by loading the BEAM_SIZE array with values other than its default values (0,0,0). In this case, no automatic fit will be performed.

In general, we discourage this usage, as it may result in a very improper flux scaling if the beam size is inappropriate. This mode should be reserved to special cases where the beam cannot be properly fitted, or to have a circular beam by taking as beam size the geometrical mean of the fitted major and minor sizes.

Fixing the beam size should be associated to the FIT /JVM command to minimize (but not suppress !...) errors in total flux scaling.

18.10.2 FIT Results

Beam parameters are returned in the BEAM_FITTED array which is analogous to the BEAM_SIZE one (Major axis, Minor axis in arcsec, and Position angle, in degrees East from North), and in the BEAM_JVM variable, that contains the scaling factor of the residuals.

18.10.3 FIT /JVM_FACTOR

[CLEAN\]FIT [Plane] | [CHANNEL First Last] [FIELD First Last] [/THRESHOLD Value] /JVM_FACTOR [NoCircle]

This option instructs the FIT command to estimate the so-called JvM factor (from Jorsater van Moorsel 1995), which is the ratio of Clean beam

area to Dirty beam area. The value is returned into the BEAM_JVM variable.

The Dirty beam area is an ill-defined quantity (if no Zero spacing is included, the Dirty beam area tends toward Zero when a sufficiently large area is included). It is defined here by integrating only up to the first null of the Dirty beam.

In case of Dirty beams presenting shoulders around the main beam, this ratio may be used to adjust the contribution of residuals to the CLEAN image so that the flux is better preserved. For well-behaved beams, it should be quite close to 1.

NoCircle is an argument which, if present, indicates that the Clean and Dirty beam should not be circularized to derive the factor; this should be reserved for debugging.

18.10.4 FIT /THRESHOLD

[CLEAN\]FIT /THRESHOLD Value [/JVM_FACTOR]

Use Value instead of CLEAN_SIDELOBE to fit only regions of the beam above Value by a Gaussian. Gaussian fit may give different sizes for significantly non-Gaussian beams.

18.10.5 FIT BEAM_SIZE

BEAM_SIZE is a Real array of size 3 containing the user-specified values for the Clean beam size, with BEAM_SIZE[1] the Major axis, BEAM_SIZE[2] the Minor axis (both in arcsec) and BEAM_SIZE[3] the Position angle (in degrees, North towards East).

If BEAM_SIZE[1] and BEAM_SIZE[2] are 0, the CLEAN or FIT commands will determine the Clean beam size (available in variable BEAM_FITTED similar to BEAM_SIZE) by adjusting the dirty beam main peak.

If only one of BEAM_SIZE[1] and BEAM_SIZE[2] is non zero, it will be used as the Clean beam size for a circular beam.

If both are non zero, BEAM_SIZE totally specifies the Clean beam size. BEAM_FITTED is thus set to the BEAM_SIZE values when used.

18.10.6 FIT CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (BEAM_SIZE variable) when 0. The default is 0.30.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

18.11 MAP_COMBINE

```
   [CLEAN\]MAP_COMBINE OutCube CODE In1 In2 [Off] [/FACTOR A1 A2]
[/THRESHOLD T1 T2] [/BLANKING Bval]
```

Combine in different ways two input images (or data cubes)...

```
MAP_COMBINE also supports an "intuitive" , more mathematical syntax,
    MAP_COMBINE OutCube = [A1*]In1 Oper [A2*]In2 [Off]
[/THRESHOLD T1 T2] [/BLANKING Bval]
```

where Oper can be any of the values allowed for CODE, or one of the 4 standard operators, +-/*, and scale factors are given in a natural way.

OutCube can be a file name or an existing SIC Image variable. The distinction is made by the existence of a "." in the name. If it is a file, it is created like the In1 Variable If it is a SIC Image variable, it must exist and match the shape of In1.

In1 can be a file name or an existing SIC Image variable.

In2 can be a file name or an existing Image variable. The rank of In2 must be smaller than or equal to that of In1, and other dimensions must match.

CODE is the Operation Code: see HELP MAP_COMBINE CODE for details. Off is an optional offset for the operation.

A1 and A2 are scale factors applied to In1 and In2 if needed (default 1).

T1 and T2 are thresholds below which no result is computed (default none).

18.11.1 MAP_COMBINE CODE

```
    [CLEAN\]MAP_COMBINE OutCube CODE In1 In2 [Off] [/FACTOR A1 A2]
  [/THRESHOLD T1 T2] [/BLANKING Bval]

  CODE is the operation code. Allowed values are
ADD      or PLUS    OutCube = A1*In1 + A2*In2 + Off
SUBTRACT or MINUS   OutCube = A1*In1 - A2*In2 + Off
DIVIDE   or OVER    OutCube = A1*In1 / A2*In2 + Off
MULTIPLY or TIMES   OutCube = A1*In1 * A2*In2 + Off
OPACITY             OutCube = -Log( A1*In1 / A2*In2 + Off)
INDEX               OutCube = Log( A1*In1 / A2*In2) / Log(Nu1/Nu2)
where Nu1 is the Frequency of In1, and Nu2 that of In2.  Off is 0 if not
specified.

  With the "intuitive" mathematical syntax,
      MAP_COMBINE OutCube = [A1*]In1 Oper [A2*]In2 [Off]

Oper  can  be any of the above values for CODE, or one of the 4 standard
operators, +-/*, and scale factors are given in a natural way,  default-
ing to 1 if not present.
```

18.11.2 MAP_COMBINE /BLANKING

```
    [CLEAN\]MAP_COMBINE OutCube CODE In1 In2 [/FACTOR A1 A2]
  [/THRESHOLD T1 T2] /BLANKING Bval

Specify  the Blanking value to be used in the OutCube. If not specified,
the Blanking from In1 is used instead, and that of In2  if  In1  has  no
Blanking.
```

18.11.3 MAP_COMBINE /FACTOR

```
    [CLEAN\]MAP_COMBINE OutCube CODE In1 In2 /FACTOR A1 A2
  [/THRESHOLD T1 T2] [/BLANKING Bval]

Specify  the  factors to apply to In1 and In2. Default is 1.0.  This op-
tion is not available in the "intuitive" math syntax, where the  factors
are specified directly.
```

18.11.4 MAP_COMBINE /THRESHOLD

```
    [CLEAN\]MAP_COMBINE OutCube CODE In1 In2 [/FACTOR A1 A2]
  /THRESHOLD T1 T2] [/BLANKING Bval]

Specify  the thresholds above which the computation is valid for In1 and
```

In2. For pixels below this threshold, the OutCube is blanked.

Default is no threshold (-huge(0.)).

18.12 MAP_COMPRESS

[CLEAN\]MAP_COMPRESS WhichOne Nc [First TYPE] [Output]

Resample (in frequency/velocity) a data cuvbe by averaging NC adjacent channels.

WhichOne indicates which data cube must be compressed. Allowed values are
- DIRTY, CLEAN or SKY
- *, which means all of the previous existing ones
- Any SIC Image Variable that is not a built-in buffer in IMAGER
- An existing GILDAS data file

MAP_INTEGRATE only works on 3-D data cubes, replacing them by 2-D images.

Output is a string indicating where the results will be placed.

18.12.1 MAP_COMPRESS Output

[CLEAN\]MAP_COMPRESS WhichOne Nc [First TYPE] [Output]

Output is a string indicating where the results will be placed. It can be
- Any SIC Image Variable that is not a built-in buffer in IMAGER
  The variable must exist and be of adequate size.
- A file name. The command will write the result as a GILDAS data file
  with this name.

The distinction between SIC Image Variable and Filenames is based on the presence of a dot (".") or semi-colon (":") that indicate that the string refers to a filename.

Output is compulsory unless WhichOne referes to one of the allowed built-in buffer names (DIRTY, CLEAN or SKY) and forbidden if WhichOne is *. When Output is not present, the result are written in place on the specified buffers.

18.13 MAP_INTEGRATE

[CLEAN\]MAP_INTEGRATE WhichOne Min Max Type [Output]

Compute the integrated intensity map(s) over the specified range from the specified data cube. Type can be VELOCITY FREQUENCY or CHANNELS.

WhichOne indicates which data cube must be compressed. Allowed values are
- DIRTY, CLEAN or SKY
- *, which means all of the previous existing ones
- Any SIC Image Variable that is not a built-in buffer in IMAGER
- An existing GILDAS data file

MAP_INTEGRATE only works on 3-D data cubes, replacing them by 2-D images.

Output is a name that indicates where the results will be placed,

18.13.1 MAP_INTEGRATE Output

```
[CLEAN\]MAP_INTEGRATE WhichOne Min Max Type [Output]
```

Output is a string indicating where the results will be placed. It can be
- Any SIC Image Variable that is not a built-in buffer in IMAGER
  The variable must exist and be of adequate size.
- A file name. The command will write the result as a GILDAS data file
  with this name.

The distinction between SIC Image Variable and Filenames is based on the presence of a dot (".") or semi-colon (":") that indicate that the string refers to a filename.

Output is compulsory unless WhichOne referes to one of the allowed built-in buffer names (DIRTY, CLEAN or SKY) and forbidden if WhichOne is *. When Output is not present, the result are written in place on the specified buffers.

18.14 MAP_REPROJECT

```
[CLEAN\]MAP_REPROJECT Output Input
[/BLANKING Bval Eval]    [/LIKE Template]
[/PROJECTION Type Cx Cy [Angle]]
[/SYSTEM Type [Equinox]]
[/X_AXIS Nx Ref Val Inc]
[/Y_AXIS Ny Ref Val Inc]
```

Resample spatially a Map or Cube according to the specified projection (and coordinate system). The characteristics of the Output Cube can be controlled by the /LIKE option, or explicitely by a combination of /PRO-

JECTION, /SYSTEM, /X_AXIS, /Y_AXIS options.

Input can be a GILDAS data file, or a SIC image variable. Output is a GILDAS data file that is created by the command.

18.14.1 MAP_REPROJECT /BLANKING

```
[CLEAN\]MAP_REPROJECT Output Input
/BLANKING Bval Eval    [/LIKE Template]
[/PROJECTION Type Cx Cy [Angle]]
[/SYSTEM Type [Equinox]]
[/X_AXIS Nx Ref Val Inc]
[/Y_AXIS Ny Ref Val Inc]
```

Specify the desired Blanking value. If not present, this is inherited from the Input, or the Template if specified.

18.14.2 MAP_REPROJECT /LIKE

```
[CLEAN\]MAP_REPROJECT Output Input
/LIKE Template [/BLANKING Bval Eval]
```

Specify the shape and projection of the desired Output Cube from a Template. Template is either a GILDAS data file, or a SIC Header (or Image) variable.

The Blanking value, if not specified through /BLANKING, will be inherited from the Template.

The /LIKE option is incompatible with the /PROJECTION, /SYSTEM, /X_AXIS and /Y_AXIS options.

18.14.3 MAP_REPROJECT /PROJECTION

```
[CLEAN\]MAP_REPROJECT Output Input
[/BLANKING Bval Eval]
/PROJECTION Type Cx Cy [Angle]
[/SYSTEM Type [Equinox]]
[/X_AXIS Nx Ref Val Inc]
[/Y_AXIS Ny Ref Val Inc]
```

Specify the desired Projection center. If Angle is not specified, it is set to 0. Any argument with a value of * is left unchanged from the Input Cube value.

This option is incompatible with the /LIKE option.

18.14.4 MAP_REPROJECT /SYSTEM

```
[CLEAN\]MAP_REPROJECT Output Input
[/BLANKING Bval Eval]
[/PROJECTION Type Cx Cy [Angle]]
/SYSTEM Type [Equinox]
[/X_AXIS Nx Ref Val Inc]
[/Y_AXIS Ny Ref Val Inc]
```

Specify the desired coordinate System. Equinox default to 2000.0 for EQUATORIAL system if not specified.

This option is incompatible with the /LIKE option.

18.14.5 MAP_REPROJECT /X_AXIS

```
[CLEAN\]MAP_REPROJECT Output Input
[/BLANKING Bval Eval]
[/PROJECTION Type Cx Cy [Angle]]
[/SYSTEM Type [Equinox]]
/X_AXIS Nx Ref Val Inc
[/Y_AXIS Ny Ref Val Inc]
```

Specify the desired X axis sampling. Any argument with a value of * is left unchanged from the Input Cube value.

This option is incompatible with the /LIKE option.

18.14.6 MAP_REPROJECT /Y_AXIS

```
[CLEAN\]MAP_REPROJECT Output Input
[/BLANKING Bval Eval]
[/PROJECTION Type Cx Cy [Angle]]
[/SYSTEM Type [Equinox]]
[/X_AXIS Nx Ref Val Inc]
/Y_AXIS Ny Ref Val Inc]
```

Specify the desired Y axis sampling. Any argument with a value of * is left unchanged from the Input Cube value.

This option is incompatible with the /LIKE option.

18.15 MAP_RESAMPLE

```
[CLEAN\]MAP_RESAMPLE WhichOne [Nc Ref Val Inc] [Result] [/LIKE Mold]
```

Resample 3-D data cubes on a different velocity scale.
Nc new number of channels
Ref New reference pixel
Val New velocity at reference pixel
Inc Velocity increment
WhichOne indicates which data cube must be compressed. Allowed values are
- Built-in buffers DIRTY, CLEAN, SKY, SINGLE or *, which means all of the previous existing ones. Resampling is done in place.
- Any SIC Image Variable that is not a built-in buffer in IMAGER.
- A GILDAS data file

For the two later cases, the Result argument indicates where the resampled data cube is stored. Result can be a file name or an existing SIC variable of adequate dimensions.

MAP_RESAMPLE only works on 3-D data cubes.

18.15.1 MAP_RESAMPLE /LIKE

```
[CLEAN\]MAP_RESAMPLE WhichOne [Result] [/LIKE Mold]
```

Resample 3-D data cubes on the same velocity scale than the specified Mold. Mold can be a SIC variable or GILDAS data file.

18.16 MAP_SMOOTH

```
[CLEAN\]MAP_SMOOTH WhichOne Nc [Result] [/ASYMMETRIC]
```

Smooth a 3-D data cubes by Nc channels along the velocity axis.
Nc number of channels to be averaged in smoothing.
WhichOne indicates which data cube must be compressed. Allowed values are
- Built-in buffers DIRTY, CLEAN, SKY, SINGLE or *, which means all of the previous existing ones. Smoothing is done in place.
- Any SIC Image Variable that is not a built-in buffer in IMAGER.
- A GILDAS data file

For the two later cases, the Result argument indicates where the smoothed data cube is stored. Result can be a file name or an existing SIC variable of adequate dimensions.

By default, the smoothing does not modify the spectral sampling of the cubes. While this naturally occurs when the smoothing number Nc is odd, the Nc even case requires a specific handling. For Nc even, the

smoothed channel at channel "i" sums up half of channel i-Nc/2, half of channels i+Nc/2 and all channels in between. Thus for Nc=2, this is a Hanning smoothing.

Like for UV_SMOOTH, this behaviour can be modified by option /ASYMMETRIC.

18.16.1 MAP_SMOOTH /ASYMMETRIC

[CLEAN\]MAP_SMOOTH WhichOne Nc [Result] /ASYMMETRIC

Make no attempt to preserve the spectral sampling when Nc is even: the reference channel is then shifted by half the channel spacing.

For odd Nc, the smoothing is naturally symmetric, so the option has no impact.

18.17 SPECIFY

[CLEAN\]SPECIFY Keyword [Values...] [/FOR SicVariable]

Specify or modify some information in internal buffers or in a Sic Image Variable.

Recognized values for Keyword are BLANKING, FREQUENCY, LINENAME, VELOCITY, TELESCOPE

If present the /FOR option indicates which buffer or Image variable should be affected. If no /FOR option is given, all relevant available buffers are affected (CLEAN, DIRTY, UV, etc...)

18.17.1 SPECIFY BLANKING

SPECIFY BLANKING Value /FOR SicVariable

Specify or Modify the blanking value in the specifed Image variable SicVariable. Option /FOR is mandatory for this action.

18.17.2 SPECIFY FREQUENCY

SPECIFY FREQUENCY Value

Modify the rest frequency and recompute the velocity scale accordingly. Value is the new rest frequency in MHz

18.17.3 SPECIFY LINENAME

SPECIFY LINENAME Name
Change the spectral line name.

18.17.4 SPECIFY VELOCITY

SPECIFY VELOCITY Value
Modify the source velocity and recompute the rest frequency scale accordingly Value is the new velocity in km/s.

18.17.5 SPECIFY TELESCOPE

SPECIFY TELESCOPE Name

Add or Replace the telescope section with the parameters (name, size, position) for the specified telescope name, essentially to get the most appropriate beam parameter.

A Telescope section is required for MOSAIC. The beamsize will depend on telescope diameter and frequency, with a telescope dependent factor. The default beam size is 1.13 Lambda/D.

18.18 MOSAIC

[CLEAN\]MOSAIC On|Off

Turn on or off the mosaic mode for deconvolution. Note that a READ PRIMARY command, or a UV_MAP command working on a Mosaic UV Table, automatically switches on the mosaic mode. The program prompt changes to inform the user of the current operating mode for deconvolution.

18.19 MX

[CLEAN\]MX [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY]

Make and deconvolve maps starting from a UV table. It combines UV_MAP and CLEAN in a single step.

The mapping process is identical to UV_MAP. It makes a map from UV data by griding the UV data using a convolving function, and then Fast Fourier Transforming the individual channels. However, MX always produces a

single beam for all channels, thus neglecting frequency change between channels. MX enables to shift the map center and rotate the image, by shifting the phase tracking center and rotating the UV coordinates of the input UV table.

The CLEAN algorithm is similar to the CLARK method, but with major cycles operating directly on the ungridded UV table rather than in the image plane. Accordingly, aliasing affects only of the residuals, not the clean components. It is thus more accurate but also slower than CLARK as it asks for the gridding step at each major cycle. MX also shares the same limitation as CLARK on large sidelobes.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is possible, a sensible value of each parameter will automatically be computed from the context if the value of the corresponding variable is set to its default value, i.e. zero value and empty string. A few variables are initialized to "reasonable" values.

[CLEAN\]CLEAN ?
Will list all main CLEAN_* variables controlling the CLEAN parameters for the current METHOD.

HELP CLEAN Variables will give a more complete list.

[CLEAN\MX ?
Will list all MAP_* and CLEAN_ * variables controlling the MX parameters.

18.19.1 MX Variables:

The list of control variables is (by alphabetic order, with the corresponding old names used by Mapping on the right)

```
New names       [   unit]       -- Description --    % Old Name
BEAM_STEP       [       ]  Channels per dirty beam  % MAP_BEAM_STEP
MAP_CELL        [ arcsec]  Image pixel size
MAP_CENTER      [ string]  RA, Dec of map center, and Position Angle
MAP_CONVOLUTION [       ]  Convolution function     % CONVOLUTION
MAP_FIELD       [ arcsec]  Map field of view
MAP_POWER       [       ]  Maximum exponent of 3 and 5 allowed in MAP_SIZE
MAP_PRECIS      [       ]  Fraction of pixel tolerance on beam matching
MAP_ROBUST      [       ]  Robustness factor        % UV_CELL[2]
MAP_ROUNDING    [       ]  Precision of MAP_SIZE
MAP_SIZE        [       ]  Number of pixels
MAP_TAPEREXPO   [       ]  Taper exponent           % TAPER_EXPO
```

```
MAP_TRUNCATE    [      %]  Mosaic truncation level
MAP_UVCELL      [      m]  UV cell size             % UV_CELL[1]
MAP_UVTAPER     [m,m,deg]  Gaussian taper           % UV_TAPER
MAP_VERSION     [       ]  Code version (0 new, -1 old)


Basic CLEAN parameters
CLEAN_GAIN       [       ] Loop gain
CLEAN_STOP       [       ] Stopping criterium (string)
CLEAN_NITER      [       ] Maximum number of clean components
CLEAN_FRES       [      %] Maximum value of residual (Fraction of peak)
CLEAN_ARES       [Jy/Beam] Maximum value of residual (Absolute)
CLEAN_POSITIVE   [       ] Minimum number of positive components at start
CLEAN_NKEEP      [       ] Min number of components before convergence

Clean beam user specification
BEAM_SIZE       [ 0 0 0 ] Major, Minor (in arcsec) and PA (in Degree)

Method dependent CLEAN parameters
CLEAN_INFLATE    [      ] Maximum Inflation factor for UV_RESTORE (MULTISCAL
CLEAN_NCYCLE     [      ] Max number of Major Cycles (SDI & CLARK methods)
CLEAN_NGOAL      [      ] Max number of comp. in Cycles (ALMA method)
CLEAN_RATIO      [      ] Smoothing factor (MRC default 0: guess, otherwise
CLEAN_SEARCH     [      ] Threshold to search Clean Comp. in a Mosaic (def 0
CLEAN_SIDELOBE   [      ] Min threshold to fit the synthesized beam
CLEAN_SMOOTH     [      ] Smoothing ratio: MRC (def 2 or 4)  and MULTISCALE
CLEAN_SPEEDY     [      ] Speed-up factor (CLARK)
CLEAN_TRUNCATE   [      ] Threshold for restoring a Mosaic (def 0.2)
CLEAN_WORRY      [      ] Worry factor (CLARK and MULTISCALE)

Old (seldom used) names like in MAPPING
BLC              [  pixel] Bottom left corner of cleaning box
TRC              [  pixel] Top right corner of cleaning box
BEAM_PATCH       [  pixel] Size of cleaning beam ** not clear **


Restoration control CLEAN parameters
CLEAN_RESIDUAL   [-1,0,1]  Control how residuals are added or not - For debu
```

18.20 PRIMARY

```
[CLEAN\]PRIMARY [BeamSize] [/TRUNCATE Percent]
```

Apply approximate primary beam correction to a single field deconvolved (CLEAN) image, in order to create the sky brightness image (named SKY). The truncation level is given by default by variable CLEAN_TRUNCATE, and can be overriden by the /TRUNCATE option.

The primary beam model is a simple Gaussian. If BeamSize (in radian) is specified, uses the corresponding half-power beam size to determine the Gaussian beam. If not, the parameters are taken from the telescope parameters, as found in the telescope section of the CLEAN image and the observing frequency from the CLEAN image (e.g. for ALMA it uses 1.13 Lambda/D).

The SKY image is written with extension .lmv-sky by command WRITE.

18.20.1 PRIMARY /TRUNCATE

```
[CLEAN\]PRIMARY [BeamSize] /TRUNCATE Percent
```

Specify the truncation level. Default (in % of peak beam response) is given by the CLEAN_TRUNCATE variable. Values are blanked beyond this.

18.21 READ

```
[CLEAN\]READ Buffer File [/COMPACT] [/FREQUENCY RestFreq]
[/RANGE Min Max Type] [/NOTRAIL]
```

Read the specified internal buffer (BEAM, CCT, CGAINS, CLEAN, DIRTY, MASK, MODEL, PRIMARY, RESIDUAL, SKY, SUPPORT, SINGLEDISH, UV) from input File. Default extension is .uvt for UV, CGAINS and MODEL, and for the others, in order, .beam, .cct, .lmv-clean, .lmv, .msk, .lobe, .lmv-res, for Data Cubes when the extension is given as .fits (see HELP READ FITS for more details).

A subset of all available channels can be indicated through the /RANGE option, even for UV tables.

The corresponding buffer is available as a SIC image-like variable of the same name, so that one can use e.g. HEADER DIRTY command.

READ * Name will attempt to read all existing files of same Name with the standard file types corresponding to the respective buffers.

The /COMPACT option is used to load the ACA-specific internal buffer used in the ALMA joint deconvolution method.

The /NOTRAIL allows to ignore trailing columns in UV data. This is normally not recommended and should be used for debug only. An exception is for "false" mosaics with just 1 field: see HELP READ /NOTRAIL for details.

18.21.1 READ Buffers

```
[CLEAN\]READ Buffer File [/COMPACT]

  The recognized Buffer names for READ are:
  BEAM        Synthesized Dirty beam
  CCT         Clean Component list
  CGAINS      Complex gains for APPLY command
  CLEAN       Deconvolved Clean image
  DIRTY       Dirty image
  PRIMARY     Primary beam
  RESIDUAL    Residual image
  MASK        Spatial mask
  MODEL       UV table for Clean components
  SINGLEDISH  Single-Dish table or data cube
  SKY         Primary beam-corrected deconvolved sky brightness
  SUPPORT     Polygon enclosing the Support for Cleaning
  UV_DATA     UV data table
  UV_FIT      UV_FIT results table (for SHOW UV_FIT)
The  MODEL UV table is for use in conjunction with commands in the CALI-
BRATE\ language, i.e. for Self-Calibration.

The following ones are allowed with the /COMPACT option
  BEAM        Synthesized Dirty beam
  DIRTY       Dirty image
  PRIMARY     Primary beam
  RESIDUAL    Residual image
  UV_DATA     UV data table
```

18.21.2 READ FITS

```
[CLEAN\]READ Buffer File [/COMPACT]

For most buffers of type "Data Cube" (e.g. BEAM, CLEAN, DIRTY,  PRIMARY,
RESIDUAL,  SINGLEDISH, SKY), the input File can be in FITS format.  READ
will check for the appropriate data format  when  appropriate.   If  the
buffer does not accept FITS data format, an error message is issued.

UVFITS data format is not supported for UV data, because a number of im-
portant informations is sometimes stored  in  FITS  Extensions.   UVFITS
format  files should be converted to UV Tables in the GILDAS data format
using the "@ fits_to_uvt" script.
```

18.21.3 READ Optimisation

Reading can be lengthy, especially for ALMA data. IMAGER attempts to minimize read operations by checking if anything has changed since the last command. This capability is enable if the SIC variable MAPPING_OPTIMIZE is non zero, disabled otherwise.

Currently, the READ command always display a message about what reading may have been skipped, and whether this possible optimization has been overridden by user choice.

18.21.4 READ /COMPACT

[CLEAN\]READ Buffer File /COMPACT [/RANGE Min Max Type]

Read the specified internal buffer (UV, MODEL, BEAM, PRIMARY, DIRTY, CLEAN, MASK, CCT) from input File to the "compact array" data area.

18.21.5 READ /FREQUENCY

[CLEAN\]READ Buffer File /FREQUENCY RestFreq [/RANGE Min Max Type]

Read the specified internal buffer and reset the velocity scale to the corresponding rest frequencies. Velocities specified in the /RANGE Min Max VELOCITY option would then refer to this new frequency.

18.21.6 READ /NOTRAIL

[CLEAN\]READ Buffer File /NOTRAIL [/FREQUENCY Freq] [/RANGE Min Max Type]

When reading UV Tables, ignores any trailing column. Trailing columns normally appear for mosaics. However, ALMA sometimes uses known proper motions to shift (by small amounts) phase centers between two observing periods, yielding pseudo-mosaics with tiny (in general insignificant) displacements. The /NOTRAIL option allows to ignore these details.

18.21.7 READ /PLANES

[CLEAN\]READ Buffer File /PLANES Min Max

Read only the last axis "planes" between the First and Last implied by Min and Max. For an LMV-ordered cube, this would be equivalent to
READ Buffer File /RANGE Min Max CHANNEL

However, for transposed data cubes, VML or LVM-ordered for example, this is not true at all when a Frequency/Velocity axis is present: /RANGE would specify a subset of this(yet) axis, while /PLANES always indicate a subset of the 3rd axis of the cube.

Min and Max indicate offsets from Plane 1 and the number of planes Nplane. Thus Max can be negative: it then indicates Last = Nplane-Max. Also Min=0 and Max=0 implies loading all the channels.

18.21.8 READ /RANGE

[CLEAN\]READ Buffer File /RANGE Min Max Type

Load only the channels between the First and Last defined by Min Max and Type. Type can be CHANNEL, VELOCITY or FREQUENCY. If the Data cube has no Frequency/Velocity axis, only CHANNEL is allowed.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies loading all the channels.

The /RANGE option is not allowed for FITS files.

For UV tables with more than 1 Stokes parameter, which are **NOT** fully supported by IMAGER, the meaning of "channel" is ambiguous.

18.21.9 READ SINGLE

[CLEAN\]READ SINGLE File[.ext] [/RANGE Min Max Type]

Read the "Single Dish" data set. It can be a Class table (.tab), or a 3-D data cube (.lmv order). The data set is available as the SINGLE image variable.

If it is a 3-D data cube, the SHORT image variable is also defined (the data area is shared with that of the SINGLE imager variable, but the Headers are separate). If it is a Class table, the SHORT image becomes undefined. It will be computed by commands UV_SHORT (see HELP UV_SHORT Step_1) or XY_SHORT.

18.22 SUPPORT

[CLEAN\]SUPPORT [Polygon] [/CURSOR] [/MASK] [/PLOT] [/RESET] [/VARIABLE]

Define and/or plot the support inside which to search for CLEAN components. The support can be defined through a mask (see /MASK option) or a polygon, depending on the selected options. The /PLOT option can then be used to plot it.

A polygon stored in a file, or in a Sic variable (/VARIABLE option), can be loaded as the polygon support.

See also the [ADVANCED\]MASK command for a complementary way of defining supports, in particular 3-D supports.

18.22.1 SUPPORT /CURSOR

[CLEAN\]SUPPORT /CURSOR

With option /CURSOR, SUPPORT calls the interactive cursor to define the polygon summits. Type any key to go to next summit, D to correct the last one and type E to end the polygon definition. The last polygon side will then appear. The polygon definition may be aborted by typing Q. For graphical displays, you may use the mouse buttons for the commands. The left mouse button draws a vertex, the middle mouse button deletes the last vertex, and the right mouse button ends the polygon definition.

The resulting support is available in the Sic structure SUPPORT:
- SUPPORT%NXY [Integer] Number of summits
- SUPPORT%X [Double] X coordinates
- SUPPORT%Y [Double] Y coordinates

18.22.2 SUPPORT /MASK

[CLEAN\]SUPPORT /MASK or [ADVANCED\]MASK USE

Use the mask defined by READ MASK or by the MASK command as the clean support. This does not suppress the current polygon: it can be re-instated by a simple SUPPORT command with no argument.

The Mask can be a 3-D array: CLEAN will find out which mask plane must be used for each spectral channel.

Caution: [CLEAN\]READ MASK and [ADVANCED\]MASK command do not perform an implicit SUPPORT /MASK command. Only the ADVANCED\MASK USE command does it explicitely.

18.22.3 SUPPORT /PLOT

```
[CLEAN\]SUPPORT [Name] /PLOT
```

Plot the current (or specified) polygon, or plot the current mask (if it is 2-D only).

For 3-D masks, use the VIEW MASK command instead.

18.22.4 SUPPORT /RESET

```
[CLEAN\]SUPPORT /RESET
```

Reset any support to default. This deletes the current polygon support. This does not unload any mask defined by READ MASK. Such a mask can be re-instated by SUPPORT /MASK.

18.22.5 SUPPORT /THRESHOLD

```
[CLEAN\]SUPPORT /THRESHOLD [Raw Smooth [Length [Guard]]
```

*** Obsolete - see [ADVANCED\]MASK THRESHOLD instead ***

18.22.6 SUPPORT /VARIABLE

```
[CLEAN\]SUPPORT VarName /VARIABLE
```

Load the support from the Sic variable VarName. VarName can be an array of the form:

```
VarName[NXY,2]  Real or Double
```

or a structure of the form (i.e. same as output):

```
VarName%NXY             Integer or Long
VarName%X[VarName%NXY]  Real or Double
VarName%Y[VarName%NXY]  Real or Double
```

18.23 UV_BASELINE

```
[CLEAN\]UV_BASELINE [Degree] [/CHANNELS Channel_List]
[/FREQUENCY List Of Frequencies] [/RANGE Min Max [TYPE]]
[/VELOCITY List of Velocities] [/WIDTH Width [TYPE]]
[/FILE FileIn [FileOut]]
```

Subtract a continuum from a line UV data set, by fitting a baseline for

each visibility. The channels to be ignored in this process (i.e. the ones including the line emission) can be specified either by the /FREQUENCY or /VELOCITY options in combination with the /WIDTH option, or by a channel list with /CHANNELS or a range (of channels, frequencies or velocities) with /RANGE.

By default, the command works on the current UV data, unless the option /FILE is specified.

With no options (or with only /FILE), the command behaves as UV_BASELINE /CHANNELS PreviewChannels. See UV_PREVIEW for details.

18.23.1 UV_BASELINE /CHANNELS

```
[CLEAN\]UV_BASELINE /CHANNELS Channel_List
```

Channel_List must be a 1-D SIC variable containing the list of channels to filter out.

18.23.2 UV_BASELINE /FILE

```
  [CLEAN\]UV_BASELINE [Degree] [/CHANNELS Channel_List]
[/FREQUENCY List Of Frequencies] [/RANGE Min Max [TYPE]]
[/VELOCITY List of Velocities] [/WIDTH Width [TYPE]]
/FILE FileIn [FileOut]
```

Use the UV data in the file FileIn, and write the continuum-free visibilities in the file FileOut. If FileOut is not specified, "-line" is appended to the name indicated by FileIn.

18.23.3 UV_BASELINE /FREQUENCY

```
[CLEAN\]UV_BASELINE /FREQUENCY F1 [... [Fn]] [/WIDTH Width [TYPE]]
```

Specify around which frequencies the line emission should be filtered. Frequencies F1 to Fn must be in MHz. The full width of the filtering window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

Tip: it can be convenient to have a list of SIC variables containing the frequencies of the most intense spectral lines, e.g.

```
HCO10 = 89188.52
```

18.23.4 UV_BASELINE /RANGE

```
[CLEAN\]UV_BASELINE /FREQUENCY F1 [... [Fn]] /RANGE Min Max [TYPE]
```

Indicate that channels between the First and Last defined by Min Max and Type contain line emission and should be ignored in the baseline fitting. Type can be CHANNEL, VELOCITY or FREQUENCY.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies loading all the channels.

18.23.5 UV_BASELINE /VELOCITY

```
[CLEAN\]UV_BASELINE /VELOCITY V1 [... [Vn]] [/WIDTH Width [TYPE]]
```

Specify around which velocities the line emission should be filtered. Velocities V1 to Vn must be in km/s. The full width of the filtering window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

18.23.6 UV_BASELINE /WIDTH

```
[CLEAN\]UV_BASELINE /FREQUENCY F1 [... [Fn]] /WIDTH Width [TYPE]
```

Specify the full width of the window around every frequency given in the /FREQUENCY option. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width. The default width is the current channel width.

18.24 UV_CHECK

```
[CLEAN\]UV_CHECK  Beams|Nulls|Integration [/FILE FileIn]
```

Check current UV data for weight consistency, or the UV data in the specified file if the /FILE option is present.

BEAMS
List which channel range can be processed with the same synthesized beam. See HELP UV_CHECK BEAM_RANGE for further details.

NULLS

```
Check if there are null visibilities with non-zero weights, and flag
them if found.

INTEGRATION
Attempt to estimate the integration time. This is done by comparing
the time stamps of consecutive observations, unless an INTEGRATION
column is present.
```

18.24.1 UV_CHECK /FILE

```
[CLEAN\]UV_CHECK  Beams|Nulls|Integration /FILE FileIn

Check the UV data in the specified file.
```

18.24.2 UV_CHECK BEAM_RANGES

```
UV_CHECK BEAMS returns a variable named BEAM_RANGES[3,Nbeam] where
Nbeam is the number of different synthesized beams found in the UV
data (based on the visibility weights).

BEAM_RANGES[1,i]  is the first channel of Beam number i
BEAM_RANGES[1,2]  is the last channel of Beam number i
BEAM_RANGES[1,3]  is the total weight of visibilities of Beam number i

BEAM_RANGES is later used by UV_PREVIEW to discard edge channels
that often do not have the same weights as the bulk of channels.

Variable NBEAM_RANGES indicates the number of beam ranges. It is -1
if UV_CHECK has not yet been used.

Most automatic imaging processes will assume BEAM_RANGES is relatively
simple (2 edges and a dominant main  range)  and  simply  discard  the
edges
if any, or complain if is too complex.

For complex datasets with NBEAM_RANGES greater than 2 or 3,
BEAM_RANGES could be used to segment the imaging and deconvolution
process into ranges having a common beam. Setting MAP_BEAM_STEP
to 1 is another possibility, but that creates 1 beam per channel,
thus doubling the used space.
```

18.25 UV_COMPRESS

```
[CLEAN\]UV_COMPRESS [Nc]  [/CONTINUUM] [/FILE FileIn FileOut]

Resample the UV data by averaging NC adjacent channels.
```

With no /FILE option, the current UV table obtained by READ UV is resampled. All further UV commands work on the "Resampled" UV table. The "Resampled" UV table is a simple copy of the original one after a READ UV command, or after a UV_RESAMPLE or UV_COMPRESS commands with no arguments and no options.

With the /FILE option, the UV data is read from file FileIn and the resampled output is written in file FileOut.

With the /CONTINUUM option, the argument can be omitted, and the best value for Nc is derived from the current field of view, to avoid bandwidth smearing.

18.25.1 UV_COMPRESS /CONTINUUM

```
[CLEAN\]UV_COMPRESS /CONTINUUM [/FILE FileIn FileOut]
```

Resample the UV data by averaging Nc adjacent channels. The value for Nc is derived from the current field of view, to avoid bandwidth smearing. This command is intended to provide a compact form of UV data for continuum-only imaging, through the UV_MAP /CONTINUUM command. Contrary to the output of the UV_CONTINUUM command, it is also suitable for Self-Calibration (and it is more compact).

With no /FILE option, the current UV table obtained by READ UV is resampled.

With the /FILE option, the UV data is read from file FileIn and the resampled output is written in file FileOut.

Caution: /CONTINUUM may not work with the /FILE option if the Minimun and Maximum baselines are not available in the FileIn header.

18.25.2 UV_COMPRESS /FILE

```
[CLEAN\]UV_COMPRESS Nc /FILE FileIn FileOut
```

Use the UV data in the file FileIn, and write the data after averaging by NC adjacent channels in the file FileOut.

18.26 UV_CONTINUUM

```
[CLEAN\]UV_CONTINUUM [Naver] [/INDEX Value [Frequency]]
[/RANGE Min Max TYPE]
```

Transform the (presumably spectral line) UV data set loaded by READ UV into a "continuum" data set.

The transformation selects line channels defined by the /RANGE option, average them by groups of Naver contiguous channels, and concatenate the resulting visibilities into a "continuum" UV table. If not present, Naver is derived from the current field of view MAP_FIELD and required precision, MAP_PRECIS to avoid bandwidth smearing at field egdes.

For each output channel, and for all visibilities, the U and V coordinates are rescaled to the mean observing frequency, and the resulting (single-channel) visibilities are concatenated into the "continuum" UV data set. The continuum dataset becomes the current UV data. Flagged channels are ignored: this allows to mask channels containing spectral lines (see UV_FILTER).

If no /RANGE option is defined, all channels are selected.

18.26.1 UV_CONTINUUM /INDEX

```
[CLEAN\]UV_CONTINUUM Naver [First Last] /INDEX Value [Frequency]
```

Specify which spectral index to be used when scaling the visibilities as a function of frequency.

TO BE IMPLEMENTED: Frequency, if specified, will be used as a reference frequency instead of the mean observing frequency.

18.26.2 UV_CONTINUUM /RANGE

```
[CLEAN\]UV_CONTINUUM Naver  /RANGE Min Max TYPE
[/INDEX Value [Frequency]]
```

Indicate that channels between the First and Last defined by Min Max and Type should be considered to produce the output continuum UV table. Type can be CHANNEL, VELOCITY or FREQUENCY.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies all the channels.

18.27 UV_EXTRACT

```
[CLEAN\]UV_EXTRACT /RANGE Min Max [TYPE]  [/FILE FileIn [FileOut]]
```

Extract a subset of all available channels. The command works on the current UV buffer, unless the /FILE option is specified

18.27.1 UV_EXTRACT /FILE

[CLEAN\]UV_EXTRACT /RANGE Min Max [TYPE] /FILE FileIn [FileOut]

Extract the specified range from the UV data in the file FileIn, and write the resulting visibilities in the file FileOut. If FileOut is not specified, "-ext" is appended to the name indicated by FileIn.

18.27.2 UV_EXTRACT /RANGE

[CLEAN\]UV_EXTRACT /RANGE Min Max [TYPE] [/FILE FileIn [FileOut]]

Indicate that channels between the First and Last defined by Min Max and Type contain line emission and should be filtered out. Type can be CHANNEL, VELOCITY or FREQUENCY.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies all the channels.

18.28 UV_FIELDS

[CLEAN\]UV_FIELDS RaVar DecVar [Unit] [/CENTER Ra Dec] /FILE File

Replace fields ID by Fields offsets in a Mosaic UV Table. This can only work for a Mosaic UV table with Fields offsets, or Phase center offsets. It cannot operate on a UV table with Pointing center offsets.

For files imported from UVFITS, the Fields offsets are found in the AIPS SU Table which is not decoded by the FITS command, but only through the DEFINE FITS command.

The UV_FIELDS command allows to convert the Field IDs column of the UV table into Phase Offsets columns, and insert the proper coordinates. Unit specifies the unit of the RaVar and DecVar variables. It can be DEGREE, MINUTE, SECONDS, RADIAN or ABSOLUTE.

If ABSOLUTE, RaVar and DecVar are assumed to character variable arrays handling the Right Ascension and Declinations of the fields expressed in sexagesimal format.

The typical use is after a DEFINE FITS A File.uvfits command, which gives the (Ra,Dec) positions of the field centers in Degrees in A%aips_su%col%raepo and A%aips_su%col%decepo variables:

```
fits File.uvfits to File.uvt
define fits A File.uvfits
define double ra dec
define char*16 cra cdec
let cra ra /sexag d H
let cdec dec /sexag d d
compute ra mean a%aips_su%col%raepo
compute dec mean a%aips_su%col%decepo
!
uv_fields a%aips_su%col%raepo a%aips_su%col%decepo DEGREE -
  /FILE File.uvt /CENTER 'CRa' 'CDec'
!
```

This command is used in the fits_to_uvt.ima script. Note that a proper use also requires to set the reference coordinate to Ra and Dec consistently in the UV Table.

18.28.1 UV_FIELDS /CENTER

```
[CLEAN\]UV_FIELDS RaVar DecVar [Unit] /CENTER Ra Dec /FILE File
```

Specify the reference center (Pointing and Phase) of the Mosaic UV table with Fields ID. In this case, RaVar and DecVar are true Ra and Dec (angles in the specified Unit), not Offsets. The Offsets are computed from the projection information in the UV table.

Ra and Dec should be in usual sexagesimal notation.

Without /CENTER, RaVar and DecVar are Offsets from the Phase and Pointing center of the Mosaic, as given in the UV data header, unless Unit is ABSOLUTE.

18.28.2 UV_FIELDS /FILE

```
[CLEAN\]UV_FIELDS RaVar DecVar [Unit] [/CENTER Ra Dec] /FILE File
```

Modify the specified File.uvt handling the UV table.

If the option is not present, it would instead modify the current UV data in IMAGER. However, the READ command prevents to load UV tables that require the action of the UV_FIELDS command, so in practice, the /FILE option becomes mandatory.

18.29 UV_FILTER

```
[CLEAN\]UV_FILTER [/RESET] [/ZERO] [/CHANNELS Channel_List]
[/FREQUENCY List Of Frequencies] [/RANGE Min Max [TYPE]]
[/VELOCITY List of Velocities] [/WIDTH Width [TYPE]]
[/FILE FileIn [FileOut]]
```

"Filter" line emission, by flagging the corresponding channels. The channels can be specified either by the /FREQUENCY or /VELOCITY options in combination with the /WIDTH option, or by a channel list with /CHANNELS or a range (of channels, frequencies or velocities) with /RANGE.

By default, channels to be filtered are flagged (weights becoming negative). Option /ZERO can be used to erase them: weights and visibilities are set to zero, see HELP UV_FILTER /ZERO. On the contrary, option /RESET can be used to unflag them (and is thus conflicting with option /ZERO).

The command works on the current UV data, unless the option /FILE is specified.

With no options (except possibly /FILE and /ZERO), the command behaves as UV_FILTER /CHANNELS PreviewChannels. See UV_PREVIEW for details.

18.29.1 UV_FILTER /CHANNELS

```
[CLEAN\]UV_FILTER [/RESET] [/ZERO] /CHANNELS Channel_List
```

Channel_List must be a 1-D SIC variable containing the list of channels to filter out.

18.29.2 UV_FILTER /FILE

```
[CLEAN\]UV_FILTER [/ZERO] [/CHANNELS Channel_List]
[/FREQUENCY List Of Frequencies] [/RANGE Min Max [TYPE]]
[/VELOCITY List of Velocities] [/WIDTH Width [TYPE]]
/FILE FileIn [FileOut]
```

Use the UV data in the file FileIn, and write the line-free visibilities in the file FileOut. If FileOut is not specified, "-cont" is appended to the name indicated by FileIn.

This option is conflicting with the /RESET option.

18.29.3 UV_FILTER /FREQUENCY

[CLEAN\]UV_FILTER [/RESET] [/ZERO] /FREQUENCY F1 [... [Fn]] [/WIDTH Width]

Specify around which frequencies the line emission should be filtered. Frequencies F1 to Fn must be in MHz. The full width of the filtering window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

Tip: it can be convenient to have a list of SIC variables containing the frequencies of the most intense spectral lines, e.g.

```
HCO10 = 89188.52
```

18.29.4 UV_FILTER /RANGE

[CLEAN\]UV_FILTER [/RESET] [/ZERO] /RANGE Min Max [TYPE]

Indicate that channels between the First and Last defined by Min Max and Type contain line emission and should be filtered out. Type can be CHANNEL, VELOCITY or FREQUENCY.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies loading all the channels.

18.29.5 UV_FILTER /RESET

[CLEAN\]UV_FILTER /RESET [/CHANNELS Channel_List]
[/FREQUENCY List Of Frequencies] [/RANGE Min Max [TYPE]]
[/VELOCITY List of Velocities] [/WIDTH Width [TYPE]]

Unflag the channels matching the selection criteria in the options. If no option is present, unflag channels in the PreviewChannels list, or all channels if that list does not exist.

The /RESET option is conflicting with /ZERO, and not valid in the /FILE mode either.

18.29.6 UV_FILTER /VELOCITY

[CLEAN\]UV_FILTER [/RESET] /VELOCITY V1 [... [Vn]] [/WIDTH Width [TYPE]]

Specify around which velocities the line emission should be filtered. Velocities V1 to Vn must be in km/s. The full width of the filtering window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

18.29.7 UV_FILTER /WIDTH

[CLEAN\]UV_FILTER [/RESET] [/ZERO] /FREQUENCY F1 [... [Fn]] /WIDTH Width [TYPE]

[CLEAN\]UV_FILTER [/RESET] [/ZERO] /VELOCITY V1 [... [Vn]] /WIDTH Width [TYPE]

Specify the full width of the filtering window around every frequency given in the /FREQUENCY option or any velocity given in the /VELOCITY option. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

18.29.8 UV_FILTER /ZERO

```
[CLEAN\]UV_FILTER [/RESET] /ZERO [/CHANNELS Channel_List]
[/FREQUENCY F1 [... [Fn]] [/VELOCITY V1 [... [Vn]]
[/RANGE Min Max [TYPE]] [/WIDTH Width [TYPE]]
```

Erase filtered channels (set weight and visibilities to zero) rather than simply flagging them (weight set to negative value). This can be more convenient for further display, but is not reversible.

By default, channels to be filtered are flagged (weights becoming negative, and data can be unflagged by UV_FLAG). Flagged channels are ignored in the averaging process, such as UV_RESAMPLE or UV_CONT. However, as UV_MAP (in general) uses only one channel to define the weights for all others, these flagged channels will nevertheless appear in the imaged data cube.

See UV_CHECK for information about handling flagged channels and different beams in a single UV table.

18.30 UV_FLAG

[CLEAN\]UV_FLAG [/ANTENNA Iant] [/DATE ...] [/FILE FileIn [FileOut]]

```
  [/RESET]
```

Without the /FILE option, display the current UV data and calls the cursor to interactively select a region where UV data will be flagged or unflagged (sign of weight is made negative or positive). The /RESET option is used to unflag the data.

UV data flagged using command UV_FLAG can be saved on file with command WRITE UV File. Detailed information about the display control of the UV data may be found in the SHOW UV help.

See HELP UV_FLAG /FILE for direct operation on a UV data file.

Subsequent mapping with UV_MAP will ignore flagged data. However, for multi-channel imaging, the weight column is (by default) taken from one channel, so that channel-based flagging is will not be recognized in the default mode, but only in the "One Beam per Channel" mode.

18.30.1 UV_FLAG /ANTENNA

```
  [CLEAN\]UV_FLAG /ANTENNA Iant [/DATE ...] [/FILE FileIn [FileOut]]
[/RESET]
```

Restrict the flagging/unflagging to the specified antenna. Only one antenna at a time can be specified so far. If /ANTENNA is not present, the action concerns all antennas.

18.30.2 UV_FLAG /DATE

```
[CLEAN\]UV_FLAG [/ANTENNA Iant] /DATE Date  [/RESET]
```

Restrict the interactive flagging/unflagging to the specified Date.

```
[CLEAN\]UV_FLAG /DATE StartDate [StartUT [EndDate [EndUT]]]
    /FILE FileIn [FileOut]] [/ANTENNA Iant] [/RESET]
```

With the /FILE option, the /DATE option has a more flexible syntax. It understands any of the following

```
  /DATE StartDate                          The whole date is flagged
               from UT=0 to UT=24 h
  /DATE StartDate EndDate                  The whole range of dates
               from StartDate to EndDate is flagged
  /DATE StartDate StartUT EndUT            Time range from StartUT
               to EndUT in StartDate is flagged
  /DATE StartDate StartUT EndDare EndUT    Time range from StartUT
               in StartDate to EndUT in EndDate is flagged
```

Date format is DD-MMM-YYYY, and Time is in hours, sexagesimal notation (hh:mm:ss.ss) allowed.

18.30.3 UV_FLAG /FILE

[CLEAN\]UV_FLAG /FILE FileIn [FileOut] [/ANTENNA Iant] [/DATE ...] [/RESET]

Flag data found in FileIn, and store the result in FileOut, or in place if FileOut is not specified. Note that the UV_FLAG command has no memory: it caannot restore the flagging to a previous state, but only flag or unflag from the current flag state.

18.30.4 UV_FLAG /RESET

[CLEAN\]UV_FLAG /RESET [/FILE FileIn [FileOut]] [/ANTENNA Iant] [/DATE ...]

Unflag the specified range, instead for flagging it. Note that the UV_FLAG command has no memory: it cannot restore the flagging to a previous state, but only flag or unflag from the current flag state.

18.30.5 UV_FLAG CHANNEL

Channel range to be flagged/unflagged.

18.31 UV_MAP

[CLEAN\]UV_MAP [CenterX CenterY Unit] [ANGLE AngleValue] [/FIELDS FieldList] [/TRUNCATE Percent] [/RANGE Min Max Type] [/CONT [Naver]] [/INDEX Alpha]

Compute a dirty map and beam from a UV data. UV data must have been loaded from a UV table by command "READ UV File". UV_MAP processes single fields as well as Mosaics.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is possible, a sensible value of each parameter will automatically be computed from the context if the value of the corresponding variable is set to its default value, i.e. zero value and empty string. A few variables are initialized to "reasonable" values.

The arguments define the Map center and orientation (Projection system). Unit can be any of DEGREE, MINUTE, SECOND and RADIAN, or ABSOLUTE. In the latter case, CenterX should be the Right Ascension and CenterY the Declination in sexagesimal notation. Otherwise, they are offsets from the current projection center in the specified Unit.

AngleValue is the ** absolute ** rotation of the projection (counted East from North as usual), which means the angle between the Celestial North and the Y axis in the resulting image.

If no argument is given, UV_MAP uses the variable MAP_CENTER to derive the projection system. If it is empty, it does not change it.

```
    [CLEAN\]UV_MAP ?
Will list all MAP_* variables controlling the UV_MAP parameters
```

18.31.1 UV_MAP Mosaics

```
    [CLEAN\]UV_MAP [[CenterX CenterY Unit] [ANGLE AngleValue]
  /FIELDS FieldList [/TRUNCATE Percent]
```

The UV data can be a Mosaic UV table. In this case, UV_MAP will image the Mosaic, using appropriate primary beam size and truncation level.

By default, the primary beam size is taken from the telescope parameters, either as found in the telescope section of the UV table and the observing frequency (e.g. for ALMA it uses 1.13 Lambda/D). If absent, the telescope section information can be added by command SPECIFY TELESCOPE.

The truncation level for individual pointings is taken from variable MAP_TRUNCATE or from the /TRUNCATE option argument. It should be 0 for high dynamic range. Non zero values may be used with caution in case of large pointing errors.

18.31.2 UV_MAP /CONT

```
    [CLEAN\]UV_MAP [CenterX CenterY Unit] [ANGLE AngleValue] /CONT
[Naver] [/INDEX Alpha] [/RANGE Min Max Type]
```

Produce a Continuum image (from all channels, or those selected by the specified Velocity, Frequency or Channel range (depending on Type) between Min Max when the option /RANGE is present), using a Multi-Frequency Synthesis, where (u,v) coordinates are scaled with Frequencies to produce an optimal beam.

Naver is the number of initial channels that will share the same (u,v) coordinates. If not present, it is computed from MAP_FIELD and MAP_CELL to limit bandwidth smearing.

In essence, UV_MAP /CONT behaves as the combination of UV_CONTINUUM followed by UV_MAP, but the pseudo-continuum UV Table created by UV_CONTINUUM is not kept by UV_MAP /CONT.

A spectral index can be specified using option /INDEX.

18.31.3 UV_MAP /FIELDS

```
[CLEAN\]UV_MAP [CenterX CenterY Unit] [ANGLE AngleValue]
/FIELDS FieldList [/TRUNCATE Percent]
```

For a Mosaic, only image the fields specified in a 1-D Integer variable FieldList, or in the explicit list given as argument to the /FIELDS option. SHOW FIELDS will highlight the list of fields selected by this command.

18.31.4 UV_MAP /INDEX

```
[CLEAN\]UV_MAP [CenterX CenterY Unit] [ANGLE AngleValue] /CONT
[Naver] /INDEX Alpha [/RANGE Min Max Type]
```

Specify the spectral index to be used for Continuum imaging using a Multi-Frequency Synthesis method. This option is only valid with the /CONT option.

18.31.5 UV_MAP /RANGE

```
[CLEAN\]UV_MAP [CenterX CenterY Unit] [ANGLE AngleValue]
/RANGE Min Max Type [/FIELDS FieldList] [/TRUNCATE Percent]
[/CONT [Naver]] [/INDEX Alpha]
```

Indicate that only channels between the First and Last defined by Min Max and Type should be considered to produce the image. Type can be CHANNEL, VELOCITY or FREQUENCY.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies all the channels.

This is a more convenient replacement of the deprecated MCOL variable.

18.31.6 UV_MAP /TRUNCATE

```
[CLEAN\]UV_MAP [CenterX CenterY Unit] [ANGLE AngleValue]
/FIELDS FieldList /TRUNCATE Percent
```

For a Mosaic, truncate the individual pointings to the specified level (in percent). The default is to use MAP_TRUNCATE.

18.31.7 UV_MAP Variables:

```
[CLEAN\]UV_MAP ?
Will list all MAP_* variables controlling the UV_MAP parameters.
```

The list of control variables is (by alphabetic order, with the corresponding old names used by Mapping on the right)

```
New names       [   unit]       -- Description --    % Old Name
BEAM_STEP       [       ]  Channels per dirty beam  % MAP_BEAM_STEP
MAP_CELL        [ arcsec]  Image pixel size
MAP_CENTER      [ string]  RA, Dec of map center, and Position Angle
MAP_CONVOLUTION [       ]  Convolution function     % CONVOLUTION
MAP_FIELD       [ arcsec]  Map field of view
MAP_POWER       [       ]  Maximum exponent of 3 and 5 allowed in MAP_SIZE
MAP_PRECIS      [       ]  Fraction of pixel tolerance on beam matching
MAP_ROBUST      [       ]  Robustness factor        % UV_CELL[2]
MAP_ROUNDING    [       ]  Precision of MAP_SIZE
MAP_SIZE        [       ]  Number of pixels
MAP_TAPEREXPO   [       ]  Taper exponent           % TAPER_EXPO
MAP_TRUNCATE    [      %]  Mosaic truncation level
MAP_UVCELL      [      m]  UV cell size             % UV_CELL[1]
MAP_UVTAPER     [m,m,deg]  Gaussian taper           % UV_TAPER
MAP_VERSION     [       ]  Code version (0 new, -1 old)
```

See HELP UV_MAP Old_names: for deprecated variable names.

18.31.8 BEAM_STEP

```
BEAM_STEP   Integer
```

Number of channels per synthesized beam plane.

Default is 0, meaning only 1 beam plane for all channels. N (>0) indicates N consecutive channels will share the same dirty beam.

A value of -1 can be used to compute the number of channels per beam plane that ensures that the angular scale does not deviate more than a fraction of the map cell at the map edge. This fraction is controlled by variable MAP_PRECIS (default 0.1)

18.31.9 MAP_CELL

MAP_CELL[2] Real

The map pixel size [arcsec]. It is recommended to use identical values in X and Y. A sampling of at least 5 pixel per beam is recommended to ease the deconvolution. Enter 0,0 to let the task find the best values.

18.31.10 MAP_CENTER

MAP_CENTER Character String

Specify the Map center and orientation in the same way as the arguments of UV_MAP.

```
  LET MAP_CENTER "ANGLE AngleValue"   ! Rotation angle
  LET MAP_CENTER "Ax Ay Unit"         ! Center position
```

where Unit is any of DEGREE, MINUTE, SECOND, RADIAN or ABSOLUTE. These two possibilities can be combined in any order:

```
  LET MAP_CENTER "ANGLE AngleValue Ax Ay Unit"
  LET MAP_CENTER "Ax Ay Unit ANGLE AngleValue"
```

In the latter case, the keyword ANGLE can be omitted.

For Unit ABSOLUTE, Ax Ay should be the RA and Dec in standard sexagesimal notation. As usual in SIC, keywords can be abbreviated provided they are not ambiguous.

18.31.11 MAP_CONVOLUTION

MAP_CONVOLUTION Integer

Select the desired convolution function for gridding in the UV plane Choices are

```
0   Default (currently 5)
1   Boxcar
2   Gaussian
3   Sin(x)/x
4   Gaussian * Sin(x)/x
5   Spheroidal
```

Spheroidal functions is the optimal choice. So we strongly discourage use of any other convolution function, which are here for tests only.

18.31.12 MAP_FIELD

MAP_FIELD[2] Real

Field of view in X and Y in arcsec. The field of view MAP_FIELD has precedence over the number of pixels MAP_SIZE to define the actual map size when both are non-zero.

18.31.13 MAP_POWER

MAP_POWER[2] Integer

Maximum exponent of 3 and 5 allowed in automatic guess of MAP_SIZE. MAP_SIZE is decomposed in 2^k 3^p 5^q, and p and q must be less or equal to MAP_POWER.

Default is 0: MAP_SIZE is just a power of 2. A value of 1 allows approximation of any map size to 20 %, while a value of 2 allows 10 % approximation. Fast Fourier Transform are slightly slower with powers of 3 and 5, but limiting the map size can gain a lot in the Cleaning process (which can scale as MAP_SIZE^4).

18.31.14 MAP_PRECIS

MAP_PRECIS Real

Maximum mismatch in pixel at map edge between the true synthesized beam (which would have been computed using the exact channel frequency) and the computed synthesized beam with the mean frequency of the channels sharing the same beam. This is used (with the actual image size) to derive the actual number of channels that can share the same beam, i.e. the effective value of BEAM_STEP when BEAM_STEP is -1.

Default is 0.1

18.31.15 MAP_ROBUST

MAP_ROBUST Real

Robust weighting factor. A number between 0 and +infty.

Robust weighting gives the natural weight to UV cells whose natural weight is lower than a given threshold. In contrast, if the natural weight of the UV cell is larger than this threshold, the weight is set to this (uniform) threshold. The UV cell size is defined by MAP_UVCELL and the threshold value is in MAP_ROBUST.

0 means natural weighting, which is optimal for point sources. The Robust weighting factor controls the resolution: better resolution is obtained for small values (at the expense of noise), resolution approaching the natural weighting scheme for large values. Larger UV cell size give higher angular resolution (but again more noise).

MAP_ROBUST around .5 to 1 is a good compromise between noise increase and angular resolution.

18.31.16 MAP_ROUNDING

MAP_ROUNDING Real

Maximum error which can be tolerated between optimal size (given by MAP_FIELD/MAP_CELL) and its rounded value MAP_SIZE (as a power of 2^k 3^p 5^q) when rounding by floor (thus limiting the field of view). If the rounding error becomes larger, rounding is made by ceiling, which guarantees a larger field of view, but leads to bigger images.

Default is 0.05.

18.31.17 MAP_SIZE

MAP_SIZE[2] Integer

Number of pixels in X and Y. It should preferentially be a power of two, (although this is not strictly required) to speed-up the FFT. MAP_SIZE*MAP_CELL should be at least twice the size of the field-of-view (primary beam size for a single field). Enter 0,0 to let the command find a sensible map size.

MAP_SIZE is not used if MAP_FIELD is non zero.

Odd values are forbidden.

Default is 0,0, i.e. UV_MAP will guess the most appropriate values which depend on MAP_ROUNDING and MAP_POWER.

18.31.18 MAP_TAPEREXPO

MAP_TAPEREXPO Real

Taper exponent. The default is 2 (indicating a Gaussian) but smoother or sharper functions can be used. 1 would give an Exponential, 4 would be getting close to square profile...

18.31.19 MAP_TRUNCATE

MAP_TRUNCATE Real

Individual beam truncation level in PerCent. Default value is 0. Current value can be overriden by option /TRUNCATE in commands UV_MAP.

18.31.20 MAP_UVTAPER

MAP_UVTAPER[3] Real

Parameters of the tapering function (Gaussian if MAP_TAPEREXPO = 2): major axis at 1/e level [m], minor axis at 1/e level [m], and position angle [deg].

18.31.21 MAP_UVCELL

MAP_UVCELL Real

UV cell size for robust weighting [m]. Should be of the order of half the dish diameter (7.5 m for NOEMA), or smaller or even larger. It controls the beam shape in Robust weighting.

18.31.22 MAP_VERSION

MAP_VERSION Integer

[EXPERT Only] Code indicating which version of the UV_MAP and UV_RESTORE algorithm should be used. 0 is optimal. -1 is the "historical" (pre-2016) version. 1 is an intermediate version used during multi-frequency beams development.

18.31.23 Old_Names:

NAME is no longer used, and WEIGHT_MODE is obsolete.

```
MAP_RA          [  hours]  RA of map center
MAP_DEC         [    deg]  Dec of map center
MAP_ANGLE       [    deg]  Map position angle
MAP_SHIFT       [Yes/No ]  Shift phase center
```

are obsolete, superseded by MAP_CENTER. They are provided only for compatibility with older scripts.

MAP_BEAM_STEP is replaced by BEAM_STEP. An alias is provided for compatibility reasons.

WCOL (the Weight channel) and MCOL[2] (the channel range) are obsolete also. WCOL has no meaning when more than 1 beam must be produced for all channels, and should be set to 0. MCOL is superseded by the /RANGE option facility.

```
 NAME        [       ]  Label of the dirty image and beam plots
 UV_TAPER    [m,m,deg]  UV-apodization by convolution with a Gaussian
 WEIGHT_MODE [       ]  Weighting mode (NA|UN)
 UV_CELL     [m, ??  ]  UV cell size and threshold for Robust weighting
 MAP_FIELD   [ arcsec]  Map field of view
 MAP_CELL    [ arcsec]  Map cell size
 MAP_SIZE    [ pixels]  Map size in pixels (if MAP_FIELD is zero)
 MCOL        [       ]  First and Last channel to map
 WCOL        [       ]  Channel from which the weights are taken
 CONVOLUTION [       ]  Convolution function (5)
 UV_SHIFT    [       ]  Change the map phase center or map orientation?
 MAP_RA      [       ]  RA of map phase center
 MAP_DEC     [       ]  Dec of map phase center
 MAP_ANGLE   [    deg]  Map position angle
 MAP_BEAM_STEP   [       ]  Number of channels per synthesized beam plane
```

18.31.24 convolution

Older variable name for MAP_CONVOLUTION

18.31.25 map_angle

MAP_ANGLE Real

Position Angle of the direction which will become the apparent North in the map. Used only if MAP_SHIFT is YES.

Superseded by MAP_CENTER.

18.31.26 map_beam_step

MAP_BEAM_STEP Integher

Older variable name for BEAM_STEP

18.31.27 map_dec

MAP_DEC Real

Dec of map center. Used only if MAP_SHIFT is YES.

Superseded by MAP_CENTER.

18.31.28 map_ra

MAP_RA Real

RA of map center. Used only if MAP_SHIFT is YES.

Superseded by MAP_CENTER.

18.31.29 mcol

mcol[2] Integer

[Deprecated] First and Last channel to image. Values of 0 mean imaging all the planes. See UV_MAP /RANGE for a more flexible way to specify the channel range.

18.31.30 uv_cell

Older variables for MAP_UVCELL (uv

18.31.31 uv_shift

Older variable name of MAP_SHIFT (this one is also obsolescent)

18.31.32 uv_taper

Older variable name of MAP_UVTAPER

18.31.33 taper_expo

Older variable name for MAP_TAPEREXPO

18.31.34 wcol

WCOL Integer

[Obsolescent] The channel from which the weight should be taken. WCOL set to 0 means using a default channel. WCOL has no real meaning in all cases where more than one beam is computed for all channels.

18.31.35 weight_mode

weightde Character

[Deprecated] Weighting mode: Natural (optimum in terms of sensitivity) or robust (usually lower sidelobes and higher spatial resolution) weighting. This was needed in Mapping to toggle between Natural and Robust weighting, while IMAGER does that based on MAP_ROBUST value.

18.32 UV_RESAMPLE

[CLEAN\]UV_RESAMPLE [Nc Ref Val Inc] [/FILE FileIn FileOut] [/LIKE Mold]

With no option, resample the UV data loaded by READ UV on a different velocity scale. All other UV commands except UV_COMPRESS work on the "Resampled" UV table. The output spectral sampling is defined by

Nc new number of channels
Ref New reference pixel
Val New velocity at reference pixel
Inc Velocity increment

Any argument can be set to * for an automatic determination based on the values of the other arguments. The automatic determination of NC preserves the velocity coverage.

The "Resampled" UV table is a simple copy of the original one after a READ UV command, or after a UV_RESAMPLE or UV_COMPRESS command without arguments.

With the /FILE option, the internal buffers are not used: the UV data is read from file FileIn and the resampled output is written in file FileOut. In this case, the /LIKE option can be used to define the spectral sampling as identical to the Mold file.

18.32.1 UV_RESAMPLE /FILE

[CLEAN\]UV_RESAMPLE [Nc Ref Val Inc] /FILE FileIn FileOut [/LIKE Mold]

Use the UV data in the file FileIn, and write the spectrally resampled UV data in the file FileOut.

In this case, the /LIKE option can be used to define the spectral sampling as identical to the Mold file, instead of specifying it as arguments to the command.

18.32.2 UV_RESAMPLE /LIKE

[CLEAN\]UV_RESAMPLE /FILE FileIn FileOut /LIKE Mold

Use the UV data in the file FileIn, and write the spectrally resampled UV data in the file FileOut. The resampled FileOut will spectrally conform the Mold UV table specified in the /LIKE argument.

18.33 UV_RESIDUAL

```
[CLEAN\]UV_RESIDUAL [MODE] [Arg1 ... ArgN] [/FIELDS List]
```

This command subtract Clean components OR fit results from the last UV_FIT command, depending on specified argument MODE if present (can be CLEAN or UV_FIT), or on whether CLEAN (or its specialized versions HOGBOM, CLARK, etc...) or UV_FIT was done last if not. The residual UV data are available in SIC structure UV_RESIDUAL, and can be written by WRITE UV_RESIDUAL command.

The UV_RESIDUAL data can also be selected for further imaging by UV_MAP (and thus UV_RESTORE) or displayed by SHOW UV using command UV_SELECT. MODEL and UV_RESIDUAL are complementary commands (UV_DATA = UV_MODEL + UV_RESIDUAL); see HELP UV_RESIDUAL Note however.

For the Clean component the syntax is

```
[CLEAN\]UV_RESIDUAL [CLEAN] [Niter]
```

that subtracts the Niter first (default all) Clean Components from the UV data.

For the UV_FIT results, Clean component the syntax is

```
[CLEAN\]UV_RESIDUAL [UV_FIT] [F1 .. Fn]
```

that subtracts the F1 to Fn fitted functions (default all) from the UV data.

18.33.1 UV_RESIDUAL Note

UV_RESIDUAL does not yet have the /MINVAL option of command MODEL.

18.34 UV_RESTORE

```
[CLEAN\]UV_RESTORE [/COPY] [/SPEED Mode]
```

Create a Clean image from the current UV data set and the Clean Component list. The Clean Components are subtracted from the UV data set, and these residuals are gridded and Fourier transformed to compute the Residual image. This Residual image is added to the Gaussian beam convolved image of the sum of Clean components. The results are similar to those of MX, since only the residual are aliased.

This command can be used with all methods (HOGBOM, CLARK, MULTI, SDI) except for MRC which has no notion of Clean Components.

18.34.1 UV_RESTORE /COPY

```
[CLEAN\]UV_RESTORE /COPY [/SPEED Mode]
```

--- Obsolescent --- Reserved for debugging ---

The /COPY option controls whether array copys or pointers are used for some internal buffers. The option may be removed without further notice.

18.34.2 UV_RESTORE /SPEED

```
[CLEAN\]UV_RESTORE /SPEED Mode [/COPY]
```

Create a Clean image from the current UV data set and the Clean Component list, using the specified Mode to remove the Clean Components before imaging the residuals.

The argument Mode may be one of the following keyword:

AUTO Automatically select the "fastest" method, according to the current number of Visibilities and Clean components.

FFT Subtract Clean Components by FFT and interpolation onto individual visibilities.

PRECIS Subtract the Clean Components by simple Sin/Cos phase terms for each visibility

The PRECIS mode is faster for a small number of visibilities and Clean components, while the FFT mode is faster for large ones.

Instead of the above keywords, Mode may also be a number which specifies when the AUTO mode should switch from PRECIS mode to FFT mode. The default number to control the AUTO mode is 200 Millions (product of the number of visibilities and average number of Clean components per channel). This value is "sticky": if changed by /SPEED, the new values is retained until next change.

18.35 UV_REWEIGHT

```
[CLEAN\]UV_REWEIGHT MODE [Args...] [/RANGE Min Max Type]
[/FLAG Threshold]
```

Compute and/or Apply a scale factor to the weights of the current UV data. Can be used to patch e.g. JVLA data files which may have only relative weights, not absolute values indicating the noise.

MODE can be APPLY, DO, ESTIMATE or TIME.

18.35.1 UV_REWEIGHT APPLY

```
[CLEAN\]UV_REWEIGHT APPLY Factor
```

Apply the specified scale Factor to the weights. Factor can be * to used the last derived scale factor from command UV_REWEIGHT ESTIMATE. This factor is set to 1.0 by command UV_REWEIGHT APPLY, so that repetitive use of the APPLY * mode no longer affects the data.

18.35.2 UV_REWEIGHT DO

```
[CLEAN\]UV_REWEIGHT DO [Tolerance [Printout]] [/RANGE Min Max Type]
[/FLAG Threshold]
```

Derive the scale factor from the noise statistics over the channels, and use it to rescale the weights. The noise statistic is derived from the real and imaginary parts separately, taking mean value, or the minimum of the twos when the mean/min ratio is smaller than specified Tolerance (default is 1.2).

The /FLAG option indicates whether data with highly deviating weights should be flagged. UV_REWEIGHT DO combines the two actions of UV_REWEIGHT ESTIMATE (or TIME) and UV_REWEIGHT APPLY, but with a control over these outliers (that are not identified by APPLY).

Mode DO uses the ESTIMATE method if the number of channels is > 16, the TIME method if not.

Printout is a number indicating to print some correction factors every Printout visibility (no print if 0, the default) (for ESTIMATE method only).

18.35.3 UV_REWEIGHT ESTIMATE

```
[CLEAN\]UV_REWEIGHT ESTIMATE [Tolerance [Printout]] [/RANGE Min Max
Type]
```

Derive the scale factor from the noise statistics derived from the rms individual visibilities over the specified channels. The noise statistic is derived from the real and imaginary parts separately, taking mean value, or the minimum of the twos when the mean/min ratio is smaller than specified Tolerance (default is 1.2).

Printout is a number indicating to print some correction factors every Printout visibility (no print if 0, the default).

18.35.4 UV_REWEIGHT TIME

```
[CLEAN\]UV_REWEIGHT TIME [Tolerance] [/RANGE Min Max Type]
```

Derive the scale factor from the noise statistics derived from consecutive visibilities, considering the specified channel range. The noise statistic is derived from the real and imaginary parts separately, taking mean value, or the minimum of the twos when the mean/min ratio is smaller than specified Tolerance (default is 1.2).

The UV data is first baseline-time sorted to ensure that all UV data points of the UV track for any antenna pair are consecutive.

18.35.5 UV_REWEIGHT /RANGE

```
[CLEAN\]UV_REWEIGHT DO|ESTIMATE|TIME [Tolerance [Printout]] [/FLAG
Threshold]
  /RANGE Min Max [Min Max [...]] Type
```

Specify the spectral range(s) defining the channels to use in the noise estimates. More than 1 range can be specified if the Mode is not TIME.

/RANGE can not be used with keyword APPLY.

18.35.6 UV_REWEIGHT /FILE

```
[CLEAN\]UV_REWEIGHT DO|ESTIMATE|TIME [Tolerance [Printout]] [/FLAG
Threshold]
  /RANGE Min Max [Min Max [...]] Type
```

18.35.7 UV_REWEIGHT /FLAG

```
[CLEAN\]UV_REWEIGHT DO [Tolerance [Printout]] /FLAG Threshold
[/RANGE Min Max Type]
```

The /FLAG option indicates whether data with highly deviating weights should be flagged. UV_REWEIGHT DO combines the two actions of UV_REWEIGHT ESTIMATE and UV_REWEIGHT APPLY, but with a control over these outliers (that are not identified by APPLY).

Default Threshold is 3, i.e. data are considered as outliers if their re-scale factor is more than 3 times above (or below) the median value. This is adequate to spot bad data.

/FLAG is only valid for keyword DO.

18.36 UV_SHIFT

[CLEAN\]UV_SHIFT [CenterX CenterY UNIT] [ANGLE Angle] [/FILE FileInOut]

Shift the current (if no /FILE option is present) or specified (if the /FILE option is here) UV table (single field or mosaic) to a common phase center and/or rotate it.

Angle is desired position angle of the Celestial North from the Y axis (counted positive towards East, i.e. counter-clockwise in usual astronomical images), in Degree. It is thus an absolute value.

If UNIT is ABSOLUTE, CenterX and CenterY are absolute coordinates in usual sexagesimal notation.

If not, CenterX and CenterY are offsets in the specified angular unit (SECOND, MINUTE, DEGREE or RADIAN).

Although UV_MAP also provides a direct possibility to re-center the image on a specified projection (phase) center cmobined with a rotation of the main axes, the modified visibilities cannot be saved on file. It is sometimes required to do this for specific use. UV_SHIFT provides this possibility, and the shifted UV table can be written using command WRITE UV.

UV_DEPROJECT includes the UV_SHIFT capabilities, but also has additional functions, and a different syntax.

18.36.1 UV_SHIFT /FILE

[CLEAN\]UV_SHIFT [CenterX CenterY UNIT] [ANGLE Angle] /FILE FileInOut

Shift and/or rotate the UV table specified in the /FILE option (single field or mosaic) to a common phase center and given orientation.

Warning: Rotated Mosaics cannot yet be saved. IRAM has not decided upon a convention for storing this information.

18.37 UV_SMOOTH

[CLEAN\]UV_SMOOTH Nc [/ASYMMETRIC] [/FILE FileIn FileOut]

Smooth by NC channels a UV data set. UV_SMOOTH differs from UV_COMPRESS because it keeps the original spectral sampling. The resulting UV data is thus oversampled by NC channels.

Unless the /ASYMMETRIC option is present, the smoothing is symmetric, to ensure that the spectral sampling is preserved. This is implicit when Nc is odd, but for even values of Nc, it is done by smoothing by Nc+1 channels, and giving the edge channels of the smoothing kernel a half-weight. The effective bandwidth of each smoothed channel is still Nc times the original one.

As a result, UV_SMOOTH 2 is equivalent to a Hanning smoothing.

With no /FILE option, the current UV table obtained by READ UV is resampled. All further UV commands work on the "Resampled" UV table. The "Resampled" UV table is a simple copy of the original one after a READ UV command, or after a UV_RESAMPLE or UV_COMPRESS commands with no arguments and no options.

With the /FILE option, the UV data is read from file FileIn and the resampled output is written in file FileOut.

18.37.1 UV_SMOOTH /ASYMMETRIC

[CLEAN\]UV_SMOOTH Nc /ASYMMETRIC [/FILE FileIn FileOut]

Make no attempt to preserve the spectral sampling when Nc is even: the reference channel is then shifted by half the channel spacing.

For odd Nc, the smoothing is naturally symmetric, so the option has no impact.

18.37.2 UV_SMOOTH /FILE

[CLEAN\]UV_SMOOTH Nc /FILE FileIn FileOut [/ASYMMETRIC]

Read the UV data in the file FileIn, and write the data after averaging by NC adjacent channels in the file FileOut.

18.38 UV_SPLIT

[CLEAN\]UV_SPLIT [Degree [Compress]] /FILE FileIn [FileLine [File-

```
Cont]]
  [/CHANNELS Channel_List]
  [/FREQUENCY List Of Frequencies] [/RANGE Min Max [TYPE]]
  [/VELOCITY List of Velocities] [/WIDTH Width [TYPE]]
```

"Split" the UV table found in file FileIn into a Continuum-free one (FileLine) and a Line-free continuum one (FileCont), using the ranges specified by one of the other options to indicate where the line emission is (see UV_BASELINE and UV_FILTER for details). The channels can be specified either by the /FREQUENCY or /VELOCITY options in combination with the /WIDTH option, or by a channel list with /CHANNELS or a range (of channels, frequencies or velocities) with /RANGE.

Degree is the baseline fit degree (0 or at most 1). Compress is the number of line channels averaged together while producing the pure continuum (Line-free) visibilities.

The command only works with files: option /FILE is mandatory.

With only /FILE as option, the command behaves as UV_SPLIT /FILE FileIn [FileLine [FileCont]] /CHANNELS Preview%Channels. See UV_PREVIEW for details about Preview%Channels.

18.38.1 UV_SPLIT /CHANNELS

```
    [CLEAN\]UV_SPLIT [Degree [Compress]] /FILE FileIn [FileLine [File-
Cont]]
  /CHANNELS Channel_List
```

Channel_List must be a 1-D SIC variable containing the list of channels where line emission is assumed to be.

18.38.2 UV_SPLIT /FILE

```
    [CLEAN\]UV_SPLIT [Degree [Compress]] /FILE FileIn [FileLine [File-
Cont]]
  [/FREQUENCY List Of Frequencies] [/RANGE Min Max [TYPE]]
  [/VELOCITY List of Velocities] [/WIDTH Width [TYPE]]
  /FILE FileIn [FileOut]
```

Use the UV data in the file FileIn, and write the continuum-free visibilities in the file FileLine, and the line-free visibilities in FileCont. If not specified, FileLine (resp FileCont) is derived from FileIn by appending "-line" (resp "-cont") to the filename.

18.38.3 UV_SPLIT /FREQUENCY

```
    [CLEAN\]UV_SPLIT [Degree [Compress]] /FILE FileIn  [FileLine  [File-
Cont]]
    /FREQUENCY F1 [... [Fn]] [/WIDTH Width]
```

Specify around which frequencies the line emission should be found. Frequencies F1 to Fn must be in MHz. The full width of the window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

Tip: it can be convenient to have a list of SIC variables containing the frequencies of the most intense spectral lines, e.g.

```
        HCO10 = 89188.52
```

18.38.4 UV_SPLIT /RANGE

```
    [CLEAN\]UV_SPLIT  [Degree  [Compress]] /FILE FileIn [FileLine [File-
Cont]]
    /RANGE Min Max [TYPE]
```

Indicate that channels between the First and Last defined by Min Max and Type contain line emission. Type can be CHANNEL, VELOCITY or FREQUENCY.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies loading all the channels.

18.38.5 UV_SPLIT /VELOCITY

```
    [CLEAN\]UV_SPLIT [Degree [Compress]] /FILE FileIn  [FileLine  [File-
Cont]]
  /VELOCITY V1 [... [Vn]] [/WIDTH Width [TYPE]]
```

Specify around which velocities the line emission should be found. Velocities V1 to Vn must be in km/s. The full width of the window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

18.38.6 UV_SPLIT /WIDTH

```
    [CLEAN\]UV_SPLIT [Degree [Compress]] /FILE FileIn  [FileLine  [File-
Cont]]
  /FREQUENCY F1 [... [Fn]] /WIDTH Width [TYPE]

    [CLEAN\]UV_SPLIT  [Degree  [Compress]] /FILE FileIn [FileLine [File-
Cont]]
  /VELOCITY V1 [... [Vn]] /WIDTH Width [TYPE]
```

Specify the full width of the window around every frequency given in the /FREQUENCY option or any velocity given in the /VELOCITY option. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

18.39 UV_STAT

```
    [CLEAN\]UV_STAT Mode [Start [Step]]
```

UV_STAT allows the astronomer to select the best weighting and imaging parameters according to its personal trade off between angular resolution, sensitivity and field of view.

Argument Mode describes the action. Default is ALL, equivalent to HEADER+ADVISE. See HELP UV_STAT Argument for details.

18.39.1 UV_STAT Casa

CASA and GILDAS handle the weighting scheme slightly differently and with different conventions for the Robust parameter values.

CASA uses only Briggs weighting, while IMAGER offers the Briggs weighting scheme (obtained for MAP_ROBUST < 0) and a different, more progressive, weighting (obtained for MAP_ROBUST > 0).

For MAP_ROBUST < 0, the corresponding CASA Briggs value can be obtained by adding 2.

In addition, CASA apparently uses a default value for the size of the UV cell in Briggs weighting which is about the dish diameter(*). If MAP_UVCELL is 0, IMAGER will use the same default value. Otherwise, IMAGER will use the specified user input.

(*) The default value quoted before is valid for a single field. Howev-

er, the documentation is unclear in this respect and suggests that the UV cell size depends on the imaged field size. This would be strange, though. What happens for mosaics is even less clear.

18.39.2 UV_STAT Mosaics

UV_STAT handles Mosaic like Single-Field for the weighting (BEAM, BRIGGS, TAPER and WEIGHT arguments): all visibilities are considered, instead of those of just one among the various fields. Accordingly, the predicted beam sizes are often underestimated compared to what UV_MAP and FIT commands would yield.

For a fine adjustment of the beam shape and size, it is recommended to verify using UV_MAP and FIT commands and iterate if needed.

18.39.3 UV_STAT Arguments:

```
[CLEAN\]UV_STAT Mode [Start [Step]]
```

Argument Mode describes the action. Possible values are

ADVISE	Suggest values for Map size, Map field and Pixel size
ALL	As HEADER + ADVISE
BEAM	Evaluate (and optionally display) the synthesized beam
BRIGGS	Estimate beam size as function of Briggs robust parameter
CELL	Estimate beam size as function of UV cell size
DEFAULT	Reset UV_MAP parameters to Default values
HEADER	Display detailed UV header
RESET	Reset UV_MAP parameters to previous Values
SETUP	Take suggested values for Map size, Map field and Pixel size
TAPER	Estimate beam size as function of UV Taper
WEIGHT	Estimate beam size as function of Gildas robust parameter

The default value is ALL, which makes the same action as HEADER + ADVISE combined.

18.39.4 ADVISE

```
[CLEAN\]UV_STAT ADVISE
```

Display the recommended image and pixel sizes, independently of the current values of MAP_FIELD, MAP_SIZE and MAP_CELL.

18.39.5 ALL

[CLEAN\]UV_STAT ALL

Display the UV header and a few more associated values, like the recommended image and pixel sizes. Equivalent to HEADER + ADVISE

18.39.6 BEAM

[CLEAN\]UV_STAT BEAM [Show]

Evaluate the expected synthesized beam size and orientation, based on current imaging parameters MAP_UVCELL, MAP_ROBUST, MAP_UVTAPER.

If argument Show is present, the synthesized beam is also displayed. A fast gridding is used for this command to minimize computing time, so the actual synthesized beam produced by UV_MAP will be (very slightly) different. This does not significantly affect the beam size.

18.39.7 BRIGGS

[CLEAN\]UV_STAT BRIGGS

Beam sizes and noise level (in flux and brightness) will be computed for 9 different "robust" weighting parameters using the Briggs method. The Briggs parameters range from -4 (nearly uniform weighting) to 0 (natural weighting) (values are shifted by -2 compared to the Briggs parameters in CASA). The taper is taken from variable UV_TAPER. The UV cell size is specified by MAP_UVCELL in m.

See HELP UV_STAT Casa for details about Briggs parameter vs Robust parameter.

18.39.8 CELL

[CLEAN\]UV_STAT CELL [Start [Step]]

Predict the synthesized beam, expected noise level, and recommended pixel size for different values of the uv cell size for current robust weighting and taper parameters.

18.39.9 DEFAULT

[CLEAN\]UV_STAT DEFAULT

Reset MAP_SIZE, MAP_CELL, MAP_FIELD and MAP_UVCELL to their Default values (all 0). Robust parameter and Tapers are not changed. Command UV_STAT RESET does a slightly different job.

18.39.10 HEADER

[CLEAN\]UV_STAT HEADER

Display a brief summary of the UV data: number of antennas, observing dates, baseline ranges, spectroscopic information.

18.39.11 RESET

[CLEAN\]UV_STAT RESET

Reset MAP_SIZE, MAP_FIELD and MAP_CELL to their previous values after a UV_STAT SETUP. This is different from the UV_STAT DEFAULT command.

18.39.12 SETUP

[CLEAN\]UV_STAT SETUP

Compute the recommended values for the imaging: image size, pixel size, field of view, largest angular scale, etc..., and set the corresponding MAP_variables (MAP_SIZE, MAP_FIELD and MAP_CELL) to these values if they were not previously specified.

18.39.13 TAPER

[CLEAN\]UV_STAT TAPER [Start [Step]]

For TAPER, beam sizes and noise level (in flux and brightness) will be computed for 9 different tapers (from Start to Start*Step^9). Default value for Step is sqrt(2), Default value for Start is 50 m. Weighting mode, UV cell size and "robust" parameter are taken from variable MAP_UVCELL and MAP_ROBUST (i.e. one can combine Robust weighting and Tapering).

18.39.14 WEIGHT

[CLEAN\]UV_STAT WEIGHT [Start [Step]]

For WEIGHT, beam sizes and noise level (in flux and brightness) will be computed for 9 different "robust" weighting parameters (from Start to Start*Step^9). Default value for Step is sqrt(10), and default value for Start is derived to center the "robust" parameter values around 1. UV cell size is taken from variable MAP_UVCELL, and Taper is taken from variable UV_TAPER.

18.40 UV_TIME

[CLEAN]UV_TIME [?|Time] [/Weight Wcol] [/FILE FileIn]

Average in time the current UV data set to reduce the number of visibilities. Time must be in seconds. If not specified, an automatic guess is performed based on Nyquist criterium using the antenna size and baseline lengths.

With a ? as argument, return the longest reasonable time. This time depends on the field of view (that can be controlled by MAP_FIELD) and the requested precision (controlled by MAP_PRECIS). It is also limited by the UV cell size for beam forming (MAP_UVCELL).

18.40.1 UV_TIME /WEIGHT

[CLEAN]UV_TIME Time /WEIGHT Wcol

Select the weight column. Default is 0.

18.40.2 UV_TIME /FILE

[CLEAN]UV_TIME [?|Time] /FILE FileIn [FileOut] [/Weight Wcol]

Work on the specified input (FileIn) and ouput (FileOut) for the time compression instead of the UV buffer. FileIn must exists.

18.41 UV_TRIM

[CLEAN\]UV_TRIM Action [/FILE FileIn [FileOut]]

Trim the UV data of useless information, according to the Action keyword. Action can be

ANY Remove visibilities that have at least one channel flagged
FLAG Remove visibilities where all channels at flagged
TRAIL Remove trailing columns

The command works on the current UV data set, unless the /FILE option is present.

18.41.1 UV_TRIM /FILE

[CLEAN]UV_TRIM Action /FILE FileIn [FileOut]

Trim the UV data of useless information, according to the Action keyword. The UV data is taken from file FileIn and written on FileOut (default: overwrite FileIn).

18.42 UV_TRUNCATE

[CLEAN]UV_TRUNCATE Max [Min]

Truncate the UV data, by removing baselines out of the specified range Min (default 0) and Max (in meter).

18.43 WRITE

[CLEAN\]WRITE Name File [Format] [/APPEND] [/RANGE Start End Kind] [/REPLACE] [/TRIM [Any]]

WRITE the buffer specified by Name (UV, CGAINS, and BEAM, PRIMARY, DIRTY, CLEAN, RESIDUAL, SUPPORT, CCT, SKY) onto a File. Default extensions are .uvt for UV and CGAINS and .beam, .lmv, .lobe, .lmv-clean, .lmv-res, .pol, .cct, and lmv-sky respectively for the next ones. If Name does not refer to a known buffer, but to a SIC Image variable, this variable is written. The default extension is then .gdf.

Format is an optional argument indicating the output file data format. It can take the value GILDAS (the default) or FITS. FITS format is allowed for UV data and images, but not for tables like the Clean Component, or self-calibration results. Options /APPEND and /REPLACE are not useable for FITS format.

For UV data, the flagged data are written by default. They may be omitted using the /TRIM option.

WRITE * File can be used to write all modified image-like buffers (not the UV tables) under a common File name. This typically include .beam, .lmv and .lmv-clean, but also the .lmv-sky file if the PRIMARY command has been used after deconvolution.

Options /APPEND and /REPLACE are mutually exclusive, and in general used with the /RANGE option.

18.43.1 WRITE Format

[CLEAN\]WRITE Name File FITS|GILDAS [/RANGE Start End Kind] [/TRIM [Any]]

The 3rd argument of the WRITE command is an optional keyword indicating the output file data format. It can take the value GILDAS (the default) or FITS.

FITS format is allowed for UV data and Images (n-D hypercubes), but not for Tables like the Clean Component Tables, or self-calibration results.

Options /APPEND and /REPLACE are not useable for FITS format.

18.43.2 WRITE /RANGE

[CLEAN\]WRITE Buffer File [Format] /RANGE Start End Kind [/APPEND] [/REPLACE]

A range of planes can be specified through /RANGE option. Kind is the unit of the Start-End values: CHANNEL, VELOCITY, or FREQUENCY. This is also possible for UV data, except for FITS output format.

The Buffer name must be specified (* is not valid here).

18.43.3 WRITE /APPEND

[CLEAN\]WRITE Buffer File /APPEND [/RANGE Start End Kind]

The selected channels are appended to an existing file.

This is not valid for UV tables, neither for FITS output format.

The Buffer name must be specified (* is not valid here).

18.43.4 WRITE /REPLACE

[CLEAN\]WRITE Buffer File /REPLACE [/RANGE Start End Kind]

The selected channels are replaced in an existing file.

This is not valid for UV tables, neither for FITS output format.

The Buffer name must be specified (* is not valid here).

18.43.5 WRITE /TRIM

[CLEAN\]WRITE UV File /TRIM [ANY]

Remove the flagged visibilities while writing. A visibility is declared "flagged" if all channels in it are flagged, unless the keyword ANY is present. In this later case, if a single channel is flagged, the whole visibility is declared flagged.

19 CALIBRATE Language Internal Help

19.1 Language

```
APPLY         : Apply gain solution to current UV Data
COLLECT       : Gather several self-calibration solution together
DERIVE        : Derive Baseline gains from Antenna gains
SCALE_FLUX    : Adjust flux scale on a daily basis
MODEL         : Compute a UV model from the Clean Components Table
SOLVE         : Solve for complex gains using the UV model
TRANSFORM     : Apply transformation on frequency axis
UV_SELECT     : Select UV data to be displayed, imaged or written
UV_SELF       : Build the Self Calibration UV Table and dirty image
UV_SORT       : Sort and Transpose UV data for plotting
```

19.2 APPLY

```
[CALIBRATE\]APPLY [AMPLI|DELAY|PHASE [Gain]] [/FLAG [Threshold]]
```

Apply gain solution computed by MODEL and SOLVE (which are called implicitely by SELFCAL) or obtained by READ CGAINS to the current UV data. The optional arguments indicate whether this should be an AMPLITUDE, DELAY or PHASE gain, and what Gain value is used (in range 0 to 1). A DELAY gain is derived from a Phase self-calibration assuming the Phase errors represent pathlength changes (i.e. delays): phase solution is scaled with the Frequency ratio of the self-calibration and UV data to be calibrated.

If no argument is given, the current SELF_MODE (see HELP SELFCAL) is used, and the gain is 1.0.

The /FLAG option controls whether data without a valid gain solution are kept unchanged or flagged. You can actually flag data on this basis even without applying the self-calibration solution.

19.2.1 APPLY AMPLI

```
[CALIBRATE\]APPLY AMPLI [Gain] [/FLAG [Threshold]]
```

Apply an Amplitude gain calibration result. In doing so, the weights are never up-scaled, even when the Amplitude correction factor is smaller than 1. They are down-scaled for visibilities with correction factor larger than 1. The rationale behind this behaviour is to avoid giving more weights to data which has been modified in a self-calibration process.

19.2.2 APPLY DELAY

[CALIBRATE\]APPLY DELAY [Gain] [/FLAG [Threshold]]

Apply an PathLength (geometric delay) gain calibration result. The phases are assumed to scale strictly as the frequency. This mode is in general the most appropriate, since most residual phase errors are due to the atmospheric pathlength change.

However, it may produce spurious phase jumps if some instrumental phase had remained uncalibrated.

19.2.3 APPLY PHASE

[CALIBRATE\]APPLY PHASE [Gain] [/FLAG [Threshold]]

Apply a Phase gain calibration result. Any possible frequency dependence of the correction is neglected.

19.2.4 APPLY /FLAG

[CALIBRATE\]APPLY [AMPLI|DELAY|PHASE [Gain]] /FLAG [Threshold]

Apply gain solution (in AMPLITUDE, DELAY or PHASE) and flag data without a corresponding valid gain solution.

If the Threshold argument is present, all data with a correction exceeding that threshold are also flagged. This can be useful to filter out antennas that have unstable gains. A threshold of 90 degrees in phase, or 1.3 in amplitude may be appropriate.

The optional Gain argument has no effect on the Threshold: it is always the gain value in the CGAINS array that is compared to the Threshold.

USEFUL TRICK: If you forgot to flag the data while applying the self-calibration gain solution, you can still do it a posteriori without changing the amplitude and/or phase by using command

APPLY MODE 0 /FLAG Threshold

where MODE is any of AMPLI, DELAY or PHASE.

19.3 COLLECT

[CALIBRATE\]COLLECT FileFilter MergedFile [ReferenceAntenna] [/FLAG Threshold]

Average self-calibration solutions computed from different frequencies into a single one,.

FileFilter is a character string handling a filter to define all files storing the selected antenna gain solutions (files specified by SELF_SNAME in the SELFCAL command).

MergedFile is the SNR weighted average of all selected self-calibration solutions. The averaging of phases is made in DELAY mode, i.e. assuming the self-calibration phases are actually Pathlengths (Delays) at the calibration frequency.

The reference antenna can optionally be changed for simpler display in this process if argument ReferenceAntenna is present.

19.3.1 COLLECT /FLAG

[CALIBRATE\]COLLECT FileFilter MergedFile [ReferenceAntenna] /FLAG Threshold

If more than 1 self-calibration loop is present in the selected files, the unstable solutions, defined as those for which the phase difference between the last 2 iterations exceeds Threshold sigma, are flagged.

Note that this option can be used to flag solutions even if only one frequency was used (i.e. when FileFilter refers to only one file).

19.4 DERIVE

[CALIBRATE\]DERIVE AgainsFile [CgainsFile]

Derive Baseline-based (Cgains) solution from an Antenna gain solution file. The Baseline-based solution is stored in buffer CGAINS, unless a file name is given by argument CgainsFile.

19.4.1 DERIVE Example

Commands COLLECT and DERIVE are typically used to improve wide-bandwidth self-calibration solutions. At NOEMA, a self-calibration solution can be computed independently for each of the 4 Wide band UV table. Assume these are stored in Wide-'i'-phase-sol.gdf, with i varying from 1 to 4.

```
COLLECT Wide-phase*.gdf Cont-sol.gdf
DERIVE  Cont-sol.gdf Cont-cgains.uvt
READ CGAINS Cont-cgains.uvt
FOR I 1 to 4
```

```
    READ UV Wide-'i'.uvt
    APPLY DELAY /FLAG
    WRITE UV Self-Wide-'i'.uvt
  NEXT
```

will average the (antenna-based) Phase calibration solutions (COLLECT) and compute the corresponding Baseline-based gains (DERIVE). The loop then applies this better self-calibration solution to each UV table, scaling the Phases according to Frequency.

19.5 SCALE_FLUX

```
[CALIBRATE\]SCALE_FLUX Find|Apply|List|Calibrate [Args...]
```

A set of commands to check flux calibration on a day to day basis. It gives the ratio between the observed flux (in the current UV data set) and the model flux for each separate period. The Model flux can be derived from Clean Component Tables using the MODEL command, or read from an outer file using READ MODEL.

Error bars are approximate. The User-defined command SOLVE_FLUX performs a more accurate evaluation of the error, but is typically 50 times slower.

SCALE_FLUX FIND [DateTolerance [UVmin UVmax]]
determines, by linear regression, the best scaling factor to match date by date the UV data set with the MODEL data set. Separate Periods are defined when Dates differ more than DateTolerance (default 1 day). Only data with baseline lengths in the range UVmin UVmax are considered for the regression (default all).

SCALE_FLUX APPLY VarName
apply previously determined flux scale factors to the MODEL data set, previously read by READ MODEL. This is in general used only in an iterative search way, e.g. by the user-defined command SOLVE_FLUX (which calls procedure solve_flux). The resulting UV data set is loaded into the specified VarName SIC variable.

SCALE_FLUX LIST
print out dates, baselines and determined flux factors

SCALE_FLUX CALIBRATE
apply previously determined flux scale factors to the current UV data set (i.e. divide the visibilities by the scaling factor of each date, and correct the weight accordingly). This may then be written using command WRITE UV .

SCALE_FLUX SOLVE [DateTolerance [UVmin UVmax]]
combines FIND and PRINT behaviours.

19.6 MODEL

[CALIBRATE\]MODEL [Arg1 [... ArgN] [/MINVAL Value [Unit]]
[/MODE CCT|IMAGE|UV_FIT [Frequency]]

This command creates model visibilities from Clean components, Clean images or UV_FIT results from the last UV_FIT command, depending on the /MODE option. If the option is not present, the default mode is CCT or UV_FIT, depending whether CLEAN (or its specialized versions HOGBOM, CLARK, etc...) or UV_FIT was done last.

The current UV data is used to define the visibility sampling and data weights. It should have only one channel, or the same number of channels as the specified model.

The resulting UV data is available in SIC structure UV_MODEL. It can be written using WRITE UV_MODEL, or selected for further use using command UV_SELECT. MODEL and UV_RESIDUAL are complementary commands (UV_DATA = UV_MODEL + UV_RESIDUAL).

For the Clean component the syntax is
[CALIBRATE\]MODEL [MaxIter] [/MINVAL Value [Unit]] [/MODE CCT [Frequency]]
It computes visibilities on the current UV sampling using a source model made of the MaxIter first Clean Components, or of all pixel values above the given Value if /MINVAL is present.

For the Clean image, the syntax is
[CALIBRATE\]MODEL [ImageName] [/MINVAL Value [Unit]] [/MODE IMAGE]
(ImageName is currently dummy. It will be used to specify any available 3-D Image variable later.)

For the UV_FIT results, the syntax is
[CALIBRATE\]MODEL [F1 .. Fn] [/MODE UV_FIT]
which computes a model using the F1, ..., Fn fitted functions (default all) for the current UV sampling.

UV_MODEL can also be selected by using command UV_SELECT for further imaging by UV_MAP (and thus UV_RESTORE) or display by SHOW UV.

19.6.1 MODEL /MINVAL

[CALIBRATE\]MODEL [MaxIter] /MINVAL Value [/MODE CCT [Frequency]]

For the CCT mode, construct the source model using all Clean Components until MaxIter (all if MaxIter is 0 or not specified). These components are stacked on a grid, and then all pixels above the given Value are taken as source model to derive visibilities.

[CALIBRATE\]MODEL [ImageName] /MINVAL Value [Unit] /MODE IMAGE [Frequency]

For the IMAGE mode, all pixels above the given Value are taken as source model to derive visibilities. An outer band of 1/8th of the image size is also set to 0 at map edges to avoid aliasing issues. Unit can be Jy, mJy, K or sigma. The default value is Jy.

The /MINVAL option is not valid in UV_FIT mode.

19.6.2 MODEL /MODE

[CALIBRATE\]MODEL [Arg] [/MINVAL Value [Unit]] /MODE CCT|IMAGE|UV_FIT [Frequency]

Specify which input data is used to compute the model visibilities: Clean Component Table (CCT), Clean image (IMAGE) or UV_FIT results.

By default, the UV_MODEL output UV data takes its Frequency from the input Image or CCT data set. However, this can be supersed by the optional argument Frequency, which can take the value IMAGE (the default), UV or any frequency in MHz.

If not specified, the MODE is derived from the last executed operation (CLEAN or UV_FIT)

19.7 SOLVE

[CALIBRATE\]SOLVE Time SNR [Reference]
/MODE [Phase|Amplitude] [Antenna|Baseline] [Flag|Keep]

Solve the baseline or antenna based gains using the current UV data and current MODEL.

Time is the integration time for the solution. SNR is the minimum Signal to Noise Ratio required to find a solution.

19.7.1 SOLVE /MODE

[CALIBRATE\]SOLVE Time SNR [Reference]
/MODE [Phase|Amplitude] [Antenna|Baseline] [Flag|Keep]

Depending on the /MODE arguments, the gains can be antenna-based or baseline-based, and include Phase or Amplitude, and data without solutions either KEEPed or FLAGged,

19.8 TRANSFORM

[CALIBRATE\]TRANSFORM Operation FileIn FileOut [Control]

Apply a transformation along the Frequency axis of a data cube. Currently recognized Operation values are

FFT	Compute the (complex, hermitian) Fourier Transform
WAVE	Compute a Wavelet Transform

FileIn is an Input data cube or the result of a previous transformation operation.

FileOut is the transformation result. The transformation is reversible:
TRANSFORM Oper In Out [ControlIn] followed by
TRANSFORM Oper Out In [ControlOut]
re-creates the In file provided the adequate Control[s] are properly specified.

Input Data Cubes can be in LMV (Position, Position, Velocity) or VLM format.

Output Files have a format that depends on the Operation. See HELP TRANSFORM FFT and HELP TRANSFORM WAVE for details.

19.8.1 TRANSFORM FFT

[CALIBRATE\]TRANSFORM FFT FileIn FileOut [Nchan]

Compute the Fourier Transform of FileIn along the Frequency/Velocity axis. The direction of the Fourier Transform depends on the nature of FileIn.

If FileIn is a standard Data Cube, a direct Fourier transform is performed, that leads to 4-D cube as FileOut, in order (position,position,complex,velocity). Its 3rd axis has 2 pixels that contain the Real and Imaginary parts. The number of channels is extended to the nearest integer of the form 2^n 3^p 5^q where p and q are less or equal to 2.

Conversely, if FileIn is such a 4-D cube, an inverse Fourier transform is performed, leading to a (position,position,velocity) cube as FileOut.

Optionally, Nchan can specify how many channels are actually retained in this case.

19.8.2 TRANSFORM WAVE

[CALIBRATE\]TRANSFORM WAVE FileIn FileOut Direction

Compute a Wavelet Transform of FileIn along the Frequency/Velocity axis. Direction indicates if a direct or inverse Wavelet transformation is applied.

If Direction = 0, the number of channels is extended to the nearest power of 2, and a direct wavelet transform is performed

If Direction > 0, an inverse wavelet transform is performed (that assumes a power of 2 in number of channels in FileIn) and the first Direction channels only are kept in FileOut.

Thus, for a data cube of Nc channels, TRANSFORM WAVE In Out 0 followed by TRANSFORM WAVE Out In Nc re-creates the original data.

19.9 UV_SELF

[CALIBRATE\]UV_SELF [CenterX CenterY UNIT [Angle]]
[/RANGE [Min Max Type]] [/RESTORE]

Use (and if specified and/or needed create) the "Self Calibrated" UV dataset to make a dirty image, instead of using the current UV table.

UV_SELF utilizes UV_MAP for imaging. See HELP UV_MAP for parameters.

19.9.1 UV_SELF /RANGE

[CALIBRATE\]UV_SELF [CenterX CenterY UNIT [Angle]] /RANGE [Min Max Type]

Create and image the "Self Calibrated" UV data.

The "Self Calibrated" UV dataset is created from the current UV data set by extracting the range of channels specified by the /RANGE arguments Min Max Type. Type can be CHANNEL, VELOCITY or FREQUENCY. If /RANGE has no argument, all channels are averaged together.

It is then updated by command SOLVE at each self-calibration loop. See

SOLVE and ADVANCE\SELFCAL for details.

19.9.2 UV_SELF /RESTORE

[CALIBRATE\]UV_SELF /RESTORE

As UV_RESTORE but for the self-calibrated UV table.

Restores the Clean image from the Clean Component Table by removing the components from the Self-calibrated UV data and imaging the residuals before adding them to the convolved Clean components.

See UV_RESTORE for details.

19.10 UV_SELECT

[CALIBRATE\]UV_SELECT [Key]

Select which UV data set will be used by commands UV_MAP, UV_RESTORE, SHOW UV or WRITE UV. Key can be any of following:

DATA or UV_DATA	to specify the UV data obtained by READ UV
MODEL or UV_MODEL	to specify the UV data obtained by READ MODEL, or computed by command MODEL
RESIDUAL or UV_RESIDUAL	to specify the UV data computed by command UV_FIT

No other UV related command is affected by UV_SELECT. They all work on the UV_DATA dataset.

19.11 UV_SORT

[CALIBRATE\]UV_SORT Key [/FILE FileIn FileOut]

Sort and transpose the UV data set. The command has two different behaviours, depending on the /FILE option.

Without /FILE, the command works on the current UV data, loaded by command READ UV File, and creates a transposed, ordered copy of the UV data. The Key can be TIME for Time-Baseline ordering, BASE for Baseline-Time ordering. The sorted UV data is then available in variable UVS for further plotting. This is only done in an internal buffer. WRITE UV will **NOT** write this sorted, transposed, buffer.

With the /FILE option, the command creates in FileOut a sorted (but not transposed) copy of the UV data file specified in FileIn.

19.11.1 UV_SORT /FILE

```
[CALIBRATE\]UV_SORT Key /FILE FileIn FileOut
```

Creates in Fileout a sorted copy of the UV data found in FileIn. Key can be The Key can be TIME for Time-Baseline ordering, BASE for Baseline-Time ordering, or FIELDS for Mosaics. TIME ordering is required for efficient operation of the time averaging command UV_TIME.

The command compares the file size to the available RAM memory (more specifically the size indicated by the logical name SPACE_GILDAS) to decide whether the operation can be done in Memory only or needs intermediate files to perform the sorting.

20 ADVANCED Language Internal Help

20.1 Language

```
FEATHER       : Add short spacings (image feathering)
FLUX          : Compute integrated flux from Support or Mask
HOW_TO        : Getting help on hwo to perform some action
MAP_CONTINUUM : Determine the continuum image from a spectral cube
MAP_POLAR     : Derive Polarization Images from Stokes Images
MASK          : Define the MASK
MFS           : Multi Frequency Synthesis (under development - not functiona
MOMENTS       : Compute Moments for data cubes
PROPER_MOTION : Apply proper motion information to UV data
EXTSTOKES       : Extract one Stokes parameter from multi-polarization UV t
UV_ADD        : Add some extra column to the current UV data
UV_CIRCLE     : Deproject and compute radial visibility profile
UV_CORRELATE  : Spectral matching for UV data
UV_DEPROJECT  : Deproject UV data
UV_FIT        : Fit UV data with simple functions
UV_MERGE      : Merge (possibly many) UV tables
UV_MOSAIC     : Split or Build a Mosaic UV table
UV_PREVIEW    : Quick look at the spectral aspects of the UV data
UV_RADIAL     : Deproject and compute azimutal average of UV data
UV_SHORT      : Compute and add short spacings to UV data
XY_SHORT      : Compute the short spacing image from the SINGLE Dish table
```

20.2 FEATHER

```
[ADVANCED\]FEATHER [?] [/FILE Combined HIGHRes LOWres] [/REPROJECT]
```

Combines a data cube containing High resolution data (HIGHres) with one containing the short spacing data (LOWres). Automatic reprojection and spatial resampling of the LOWres data may be done if the option /REPROJECT is specified.

The method is to hybridize in the UV plane the data, retaining short UV spacings from LOWres and long UV spacings from HIGHres. It is controlled mainly by variable FEATHER_RADIUS, the transition radius. Other variables allow some fine tuning.

FEATHER ? will list the input parameters

20.2.1 FEATHER /REPROJECT

```
[ADVANCED\]FEATHER [/FILE Combined HIGHRes LOWres] /REPROJECT
```

Allow to automatic reproject and rescale the LOWres images to the projection and sampling of the HIGHres one. If /REPROJECT is not present and the HIGHres and LOWres data do not match, FEATHER will complain and stop.

20.2.2 FEATHER /FILE

```
[ADVANCED\]FEATHER /FILE Combined HIGHRes LOWres [/REPROJECT]
```

Takes the data from files named HIGHres and LOWres and put the combined result in file named Combined. The default extension .lmv-sky.

Without the /FILE option, FEATHER takes HIGHres from the SKY buffer, LOWres from the Short Spacing buffer (SHORT, equal to SINGLE if it is a data cube, or derived by UV_SHORT if not), and put the result into an image variable named FEATHERED. The result can be written as any image by WRITE FEATHERED FileName.

Note: Currently, in all cases, the images reside in memory. A buffer mode for the /FILE option will be added later.

20.2.3 FEATHER Algorithm

FEATHER use the following steps
- Make oversampled Fourier Transform of both HIGHres and LOWres images
- Compute the truncation function f(r)
- Make the Truncated compact Fourier Transform, f(r) x T(LOW)
- Make the complement long baseline Fourier Transform, (1-f(r)) x T(HIGH)
- Sum them T(ALL) = R x f(r) x T(LOW) + (1-f(r)) x T(HIGH)
  where is taken from FEATHER_RATIO
- Make the inverse Fourier Transform
- Truncate the resulting image to original size and Mask of the HIGHres images

If LOWres is a single-dish image, f(r) should normally be the beam of that telescope to use optimal Signal-to-noise ratio from that data set. However, because of pointing errors and other calibration issues such as spectral baseline, using a sharper transition function gives better results. FEATHER uses a exp(-(r/Radius)^Expo) function, where Radius is taken from FEATHER_RADIUS and Expo from FEATHER_EXPO.

20.2.4 FEATHER Variables:

```
    FEATHER uses the folling input variables
FEATHER_RADIUS,    the transition radius
FEATHER_EXPO, an exponent controlling the transition sharpness
FEATHER_RATIO, a scale factor for the short spacing (LOWres) data.
FEATHER_RANGE, the limits of the overlapping region (see details)
```

20.2.5 FEATHER_RADIUS

```
    Real FEATHER_RADIUS  (default 15.0)
```

FEATHER_RADIUS is the transition radius (in meters) used to compute the combination function f(uv) = exp(-(uv/Radius)^Expo).

20.2.6 FEATHER_EXPO

```
    Real FEATHER_EXPO  (default 8.0)
```

FEATHER_EXPO is the exponent used to computed the combination function f(uv) = exp(-(uv/Radius)^Expo).

20.2.7 FEATHER_RATIO

```
    Real FEATHER_RATIO (default 1.0)
```

FEATHER_RATIO is the scale factor to be applied to the LOWres data in the combination.

20.2.8 FEATHER_RANGE

```
    Real FEATHER_RANGE[2]
```

FEATHER_RANGE[2] defines a region of "overlapping" UV spacings over which the ratio of LOWres and HIGHres visibilities can be checked. If the flux scale is correct, this ratio should be 1. FEATHER_RANGE defaults to FEATHER_RADIUS/1.15 and FEATHER_RADIUS*1.15.

The computed ratio is only a guide, and not used in the algorithm. A proper definition of FEATHER_RANGE requires knowledge of the initial UV coverage of the HIGHres and LOWres data sets. If FEATHER_RANGE is too large, the computed ratio has no real meaning.

20.3 FLUX

```
[ADVANCED\]FLUX Cursor|Mask|Support
```

Compute the integrated flux from the CLEAN image, using zones defined either by calling the Cursor, or by the current Support, or by the current Mask.

If the CLEAN data is a 3-D cube, the integrated flux will be a spectrum.

For FLUX MASK, if the Mask defines several separate zones, an integrated flux will be computed for each zone. Zones are ordered by decreasing number of pixels.

The FLUX results are available in the FLUX SIC structure, and can be displayed by SHOW FLUX and also SHOW COMPOSITE.

20.3.1 FLUX Limitations

FLUX MASK does not yet recognize properly Frequency dependent Masks.

20.3.2 FLUX Results

The FLUX results are available in the FLUX SIC structure

FLUX%NC	Number of channels
FLUX%FREQUENCIES[Flux%Nc]	Frequencies of the Channels
FLUX%VELOCITIES[Flux%Nc]	Velocities of the Channels
FLUX%NF	Number of Fields (Zones)
FLUX%VALUES[Flux%Nc,Flux%Nf]	Integrated Flux for each channel and Zone

For Cursor or Support defined regions, the enclosing polygon is also accessible:

FLUX%NXY	Number of polygon summits
FLUX%X	X coordinates of summits
FLUX%Y	Y coordinates of summits

20.4 HOW_TO

```
[ADVANCED\]HOW_TO  solve a simple problem
```

Ask IMAGER informations to help you solving your problem, expressed in a natural way. Examples
HOW do i subtract continuum ?

```
  HOW do i control the display ?
Rules:
  - simple words ("do" "i" "?" "the", etc...) are ignored. You can just
    type them to have a normal syntax in your question.
  - all significant words (e.g. "subtract" and "continuum") must be
    found to have a matching topic.
  - significant words use exact matching. "continuum" is not the same as
    "cont".
  - partial match is possible if a significant word ends up by *.
    "cont*" will match "continuum".
  - if no topic matches all significant words, a list of partial matches
    is given to guide you. Use a simpler question with less significant
    words among those proposed.

   HOW_TO ?
  will list all topics
```

20.5 MAP_CONTINUUM

[ADVANCED\]MAP_CONTINUUM [DIRTY|CLEAN|Other] [/METHOD KEY [Parameters [...]]]

Compute a continuum image from the Clean, Dirty or Other 3-D data set. The derivation of the continuum follows the algorithm specified in the /METHOD option (or the last specified in this way if the /METHOD option is not present).

20.5.1 MAP_CONTINUUM /METHOD

[ADVANCED\]MAP_CONTINUUM [DIRTY|CLEAN|Other] /METHOD KEY [Parameters [...]]

Change the algorithm used to compute the Continuum image. KEY is the name of the selected method. Available methods are GLOBAL, GAUSS, SCM, C-SCM, and EGM. The fastest method is GLOBAL, than GAUSS, than SCM.

The selected method is sticky: it will be used until further modified by the /METHOD option.

20.5.2 MAP_CONTINUUM GLOBAL

[ADVANCED\]MAP_CONTINUUM [DIRTY|CLEAN|Other] /METHOD GLOBAL [Threshold [Nhisto]]

Compute the integrated spectrum (over the region specified by CENTER and SIZE variables), and delineate the line-free region by Gaussian fit-

ting the histogram of the spectrum intensity.
The optional arguments control the algorithm. Threshold is clipping level in sigma units: values in range 2.5 - 5 are usable (default 3.5). Nhisto is the number of histogram bins. The default is to adjust this number depending on the number of channels in the data. This is the same algorithm as used in UV_CONTINUUM, or in a per pixel mode by the GAUSS method.

The continuum is then derived by integrating in the line-free regions. This method is fast, and gives a uniform noise since the same channels are used for all pixels. It may be inaccurate into several ways:
- Regions with strong broad line may still be contaminated by emission from the line wings
- The same applies to regions with line confusion.
- Continuum emission in regions with no line emission is not derived to the maximum sensitivity. This can be an issue in case of line confusion at some positions.

20.5.3 MAP_CONTINUUM GAUSS

[ADVANCED\]MAP_CONTINUUM [DIRTY|CLEAN|Other] /METHOD GAUSS [Threshold [Nhist]]

Compute the continuum based on the per-pixel spectrum. Each spectrum is sigma-clipped and the histogram of the intensity distribution is fitted by a Gaussian. Corrective techniques are applied to ensure the fit has converged. The position of this Gaussian gives the continuum level. A noise estimate is also derived.

The method does not deliver a spatially independent noise, but adapts the bandwidth to the line free zones for each pixel. It may fail for pixels with are close to line confusion.

The optional arguments control the algorithm. Threshold is clipping level value for the Gaussian: valuesi n range 2.5 - 4 are usable (default 3.5). Nhist is the number of histogram bins. The default is to adjust this number depending on the number of channels in the data.

20.5.4 MAP_CONTINUUM SCM

[ADVANCED\]MAP_CONTINUUM [DIRTY|CLEAN|Other] /METHOD SCM [Threshold [MaxIter]]

Compute the continuum based on the per-pixel spectrum. Each spectrum

is sigma-clipped until convergence of the noise. The method is similar to the GAUSS method, but more robust at the expense of lower speed because of the iterative median clipping. Threshold is the number of sigma used for clipping. MaxIter is the maximum number of clipping operations performed (default 6).

The method does not deliver a spatially independent noise, but adapts the bandwidth to the line free zones for each pixel. It may fail for pixels with are close to line confusion, though less than the GAUSS method.

20.5.5 MAP_CONTINUUM C-SCM

[ADVANCED\]MAP_CONTINUUM [DIRTY|CLEAN|Other] /METHOD C-SCM [Threshold [MaxIter]]

As for the SCM method, but the continuum is noise de-biased depending on the fraction of the bandwidth covered by emission and absorption lines. Marginally slower than SCM.

20.5.6 MAP_CONTINUUM EGM

[ADVANCED\]MAP_CONTINUUM [DIRTY|CLEAN|Other] /METHOD EGM [Threshold]

This method is similar to the GAUSS method, but uses an exponentially modified Gaussian to fit the intensity distribution histogram.

20.6 MAP_POLAR

[ADVANCED\]MAP_POLAR [GenericBegin [GenericEnd]] [/COMPUTE [Threshold]] [/BACKGROUND Type Min Max] [/STEP Step Xfirst Yfirst]

Derive or Display the Polarization fraction and Polarization angles images from the (I,Q,U) Stokes images.

GenericBegin indicates the file name prefix, and GenericEnd the filename postfix, including extension. The Stokes I (resp. Q, U) image is assumed to be in 'GenericName'"-I"'GenericEnd'(resp. -Q and -U) The Polarization fraction image will be called 'GenericBegin'"-P"'.Type', while the Polarization angle image is 'GenericBegin'"-A"'GenericEnd".

The default end is ".lmv-clean". However, when using with the Imager PIPELINE, continuum images are in derived from the UV Table names by

adding +C before the file extension, so GenericEnd should be "+C.lmv-clean".

MAP_POLAR ?
will simply display the current values of the "sticky" control parameters.

20.6.1 MAP_POLAR /COMPUTE

[ADVANCED\]MAP_POLAR GenericBegin [GenericEnd] [/COMPUTE [Threshold]]

Compute the Polarization Fraction image 'GenericBegin'"-P"'.Type', and the Polarization Angle image 'GenericBegin'"-A"'GenericEnd" from the 'GenericName'"-K"'GenericEnd' images where -K is all of the -I, -Q, and -U Stokes images.

The derivation is only made on pixels that are brighter than Threshold sigma in the Stokes I image. The use of the /COMPUTE option reset all parameters that control further display to their default values.

20.6.2 MAP_POLAR /BACKGROUND

[ADVANCED\]MAP_POLAR GenericBegin [GenericEnd] /BACKGROUND Type [Min Max] [/STEP Step Xfirst Yfirst]

Show the polarization vectors derived from the Polarization Angle and Fraction maps overlaid on a background image.

Type can be I for total intensity, F for fractional polarization in percent, or P for the polarized intensity. Min and Max indicate the SCALE to be used for the color scaling (through a temporary assignments of the SCALE values for SHOW command called by MAP_POLAR).

Type, Min, and Max, like Step, Xfirst, Yfirst are "sticky" (remanent): they keep the previous value until a new one is specified, or are reset to their default values by use of the /COMPUTE option. Default is I for Type, 0 for Min and Max, which implies an automatic scaling to the data range.

20.6.3 MAP_POLAR /STEP

[ADVANCED\]MAP_POLAR GenericBegin [GenericEnd] [/BACKGROUND Type Min Max] /STEP Step [Xfirst Yfirst]

Specify the Step in pixels between adjacent display of polarization vectors. If not specified, or if 0, Step is empirically derived from the angular resolution.

Xfirst and Yfirst indicate the starting pixel along the X and Y axis respectively, allowing a controlled centering of the vector distribution on the background image.

Type, Min, and Max, like Step, Xfirst, Yfirst are "sticky" (remanent): they keep the previous value until a new one is specified, or are reset to their default values by use of the /COMPUTE option. Default is 0 for Step, 1 for Xfirst and Yfirst.

20.7 MASK

```
[ADVANCED\]MASK [Key [Arguments ...]
```

Handle the MASK buffer, which can be used to select regions for Cleaning (see SUPPORT /MASK) or to mask any data cube.

Without argument (or with argument INTERACTIVE), an interactive tool is used to manipulate the mask. Valid operations are

ADD	Add a region to the mask
APPLY	Apply the mask to a buffer
CHECK	Check Clean and Mask consistency
INITIALIZE	Initialize the Mask (2D or 3D)
INTERACTIVE	Interactively define mask planes Step by Step
OVERLAY	Overlay the Mask to the Clean image
READ	Read the Mask from a file
REMOVE	Remove a region from the mask
SHOW	Show the Mask (as SHOW MASK)
THRESHOLD	Compute an automatic mask
USE	Use the Mask in Clean (as SUPPORT /MASK)
WRITE	Write the Mask on file (as WRITE MASK)

20.7.1 MASK Tricks

The MASK variable is ReadOnly. Yet, the user may want to modify it directly by himself. The following commands

```
DEFINE ALIAS WMASK MASK /GLOBAL
LET WMASK /STATUS WRITE
```

will create an alias to MASK which can be written by the user at will. This method is actually used in the Interactive MASK tool.

20.7.2 MASK ADD

```
[ADVANCED\]MASK ADD Shape [Arguments ...]
```

```
Add the specified Shape to the current mask.  Shape can be
CIRCLE      Ox Oy Diameter
ELLIPSE     Ox Oy Major Minor Angle
RECTANGLE   Ox Oy Major Minor Angle
POLYGON     File
```

where Ox Oy are the center of the shapes, Major and Minor the axes length (= side lengths for the Rectangle...) and Angle the Position Angle of the Major axis.

For the POLYGON, the cursor is called if no File argument is given; else the polygon is read from the specified File.

20.7.3 MASK APPLY

```
[ADVANCED\]MASK APPLY SicVariable
```

Apply the mask to the corresponding 3-D variable.

The space dimensions must coincide (pixel per pixel), but the spectral axes can differ. The command will select the appropriate Mask channel for each plane of the SicVariable. If the Mask is 2-D only, it will be applied to all planes.

20.7.4 MASK CHECK

```
[ADVANCED\]MASK CHECK [SicVariable]
```

Check the Mask consistency against the specified SicVariable. The default is against Clean.

20.7.5 MASK INITIALIZE

```
[ADVANCED\]MASK INITIALIZE 2D|3D
```

Initialize an empty 2-D or 3-D mask. For a 2-D mask, the interactive tool will use the mean image as a background for the definition.

20.7.6 MASK INTERACTIVE

```
[ADVANCED\]MASK INTERACTIVE [Nchan]
```

```
  Enter the interactive tool, moving Nchan channels at each
  Next or Previous button. Default is 1.

    [ADVANCED\]MASK
is equivalent to
    [ADVANCED\]MASK INTERACTIVE 1
```

20.7.7 MASK OVERLAY

```
    [ADVANCED\]MASK OVERLAY
```

Display a SHOW-like overlay of the Mask on top of the current image (default is Clean, normally). All SHOW parameters apply.

20.7.8 MASK READ

```
    [ADVANCED\]MASK READ File.[msk]

  Read the Mask from a Gildas data-cube.
```

20.7.9 MASK REMOVE

```
    [ADVANCED\]MASK REMOVE Shape [Arguments ...]

  Remove the specified Shape from the current Mask.  Shape can be
  CIRCLE      Ox Oy Diameter
  ELLIPSE     Ox Oy Major Minor Angle
  RECTANGLE   Ox Oy Major Minor Angle
  POLYGON     File
```

where Ox Oy are the center of the shapes, Major and Minor the axes length (= side lengths for the Rectangle...) and Angle the Position Angle of the Major axis.

For the POLYGON, the cursor is called if no File argument is given; else the polygon is read from the specified File.

20.7.10 MASK SHOW

```
    [ADVANCED\]MASK SHOW

  Similar to SHOW MASK command, but useing a spacing equal
  to 0.5 to illustrate the boundaries.
```

20.7.11 MASK THRESHOLD

[ADVANCED\]MASK THRESHOLD Value Unit [GUARD Guard] [SMOOTH Smooth Length]
[REGIONS Nregion]

Define a Mask from thresholding the CLEAN image. If the CLEAN image is 3-D, the Mask will be 3-D.

The method involves a first thresholding, optionally followed by a smoothing and a second thresholding to extend the support. The mask is the logical OR of the Raw and Smooth masks.

The algorithm to define the mask is controlled by 5 parameters, 4 of which are optional and specified as values after their corresponding keyword. The arguments

Value Unit

indicate the threshold and its unit under which the mask is set to 0. The unit can be %, SIGMA, NOISE, Jy, milliJy, microJy, K, milliK, microK, or NATIVE.

Value is not just a simple number: its interpretation depends on whether it has an explicit sigma (+ or -), in which case only values above of below are retained, of no sign at all, in which case comparisons are made on the absolute values.

The default is 6 sigma. The noise is taken from the computed Clean noise (clean%gil%rms) if defined, or from the theoretical noise (dirty%gil%noise) if not.

SMOOTH Smooth Length

Smooth

indicates the threshold under which the Smooth Mask is set to 0 after smoothing. There is no default. If keyword SMOOTH is not present, there is no smooth mask. This smoothed threshold is in the same units as the main one.

Length

is the size of the smoothing gaussian to derive the smooth mask from the initial mask (in arcsec). If set to *, or not specified, it takes the default which is the Clean beam major axis.

GUARD Guard

Guard

indicates the size of the guard band at edges where the mask is set to zero, in units of image size. The default is 0.18, i.e. the mask can extends a little more than the inner quarter. This is to avoid the edges where sidelobes aliasing occurs.

```
    REGIONS Regions
        Regions
```
is the maximum number of separate regions retained in the mask. The regions are ordered by decreasing size, and only the first Regions ones are kept. Default is 0, which means keep all regions.

20.7.12 MASK USE

```
[ADVANCED\]MASK USE
```

Use the MASK in Clean deconvolution.

Equivalent to SUPPORT /MASK command.

20.7.13 MASK WRITE

```
[ADVANCED\]MASK WRITE File
```

Save the Mask on a File. Default extension is .msk.

Equivalent to WRITE MASK command.

20.8 MFS

```
[ADVANCED\]MFS
```

** NOT OPERATIONAL -- UNDER DEVELOPMENT **

Multi-frequency synthesis, allowing to account for spectral index changes across the observed field of view for very wide bandwidths continuum imaging.

20.9 MOMENTS

```
[ADVANCED\]MOMENTS  [/CUTS] [/MASK] [/METHOD Mean|Peak|Parabolic]
                    [/RANGE Min Max TYPE] [/THRESHOLD Thre [Unit]]
```

Compute the 4 main "moments" of the CLEAN (or SKY) data cube, and return them as SIC Image variables. SKY has priority over CLEAN if both are defined.

The 4 moments are
M_AREA Integrated area over the velocity range

```
M_PEAK   Peak brightness value
M_VELO   Mean or peak velocity
M_WIDTH  Weighted line width
```

The options control the method, the selected part of the cube (default channels from FIRST to LAST), and the threshold for detection (default 3 sigma).

The resulting images can be displayed using command SHOW MOMENTS and saved using WRITE MOMENTS.

20.9.1 MOMENTS /CUTS

```
[ADVANCED\]MOMENTS  [/CUTS MinArea [NRegions]]
```

Restrict the moment images to the region(s) where the integrated area M_AREA is greater than MinArea. This option is exclusive of any other.

The optional NRegions number is the maximum number of separate regions to be kept (default 1).

20.9.2 MOMENTS /MASK

```
[ADVANCED\]MOMENTS  /MASK [/METHOD Mean|Peak|Parabolic]
                  [/RANGE Min Max TYPE] [/THRESHOLD Thre [Unit]]
```

Compute the 4 main moments and truncate them outside the current MASK.

20.9.3 MOMENTS /METHOD

```
[ADVANCED\]MOMENTS  /METHOD Mean|Peak|Parabola [/MASK]
                  [/RANGE Min Max TYPE] [/THRESHOLD Thre [Unit]]
```

Specify which method to be used to compute the velocity and peak values. The default method, MEAN, computes an intensity weighted velocity, and takes the peak intensity at each pixel. Method PEAK takes the velocity at the peak intensity. Method PARABOLA makes a parabolic fit over 3 channels around the peak intensity channel to derive the velocity, and takes the peak value of this parabola for the peak brightness.

20.9.4 MOMENTS /RANGE

```
[ADVANCED\]MOMENTS /RANGE Min Max TYPE [/MASK]
  [/METHOD Mean|Peak|Parabola] [/THRESHOLD Thre [Unit]]
```

Specify the range of channels to be selected. TYPE can be VELOCITY, FREQUENCY or CHANNELS. If not present, the range is defined by FIRST and

LAST variables.

20.9.5 MOMENTS /THRESHOLD

```
[ADVANCED\]MOMENTS /THRESHOLD Thre [UNIT] [/MASK]
  [/METHOD Mean|Peak|Parabola] [/RANGE Min Max TYPE]
```

Specify the minimum intensity of the input data cube to consider a given pixel,channel as valid. Keyword UNIT can be given to indicate whether Thre is given in noise unit (UNIT = SIGMA) or in data cube units (UNIT = FLUX).

The default arguments of /THRESHOLD are 3 SIGMA.

20.10 PROPER_MOTION

```
[ADVANCED\]PROPER_MOTION muRA muDec [/FILE FileIn FileOut]
```

Apply specified proper motion to current or specified UV Table. Proper motion values are specified in milliarcsec per year (mas/a).

If no /FILE option is present, they are applied to the current UV table.

20.10.1 PROPER_MOTION /FILE

```
[ADVANCED\]PROPER_MOTION muRA muDec /FILE FileIn FileOut
```

Apply specified proper motion to the specified UV Tables. Proper motion values are specified in milliarcsec per year (mas/a).

Currently, the input file FileIn and output file FileOut must differ.

20.11 STOKES

```
[CLEAN\]STOKES Key [/FILE UVin UVout]
```

Derive a single polarization UV table with the polarization state specified by Key from a multi-polarization UV table. The command work on the UV buffer unless the /FILE option is present.

Key is any of the following: NONE, I, Q, U, V, RR, LL, HH, VV, XX, YY.

A typical use is after command FITS on CASA data:
FITS Fits.uvfits TO UVin.uvt

```
STOKES NONE /File UVin Uvout
```

20.11.1 STOKES /FILE

```
[CLEAN\]STOKES Key /FILE UVin UVout
```

Derive a single polarization UV table UVout with the polarization state specified by Key from a multi-polarization UV table UVin. Key is any of the following: NONE, I, Q, U, V, RR, LL, HH, VV, XX, YY.

20.12 UV_ADD

```
[ADVANCED\]UV_ADD ITEM [Mode] [/FILE FileIn FileOut]
```

Compute and add some missing information in a UV Table, such as the Doppler correction and the Parallactic Angle

The information is derived from the Observatory coordinates, from the Telescope name in the input UV table. This can be supplied by the SPECIFY TELESCOPE command if not available.

ITEM can take values DOPPLER, PARALLACTIC or * for both.

Mode is a debug control integer which indicates in which column the information should be placed. The default is 0, i.e. the command will reuse the column of the appropriate type, or add it if not present. It may be set to 3, as often column 3 contains the Scan number, which is not a relevant information for imaging. Other values are at the user's peril...

20.12.1 UV_ADD /FILE

```
[ADVANCED\]UV_ADD ITEM [Mode] /FILE FileIn FileOut
```

Use the UV data in the file FileIn, and write the completed visibilities in the file FileOut.

20.13 UV_CIRCLE

```
[ADVANCED\]UV_CIRCLE [x0 y0 Rota Incli]
[/SAMPLING QSTEP [Unit [QMIN QMAX]]] [/ZERO [Flux]]
```

Compute a UV table containing the radial distribution of the azimutal

average of the visibilities after deprojection for inclination and orientation (Incli, Rota, in Degrees) around a specified center (x0,y0 in Radians). All arguments default to 0. ROTA is here the position angle of axis of (assumed) rotational invariance of the object. Thus, when using results from UV_FIT one should use ROTA = PA-90, where PA is the Position Angle of the Major axis.

The resulting UV table have visibilities only with V values equal to 0, so that the U values contain the radial distance in the UV plane. This is convenient to display the azimutal average of the visibilities as a function of UV distance, but cannot be used for further imaging.

See UV_RADIAL if you want to image the circular average of the brightness distribution.

20.13.1 UV_CIRCLE /SAMPLING

```
    [ADVANCED\]UV_CIRCLE  [x0 y0 Rota Incli] /SAMPLING QSTEP [Unit [QMIN
QMAX]]
  [/ZERO [Flux]]
```

Specify the sampling of the UV distances: Qstep is the step, Qmin and Qmax the min and max. Distances are in the specified unit, which can be "meter" (default) or "kwave". If /SAMPLING is not present, an automatic guess is made from the minimum and maximum baselines and the dish diameter.

20.13.2 UV_CIRCLE /ZERO

```
    [ADVANCED\]UV_CIRCLE [x0 y0 Rota Incli] /ZERO [Flux]
  [/SAMPLING QSTEP [Unit [QMIN QMAX]]]
```

Add the zero spacing flux to the azimutal average. If no value is given, the zero spacing flux is extrapolated from the shortest baselines using a parabolic interpolation centered on (u,v)=(0,0).

20.14 UV_CORRELATE

```
    [ADVANCED\]UV_CORRELATE ResultFile [/FILE Data Model]
```

Compute the Frequency Cross-Correlation spectrum of two UV data sets. The two UV data set must be referenced to the same position.

The data sets can be specified with the /FILE option. If not, UV_CORRELATE will correlate the current UV data (UV_DATA) with the Model UV data

(UV_MODEL), that can be loaded by READ UV and READ MODEL respectively. The Model UV data can also be computed by command MODEL.

ResultFile indicates on which data file the result must be written. This result is a UV table with only the (0,0) spacing, that contains the correlation spectrum. It can be displayed using command UV_PREVIEW.

See command BUNDLES\UV_DETECT (that encapsulates in a more comprehensive way UV_CORRRELATE) to compute the same result from a UV data file and an Image model.

20.14.1 UV_CORRELATE /FILE

[ADVANCED\]UV_CORRELATE /FILE Data Model

Compute the Frequency Cross-Correlation spectrum of two UV data sets in the specified file names.

The two UV data set must be referenced to the same position, and have a compatible velocity resolution.

20.15 UV_DEPROJECT

[ADVANCED\]UV_DEPROJECT x0 y0 Rota Incli

Deproject a UV table for inclination and orientation (Incli, Rota, in Degrees) around a specified center (x0,y0 in Radians). This can be useful for almost planar or cylindrical objects (e.g. galaxies, or protoplanetary disks). When using results from UV_FIT one should use ROTA = PA-90, where PA is the Position Angle of the Major axis for a proper deprojection.

The result becomes the current UV table.

20.16 UV_FIT

[ADVANCED\]UV_FIT [Func1 .. FuncN] [/QUIET] [/SAVE File] [/WIDGET N]

Fit UV data with a few simple functions. Func1 to FuncN (currently N < 5) are the names of the functions to be fitted. If not specified, the last names (or those attributed by the /WIDGET options) are used.

The models are either simple functions or linear combinations of simple functions. The results of the fitting process are the position offsets in R.A. and Dec (in arc second) of the model source from the phase ref-

erence center, and its flux (Jansky). Depending on the fitting functions additional fit results are possible. Currently supported distributions and additional fit parameters are:

```
POINT     Point source              : None
E_GAUSS   Elliptic Gaussian source  : FWHM Axes (Major and Minor), Pos Ang
C_GAUSS   Circular Gaussian source  : FWHM Axis
C_DISK    Circular Disk             : Diameter
E_DISK    Elliptical (inclined) Disk: Axis (Major and Minor), Pos Ang
RING      Annulus                   : Diameter (Inner and Outer)
EXPO      Exponential brightness    : FWHM Axis
POWER-2   B = 1/r^2                 : FWHM Axis
POWER-3   B = 1/r^3                 : FWHM Axis
E_RING    Inclined Annulus          : Inner, Outer, Pos Ang, Ratio
E_EXPO    Elliptic exponential       : FWHM Axes (Major and Minor), Pos Ang
SPERGEL   Spergel brightness profile : FWHM Axis, nu
E_SPERGEL Elliptic Spergel profile   : FWHM Axes (Maj. and Min.), Pos Ang, n
```

The function parameters are found in a SIC structure named UVF% under names UVF%PARi%PAR[7] (starting values), UVF%PARi%RANGE[7] (starting ranges) and UVF%PARi%START[7] (number of starts). The /WIDGET option is a convenient way to set these variables.

UV_RESIDUAL will compute the fit residual when used after UV_FIT, while it computes the residual from the Clean component list if used after CLEAN.

WRITE UV_FIT file[.uvfit] will save the fit results in a GILDAS table, in the same format than that of the UV_FIT task.

The Pos Ang (PA) are the Position Angles of the major axis of the distribution. To deproject the elliptical shapes, one should use the Position Angle of the rotational axis, PA-90 (see commands UV_CIRCLE, UV_DEPROJECT and UV_RADIAL)

20.16.1 UV_FIT /QUIET

[ADVANCED\]UV_FIT /QUIET [/WIDGET N]

Activate the quiet mode. Only a progress report (25%, 50% and 75% done) is issued in this case, not a per-channel message. This mode is recommended for many channels.

20.16.2 UV_FIT /SAVE

[ADVANCED\]UV_FIT /SAVE OutputFile

Save the input parameters, into a text output file. This file is then a script which can be re-executed to set the input parameters for UV_FIT.

If the data has only 1 channel, the fit results (see HELP UV_FIT Result-Values) are also written in the same file.

20.16.3 UV_FIT /WIDGET

```
[ADVANCED\]UV_FIT   /WIDGET N [/QUIET]
```

Create a Widget to specify the function names and input parameters for UV_FIT for N functions. Once these are defined by the user, clicking the GO button will launch the computation.

20.16.4 UV_FIT History

[ADVANCED\]UV_FIT [Func1 .. FuncN] [/QUIET] [/SAVE File] [/WIDGET N]

This command is similar to the pre-existing task UV_FIT, but works on the current UV buffer, thus offering a simpler integration in IMAGER.

20.16.5 UV_FIT ResultTable

The UV_FIT results are stored in an internal table named UV_FIT. This table can later be saved using command WRITE UV_FIT FileName. The table is organized as a MxN matrix, where M is the number of channels in the UV data and the organization in N is the following:
N: P1 P2 P3 Vel A1 A2 A3 Par1 Err1 Par2 Err2 ... A1 A2 A3 Par1 Err1 ...

```
where P1 = RMS of the fitting process
      P2 = number of supplied functions  (NF$)
      P3 = total number of parameter
     Vel = velocity of the i-th channel (i lies between 1 and M)
      A1 = 1 (for the first function), = 2 (for the second function)
      A2 = function code (POINT = 1, E_GAUSS = 2, ... , POWER-3 = 8)
      A3 = number of parameters of the function
    Parx = result of the fit parameter (order of appearance in PARAMxx$)
    Errx = error of the fitting process for the parameter Parx
```

For example, fitting models with the two functions POINT and C_GAUSS produce files with N=24.

SHOW UV_FIT will display plots from this table.

20.16.6 UV_FIT ResultValues

For data with only 1 channel, the fit UV_FIT results are available as variables UVF%PARi%RESULTS[7] (for the results) and UVF%PARi%ERRORS[7] (for their formal errors) for every function number i. These variables are written in the output file by the UV_FIT /SAVE OutputFile command.

They are also available in the internal table named UV_FIT, as for any other number of channels (see HELP UV_FIT ResultTable)

20.17 UV_MERGE

```
[ADVANCED]\UV_MERGE OutFile /FILES In1 In2 ... Inn
[/MODE [LINE|STACK|CONCATENATE|CONTINUUM [Index [Frequency]]]
[/SCALES F1 ... Fn] [/WEIGHTS W1 ... Wn ]
```

Merge many UV data files, with calibration factor and weight factors and (for Line data) spectral resampling as in the first one. OutFile is the name of the output UV table.

For Line data, the default is to merge lines of the same molecular transition (same Rest Frequency). However the STACK mode allows stacking UV data from different spectral lines, re-aligned in velocity. This can allow detection of molecules with many transitions.

For Continuum data (1 channel and/or option /MODE CONTINUUM), a spectral index and a reference Frequency can be specified to merge all UV tables.

20.17.1 UV_MERGE /FILES

```
[ADVANCED]\UV_MERGE OutFile /FILES In1 In2 ... Inn
[/MODE [LINE|STACK|CONCATENATE|CONTINUUM [Index [Frequency]]]
[/SCALES F1 ... Fn] [/WEIGHTS W1 ... Wn ]
```

Specify the names of the UV tables to be merged. The first one is used as a reference for resampling (line data) or Frequency (continuum data).

Instead of individual file names, a character array variable can be used to specify the input UV tables. A common use is with the DIR%FILE array created by command SIC FIND:

```
  SIC FIND A-*.uvt
  UV_MERGE M-all /FILES DIR%FILE
will merge in Line mode all files of name A-*.uvt into M-all.uvt
```

20.17.2 UV_MERGE /MODE

```
    [ADVANCED]\UV_MERGE OutFile /FILES In1 In2 ... Inn
  /MODE [LINE|STACK|CONCATENATE|CONTINUUM [Index [Frequency]]
  [/SCALES F1 ... Fn] [/WEIGHTS W1 ... Wn ]
```

Specify the merging mode. The default mode is LINE.

In LINE mode, for spectral line UV tables (more than 1 channel), the default is that the spectral lines must have the same Rest Frequency. Resampling (in velocity, which is then identical in frequency) is done on the grid of the first UV table In1.

In mode CONCATENATE, a common spectral axis encompassing those of the initial UV tables is defined, and all UV tables are resampled on this one.

In mode STACK, the Rest Frequencies can differ. Resampling is then done in velocity, and the (u,v) coordinates scaled as the Rest Frequency ratios to conserve the angular resolution of the data. The /SCALES and /WEIGHT factors can be used to incorporate prior knowledge of the expected line rations to optimize S/N.

For continuum UV tables (1 channel or option /MODE CONTINUUM), a spectral index can be specified, as well as a reference Frequency. (u,v) coordinates are scaled appropriately, as well as Flux and Weights in this case. The /SCALES and /WEIGHTS factors are applied on top of this automatic spectral index scaling. Input Line UV Tables are treated as multi-frequency Continuum ones (as in UV_CONTINUUM command).

20.17.3 UV_MERGE /SCALES

```
    [ADVANCED]\UV_MERGE OutFile /FILES In1 In2 ... Inn
  [/MODE [LINE|STACK|CONCATENATE|CONTINUUM [Index [Frequency]]]
  /SCALES F1 ... Fn [/WEIGHTS W1 ... Wn ]
```

Specify the flux scaling factors for each UV table.

For /MODE CONTINUUM, the spectral index is applied separately from this flux scale factor (by further multiplication).

20.17.4 UV_MERGE /WEIGHTS

```
  [ADVANCED]\UV_MERGE OutFile /FILES In1 In2 ... Inn
[/MODE [LINE|STACK|CONCATENATE|CONTINUUM [Index [Frequency]]]
[/SCALES F1 ... Fn] /WEIGHTS W1 ... Wn
```

Specify the weight scaling factors for each UV table. The weight scaling factor is independent of the flux scaling factor. This means that to conserve the Signa-to-Noise ration, one should normally use Wi = 1/Fi^2.

For /MODE CONTINUUM, the spectral index is applied separately from this weight scaling factor (by further multiplication).

20.18 UV_MOSAIC

```
[ADVANCED]\UV_MOSAIC Mosaic  MERGE|SPLIT [Fields [...]]
```

Built a Mosaic UV table from individual fields (MERGE keyword), or split a Mosaic UV table in individual fields (SPLIT keyword)

For MERGE, all single-field UV tables must match spectroscopically. Use command UV_RESAMPLE /LIKE on individual fields if needed to ensure this.

20.18.1 UV_MOSAIC MERGE

```
[ADVANCED]\UV_MOSAIC Mosaic  MERGE Fields [...]
```

Build a new Mosaic UV table from the single-fields UV tables indicateed by the arguments following the MERGE keyword.

The Mosaic UV table name will be Mosaic.uvt

If only 1 argument is following the keyword MERGE, it is assumed to be a SIC Character array variable containing the filenames of all single-field UV tables needed to build the mosaic. This is typically used as follows:

```
IMAGER> SIC FIND fields*.uvt
IMAGER> UV_MOSAIC my_mosaic MERGE dir%file
```

The number of fields is thus given by the size of the array.

If more than 1 argument follow the keyword MERGE, each one indicates a

separate single-field UV table. It is not possible to use a character array in this case.

20.18.2 UV_MOSAIC SPLIT

[ADVANCED]\UV_MOSAIC Mosaic SPLIT [Fields]

Split a UV table of name "Mosaic.uvt" into a series of single-fields UV tables of generic name Fields. Individual names are then Fields-1.uvt, ... Fields-n.uvt

If not specified, Fields default to the name given by Mosaic.

20.19 UV_PREVIEW

[ADVANCED\]UVPREVIEW [TAPER Ntaper] [CLIP Threshold] [HISTO Nhisto] [SMOOTH Nsmooth] [/BROWSE [Ataper]] [/FILE UvData.uvt [Drop]]

A fast previewer to figure out if there is signal and what is its spectral shape. The command attempts to find out the line free regions and to estimate a continuum region.

The output of UV_PREVIEW can be used for further processing commands (UV_BASELINE and UV_FILTER, UV_SPLIT, SPECIFY, etc...)

If a catalog is defined (see HELP CATALOG), it will also display line identification (red for detected ones, blue for the others).

The behaviour is controlled by 4 parameters than can be set through the "KEY Value" optional argument pairs. These are

```
TAPER   Ntaper    : number of scale sizes  (default 4)
CLIP    Threshold : Clipping level in Sigma (default 3.5)
SMOOTH  Nsmooth   : Number of spectral smoothing (default 3)
HISTO   Nhisto    : Size of histogram
```

These values are "sticky": they remain valid until further change, and they can be listed by

```
UV_PREVIEW ?
```

20.19.1 UV_PREVIEW /BROWSE

[ADVANCED\]UV_PREVIEW /BROWSE [Ataper] [/FILE FileIn] [Drop]]

Use a "split window" previewer instead of just one window. The bottom window displays full spectra for all tapers, the top window is a us-

er-controlled zoom of one of the tapers, with spectral line identifications if a catalog is available. The cursor is called and the user can adjust the zoom display with it. Press H for help, E for exit.

Ataper indicates the taper number (default: last one).

20.19.2 UV_PREVIEW /FILE

[ADVANCED\]UV_PREVIEW [Key Value [...]] /FILE FileIn [Drop] [/BROWSE Ataper]

Without /FILE, UV_PREVIEW works from the current UV data set.

With the /FILE option, it will pre-view the UV data set from the specified file. Edge channels are automatically dropped in this mode, as many telescopes (NOEMA or ALMA) do not produce useable data in these ones. The default drop is 5% of bandwidth on each side, but can be changed with the option Drop value (in %).

20.19.3 UV_PREVIEW Algorithm

UV_PREVIEW computes for Ntapers different tapers the spectrum towards the phase center. The taper ranges are determined from the available baseline lengths and telescope diameter. Ntapes is controlled by the value of Key TAPER.

These spectra are then smoothed by averaging several channels (1,4,16 and 64) to be sensitive to different linewdiths. The nummber of smoothed spectra per taper is controlled by the value of Key SMOOTH.

For each spectrum, UV_PREVIEW then attempts to figure out if there is line emission and the line-free channels to define the continuum level. This is based on the histogram of the intensity distribution of all channels. The most likely value and the noise level is derived from this histogram. An iterative scheme, blanking out of range (presumably line emission) channels, is used for this to converge towards a Gaussian histogram, which normally represents the noise distribution around the continuum level.

As a last step, blanked channels are accumulated in a list of channels, which thus contain possible line emission at any of the Ntapers angular scales and Nsmooth spectral resolutions.

20.19.4 UV_PREVIEW Limitations

UV_PREVIEW cannot identify lines if there are two few channels. It will only display the spectra in this case. A minimum of 32 channels is required, but confused spectra may also prevent a proper line recognition.

The line detection is based on the current specified velocity. If this is incorrect, lines may appear shifted and considered as not detected. PREVIEW%TOLERANCE indicates the matching precision in frequency (default is 2 MHz).

If variable \sicvar{REDSHIFT} exists, the frequency scale is corrected for that source Redshift, so that line identification remains possible.

20.19.5 UV_PREVIEW Output

UV_PREVIEW returns the list of possible line channels through variable PREVIEW%CHANNELS. If no line emission was identified at any scale, the list is empty and the variable does not exist.

With this list, the user can compute the continuum image, using commands UV_FILTER /CHANNELS PREVIEW%CHANNELS (or simply UV_FILTER), then UV_CONTINUUM and the usual UV_MAP and CLEAN. Alternatively, the user can filter out the continuum emission using UV_BASELINE.

If a catalog is present, UV_PREVIEW also creates several other variables. PREVIEW%FOUND indicates the spectral line falling in the frequency range, while PREVIEW%DETECTED indicates the dectected lines. These two structures have the following variables, e.g. for FOUND:
PREVIEW%Found%FREQ The line frequencies
PREVIEW%Found%LINES The line names
PREVIEW%Found%SPECIES The molecule name
PREVIEW%EDGES and PREVIEW%FREQUENCIES contains the start and end channels (for %EDGES, resp. frequencies for %FREQUENCIES) of each contiguous range of channels in PREVIOUS%CHANNELS. These variables are used for line identification and imaging in the
@ image_lines
script.

Finally, UV_PREVIEW returns in PREVIEW%FMIN PREVIEW%FMAX the frequency coverage, and in PREVIEW%FREQ the rest frequency of the most likely spectral line in the window, and in PREVIEW%LINES its name. If no spectral line has been identified, PREVIEW%FREQ is just the mean of PREVIEW%FMIN and PREVIEW%FMAX, and PREVIEW%LINES does not exist.

20.20 UV_RADIAL

[ADVANCED\]UV_RADIAL x0 y0 Rota Incli [/SAMPLING QSTEP [QMIN QMAX]] [/ZERO [Flux]]

Compute a UV table containing the radial distribution of the azimutal average of the visibilities after deprojection for inclination and orientation (Incli, Rota, in Degrees) around a specified center (x0,y0 in Radians). ROTA is here the position angle of axis of (assumed) rotational invariance of the object. Thus, when using results from UV_FIT one should use ROTA = PA-90, where PA is the Position Angle of the Major axis.

The result becomes the current UV table. The resulting UV table has a (u,v) coverage which is extended by rotation, so that it can be used to image the (rotationally symmetric) 2-D radial distribution using standard commands like UV_MAP and CLEAN. The radial profile of the brightness distribution can then be recovered by using any radial cut through this image.

Note, however, that in a SHOW UV command with "radius" along one axis, all visibilities at different azimut are superimposed. See UV_CIRCLE for a different way of computing an azimutal average, more suited for SHOW UV.

20.20.1 UV_RADIAL /SAMPLING

[ADVANCED\]UV_RADIAL [x0 y0 Rota Incli] /SAMPLING QSTEP [Unit [QMIN QMAX]]
[/ZERO [Flux]]

Specify the sampling of the UV distances: Qstep is the step, Qmin and Qmax the min and max. Distances are in the specified unit, which can be "meter" (default) or "kwave". If /SAMPLING is not present, an automatic guess is made from the minimum and maximum baselines and the dish diameter.

20.20.2 UV_RADIAL /ZERO

[ADVANCED\]UV_RADIAL [x0 y0 Rota Incli] /ZERO [Flux]
[/SAMPLING QSTEP [Unit [QMIN QMAX]]]

Add the zero spacing flux to the azimutal average. If no value is given, the zero spacing flux is extrapolated from the shortest baselines using a parabolic interpolation centered on (u,v)=(0,0).

20.21 UV_SHORT

```
[ADVANCED\]UV_SHORT [Arg] [/CHECK] [/REMOVE]
```

(documentation for Version 3.9 of command UV_SHORT)

Compute the Short Spacings from the current Single Dish dataset (read by READ SINGLE) and merge it to the current UV data. If the SINGLE dish data is a Class table, UV_SHORT also creates the SHORT image variable, containing the 3-D data cube from which short spacings are computed.

UV_SHORT takes sensible default guesses for most parameters. UV_SHORT ? lists the essential parameter values, UV_SHORT ?? some additional ones, and UV_SHORT ??? even the debugging control variables.

The current values can be overriden by the user, who need to set (and if needed to define first) the corresponding SHORT_whatever variable. SHORT_SD_FACTOR is the main one which may need to be specified by the user, as the Single Dish data is rarely in the appropriate unit.

The resulting UV table becomes the current UV data, and can be imaged, written, etc...

20.21.1 UV_SHORT /CHECK

```
UV_SHORT /CHECK
```

Verify the spectral compatibility between the "single" data set (obtained by READ SINGLE) and the UV data obtained by READ UV. The compatibility is returned in variable SHORT_STATUS, which is set to YES if both data sets can be merged by UV_SHORT.

20.21.2 UV_SHORT /REMOVE

```
UV_SHORT /REMOVE
```

Removes any short spacing from the current UV data set.

20.21.3 UV_SHORT Algorithm

UV_SHORT task computes pseudo-visibilities for short or zero spacings from a single dish table of spectra (Class table) or LMV data cube. These pseudo visibilities are appended to the current (presumably a Mosaic) UV table.

Short spacings are computed when the Interferometer dish diameter is smaller than the Single-dish diameter, Zero spacings otherwise (see HELP UV_SHORT Zero_Spacing for this case)

For short spacings, the command performs two steps
(1) Creation of a "well behaved" map from the spectra.
(2) Extraction of UV visibilities from this map.

See HELP UV_SHORT Step_i for detailed explanations of the method steps.

With recent UV tables and Single Dish CLASS table, most parameters are automatically determined. The only parameter to be specified remains SHORT_SD_FACTOR (although that one may also be determined automatically if the input single dish data set is in main-beam brightness temperature).

A parameter set to 0 value indicates the appropriate default should be used.

20.21.4 UV_SHORT Zero_Spacing

Zero spacings are computed when the single dish diameter is the same as the interferometer dish diameter. Zero spacing extraction proceeds differently for Class data tables and 3-D data cubes.

In the data cube case, the nearest pixel matching the direction of each field is taken as the Zero spacing for this field. If there is no point close enough, according to the specified position tolerance SHORT_TOLE, an error occurs.

In the Class data table case, all spectra within the specified position tolerance of a field center are averaged together to produce the Zero spacing. If none is found, an error occcurs.

20.21.5 UV_SHORT Step_1

Step (1) Creation of a "well behaved" map from the spectra. This

map is made available to the user as the SHORT datacube, and can be saved by WRITE SHORT.

Step (1) only can be performed independently by command XY_SHORT, to use the SHORT image in command FEATHER for example.

Step (1) only occurs if the input single-dish data set (read by READ SINGLE) is a table of spectra. The table format is described in the CLASS\GRID command of CLASS.
The identification of the input single-dish data set as a table of spectra is based on the Header.

It is recommended that this input table is a collection of single-dish, Nyquist sampled spectra covering twice the interferometric field of view of interest. However, UV_SHORT does *NOT* make any assumption. It thus tries to compute a "well behaved" map by linear operations (convolutions) from the original spectra, in an optimum way from signal to noise point of view. The map is extrapolated smoothly towards zero at the map edge in order to avoid further aliasing in the Fourier transform operations required in Step (2). This extrapolation has a scale length of twice the single-dish beam, in order to avoid spurious Fourier components.

In detail, UV_SHORT (or XY_SHORT) performs the following operations:
- Resampling (in space) of the original spectra on a regular grid by convolution with a small (typically 1/4 of the single-dish beam) gaussian convolving kernel. In this process, the weights of individual spectra is carried on a weight map.
- Extrapolation by zero outside the convex hull of the mapped region.
- Convolution of the result by a gaussian twice as wide as the single-dish beam. Within the convex hull of the mapped region, the smoothed map is replaced by the original map.

20.21.6 UV_SHORT Step_2

Step (2) Extraction of UV visibilities from this map.

From the given input data cube, or the "well behaved" data cube created by Step (1), UV_SHORT computes the visibilities in the following way:

- Fourier transform of the single dish map;
- Division by the Fourier transform of the single dish beam, up to a maximum spacing (SHORT_SD_DIAM, in meters);
- Inverse Fourier transform to the image plane and then for each pointing center;
- Multiplication of the image by the primary beam of the interferometer elements;

- Fourier transform back to the UV plane;
- Creation of the visibilities, with a given weight SHORT_SD_WEIGHT and an appropriate calibration factor to Janskys SHORT_SD_FACTOR.

Both the single-dish and the interferometer antennas are assumed to have gaussian beams (SHORT_SD_BEAM and SHORT_IP_BEAM, in radians).

20.21.7 UV_SHORT Variables:

Control variables for UV_SHORT are not predefined, except for the 3 main ones: SHORT_SD_FACTOR, SHORT_SD_WEIGHT and SHORT_UV_TRUNC.

All others should be defined by the user in case the default value is not appropriate, with their appropriate (Real, Char or Logical) types.

20.21.8 SHORT_DO_SINGLE

Logical value, should be YES except for test purposes.

20.21.9 SHORT_DO_PRIMARY

Logical value, should be YES except for test purposes.

20.21.10 SHORT_FTOLE

The tolerance in frequency / velocity alignment. The default is 0.1 channel at edges.

20.21.11 SHORT_IP_BEAM

Half-power beam width of the interferometer antennas, in radians. The beam is assumed to be gaussian.

Default value is 0, meaning that the beam is taken from the Telescope section if present.

20.21.12 SHORT_IP_DIAM

Interferometer diameter for which UV_SHORT will compute short spacing visibilities.

Default value is 0, meaning that the diameter is taken from the Telescope section if present.

20.21.13 SHORT_MCOL

*** Obsolescent ***

See READ SINGLE ClassTable.tab /RANGE command for an equivalent method of selecting the appropriate channel range.

The first and last column to be mapped. For tables produced by GRID command of CLASS, SHORT_MCOL[1] should be 4 and SHORT_MCOL[2] can be set to 0 to process all channels.

Default value: 4 0, appropriate for tables coming from CLASS\GRID command.

20.21.14 SHORT_MIN_WEIGHT

The minimum weights under which a given point in the map should be filled by the smooth map rather than by the gridded (original] map.

Default value: 0.01

20.21.15 SHORT_MODE

This is an integer code used for backward compatibility with an older version of the UV_SHORT task, and also for test purpose.

Allowed values are :

-1 indicates to create a single UV table with columns for the Phase center offsets only
-2 indicates to create a UV table with columns for the Pointing center offsets
-3 indicates to create a UV table with the additional columns

```
    type being Pointing or Phase, as in the original UV_TABLE$
-4  Allow to override the antenna diameter in the Telescope section
+1  indicates to append to the initial UV table the short spacings
    with Phase center offsets (which must thus match the initial UV
    table shape)
+2  indicates to append to the initial UV table the short spacings
    with Pointing center offsets (which must thus match the initial
    UV table shape)
+3  indicates to append to the initial UV table the short spacings
    (The extra column type being determined automatically).
+4  Allow to override the antenna diameter in the Telescope section

The default value is 3, i.e. automatic merging with the current  UV  ta-
ble.
```

20.21.16 SHORT_SD_BEAM

```
Half-power  beam  width  of the single dish antenna, in arcseconds.  The
beam is assumed to be gaussian.

  Default value is 0, meaning that the beam is taken from the  Telescope
section if present.
```

20.21.17 SHORT_SD_DIAM

```
Single dish diameter used to produce the input spectra, in meters.

  Default  value is 0, meaning that the diameter is taken from the Tele-
scope section if present.
```

20.21.18 SHORT_SD_FACTOR

```
Multiplicative calibration factor; it is used to convert from the single
dish map units (e.g., main-beam brightness temperature) to janskys.

A default value of 0 can be used if the original data file is in unit of
Tmb, the main beam brightness temperature, because in this case the con-
version factor can be derived from the beam size. UV_SHORT will complain
if it cannot derive the appropriate value.
```

20.21.19 SHORT_SD_WEIGHT

Weight scaling factor for the generated visibilities.

It is a relative scaling factor in the weights compared to a supposedly optimal weighting to give the best combined synthesized beam. That optimal weighting essentially gives the same weight density par unit area in the UV plane than the shortest baselines measured with the interferometer only. However, if the single-dish data has not been observed long enough, or has baselines problems for example, this weight may add noise to the overall data set, so could be down-weighted.

Default: 1.0

20.21.20 SHORT_TOLE

The tolerance in position (in radians). The behaviour differ for Short and Zero spacings and Table or 3-D cubes as Single-Dish data.

If the Single-Dish data is a table of spectra, Spectra differing by less than this amount will be added together prior to gridding. A recommended value is below 1/10th of the Single Dish primary beam. This is valid for Short Spacings and Zero Spacing cases.

If the Single-Dish data is 3-D data cube, SHORT_PTOLE is used only for Zero Spacings. If no pixel is within SHORT_PTOLE of an Interferometer pointing center, no short spacing is added for this field and an error occur.

Default value is 0, meaning using 1/16th of the Single Dish primary beam.

20.21.21 SHORT_UV_MAX

No visibility at spacings higher than SHORT_UV_MAX is generated. Theoretical consideration on the method used in this task implies that SHORT_UV_MAX should be at most (SHORT_SD_DIAM-SHORT_IP_DIAM). Smaller values may need to be applied if, for example, the pointing accuracy of the Single Dish is insufficient.

Default value is 0, meaning to use SHORT_SD_DIAM-SHORT_IP_DIAM

20.21.22 SHORT_UV_MIN

No visibility at spacings shorter than SHORT_UV_MIN is generated. If the "short spacing" data comes from a single-dish, this value could be set to Zero.

However, if, for example, one uses an ACA image as short spacings for an ALMA data set, the shortest baseline is the ACA antenna diameter, 7 m. SHORT_UV_MIN should then be set to 7 in this case (unless the ACA image includes the total power data in it already).

Default value is 0.

20.21.23 SHORT_WCOL

For tests only: The column of the spectra table containing the weights.

Default value: 0=3, appropriate for tables coming from CLASS\GRID command.

20.21.24 SHORT_WEIGHT_MODE

The weighting mode (NATURAL, UNIFORM or GRIDDED). It is advised to use 'NA' for Natural weighting. Other values are reserved for tests.

20.21.25 SHORT_XCOL

For tests only: The column of the spectra table containing X offsets.

Default value: 0=1, appropriate for tables coming from CLASS\GRID command.

20.21.26 SHORT_YCOL

For tests only: The column of the spectra table containing Y offsets.

```
  Default value: 0=2, appropriate for tables coming from CLASS\GRID com-
mand.
```

21 BUNDLES Language Internal Help

21.1 Language

```
COMBINE        : Combine Images with automatic resampling
EXPLORE        : Make a plot of spectra around a 2-D map
KEPLER         : Optimal line detection for Keplerian disk
SELFCAL        : Perform a self calibration
SPECTRAL_CLEAN : Multi-Spectral Spatial Deconvolution
UV_DETECT      : UV plane matched filter detection tool
```

21.2 COMBINE

```
  [BUNDLES\]COMBINE OutCube CODE In1 In2 [Off] [/FACTOR A1 A2]
[/THRESHOLD T1 T2] [/BLANKING Bval]

  or

[BUNDLES\]COMBINE OutCube = [A1*]In1 Oper [A2*]In2 [Off]
  [/THRESHOLD T1 T2] [/BLANKING Bval]
```

This is a superset of the MAP_COMBINE command, that automatically resamples and reproject the In1 and/or In2 data sets so that they match spatially (according to In1) and spectrally (when both are data cubes).

See HELP MAP_COMBINE for details on the combination codes, factors, thresholds and blanking.

21.2.1 COMBINE CODE

```
  [BUNDLES\]COMBINE OutCube CODE In1 In2 [Off] [/FACTOR A1 A2]
[/THRESHOLD T1 T2] [/BLANKING Bval]

  CODE is the operation code. Allowed values are
ADD      or PLUS    OutCube = A1*In1 + A2*In2 + Off
SUBTRACT or MINUS   OutCube = A1*In1 - A2*In2 + Off
DIVIDE   or OVER    OutCube = A1*In1 / A2*In2 + Off
MULTIPLY or TIMES   OutCube = A1*In1 * A2*In2 + Off
OPACITY             OutCube = -Log( A1*In1 / A2*In2 + Off)
INDEX               OutCube = Log( A1*In1 / A2*In2) / Log(Nu1/Nu2)
where Nu1 is the Frequency of In1, and Nu2 that of In2.  Off is 0 if not
specified.

Cube,  In1  and In2 are names of files or SIC Image variables, depending
on whether a dot (.) is present in them or not.
```

21.2.2 COMBINE /BLANKING

```
[BUNDLES\]COMBINE OutCube CODE In1 In2 [/FACTOR A1 A2]
[/THRESHOLD T1 T2] /BLANKING Bval
```

Specify the Blanking value to be used in the OutCube. If not specified, the Blanking from In1 is used instead, and that of In2 if In1 has no Blanking.

21.2.3 COMBINE /FACTOR

```
[BUNDLES\]COMBINE OutCube CODE In1 In2 /FACTOR A1 A2
[/THRESHOLD T1 T2] [/BLANKING Bval]
```

Specify the factors to apply to In1 and In2. Default is 1.0

21.2.4 COMBINE /THRESHOLD

```
[BUNDLES\]COMBINE OutCube CODE In1 In2 [/FACTOR A1 A2]
/THRESHOLD T1 T2 [/BLANKING Bval]
```

Specify the thresholds above which the computation is valid for In1 and In2. For pixels below this threshold, the OutCube is blanked.

Default is no threshold (-huge(0.)).

21.3 KEPLER

```
[BUNDLES\]KEPLER [?|DataCube|INIT|SHOW [Arg]] [/MASK File Threshold]
[/RESET [Script] [/VSYSTEM Value] [/VELOCITY R function(R)]
[/HFS File [Opacity]]
```

Re-align spectra from a rotating thin disk according to the projected rotation velocity at any point in the disk, and computes the combined integrated spectrum and brightness radial distribution. The projection assumes a thin disk (no flaring). By default, the disk is assumed to be in Keplerian rotation.

The command is controlled by a set of KEPLER_* variables, whose values can be verified by KEPLER ? and modified by the user. In addition to the built-in KEPLER_* variables, KEPLER_RMAX indicates the radius

DataCube is the name of the data cube to be used. It can a SIC 3-D variable or the name of a GILDAS (or FITS) data file. The default is CLEAN.

21.3.1 KEPLER INIT

```
KEPLER INIT
```

Declare and Initialize all KEPLER related variables. This may be needed to specify some of them before further use of the KEPLER command.

21.3.2 KEPLER SHOW

```
KEPLER SHOW Name|? [/HFS File.hfs [Opacity]]
```

or

```
SHOW KEPLER Name|?
```

KEPLER SHOW ? will list the control parameters of command KEPLER SHOW.

Command KEPLER SHOW Name (where KEPLER_Name is any of the result variable names, i.e. Name can be PV, PROFILE, SPECTRUM, or VELO) can be used to graphically display the results. KEPLER SHOW ALL will display a combined plot.

KEPLER SHOW SPECTRUM and KEPLER SHOW ALL also perform a Gaussian fit into the integrated line profile to derive the integrated flux, the disk systemic velocity (if the option /VDISK was not present) and the line width.

21.3.3 KEPLER /HFS

```
KEPLER SHOW Name /HFS File.hfs [Opacity]
```

Use an Hyperfine structure file, described in File.hfs to fit the velocity re-aligned, integrated spectrum. The corresponding hyperfine structure is stored in SIC structure HFS%. The optional argument Opacity can be used to specify the total opacity (sum over the hyperfine components), the default being 0.1. If Opacity is negative, the opacity is then adjusted by the profile fitting procedure, using -Opacity as starting value.

Fit results are available in SIC structure G_%...

21.3.4 KEPLER /MASK

```
[BUNDLES\]KEPLER /MASK [File [Threshold]] [/VSYSTEM Value] [/VELOCITY R function(R)]
```

(Experimental)

Generate an approximate spatial mask for further use, e.g. in CLEAN or other masking operations. The brightness is assumed to decrease with radius, and a local line width of 0.3 km/s is assumed to derive the channel-dependent mask. Convolution by the current Clean beam is performed.

If a File argument is present, the resulting mask data cube is stored in File (with default extension

If present, Threshold (in range [0,1]) indicates that pixels below the apparent peak brightness times the Threshold should be out of the mask (i.e. have a value 0), the others in the mask (a value 1).

If Threshold is not present, the mask values are just taken "as is", without being re-mapped to a 0,1 mask. A simple SIC formula can be used to convert them to a 0,1 mask later.

21.3.5 KEPLER /RESET

KEPLER /RESET [Script]

Deletes the result variables (see HELP KEPLER Results) and, if present, executes the specified Script file that can contain any command to reset the KEPLER_... input variables.

21.3.6 KEPLER /VELOCITY

[BUNDLES\]KEPLER [/VSYSTEM Value] /VELOCITY R Function(R)

Use the functional form specified by Function(R) for the rotation velocity as a function of Radius. The default is the Keplerian law KEPLER_VMASS*sqrt(R/100).

Note: This can allow to use the KEPLER command to handle rotation curves for galaxies, for example. However, the choice of radial units is not obvious in this case. It can be made simple by setting the "distance" KEPLER_DIST to 1 pc, so that the radii R are then just in arcsecond.

21.3.7 KEPLER /VSYSTEM

KEPLER [DataCube] /VSYSTEM [Value] [/VELOCITY R Function(R)]

Specify that radial profile should be computed at velocity KEPLER_VDISK (and optionally reset KEPLER_VDISK = Value]. No fit for the disk velocity is performed by KEPLER SHOW PROFILE after that.

21.3.8 KEPLER /WIDGET

KEPLER /WIDGET

Activate the Widget to control KEPLER command variables and actions. Still under development, so all functions may not be controlled in this way. , e.g. no HFS mode, no velocity function control yet.

21.3.9 KEPLER Variables:

Most input variables are pre-defined by the code. The exception is KEPLER_RMAX, which can be defined by the user but has a default value derived from KEPLER_ROUT otherwise.

21.3.10 KEPLER_X0

X offset of disk center (in arcsec)

21.3.11 KEPLER_Y0

Y offset of disk center (in arcsec)

21.3.12 KEPLER_ROTA

Position angle of the projection of the disk rotation axis (in degree, East from North)

21.3.13 KEPLER_INCLI

Disk inclination (in degree)

21.3.14 KEPLER_DIST

Disk distance (in pc).

21.3.15 KEPLER_VMASS

Disk rotation velocity at 100 au in km/s. Should be equal to 2.98 * sqrt(M*/Msun) from Kepler laws.

21.3.16 KEPLER_RMIN

Inner radius for the Radius-Velocity diagram (in au).

21.3.17 KEPLER_RMAX

Outer radius for the Radius-Velocity diagram (in au).
KEPLER_RMAX must be greater then KEPLER_ROUT.

KEPLER_RMAX is an optional variable. By default (i.e.
if the variable has not been created by the user),
the maximum radius is taken as KEPLER_ROUT + 5*KEPLER_STEP.

21.3.18 KEPLER_RINT

Inner radius (in au) for the integrated spectrum, computed by summing up spectra from KEPLER_RINT to KEPLER_ROUT.

21.3.19 KEPLER_ROUT

Outer radius (in au) for the integrated , computed by summing up spectra from KEPLER_RINT to KEPLER_ROUT.

21.3.20 KEPLER_STEP

Sampling step in au. Must be compatible with the angular resolution.
Note that the spectral resolution allows some gain in spatial resolution compared to the beam size.

21.3.21 KEPLER_THETA

Maximum angle (in degree) from the disk projected major axis beyond which the spectra are not considered in the averaging process.

This angle is needed because the projected rotation velocity is degenerate to zero along the minor axis, which results in a lower effective angular resolution in this direction. This effect depends on the disk inclination: a larger angle can be used for less inclined disks.

The default is 60 degree. The Azimut coverage can be visualized using command SHOW KEPLER_VELO.

21.3.22 KEPLER_AZIMUT

KEPLER_0 in the computation (Azimut Zero corresponds to the oriented minor axis, i.e. is at KEPLER_ROTA in the sky plane).

For example 90 270 indicates Az between 90 and 270 degrees, while 270 90 indicates As from 0 to 90 and 270 to 360.

The Azimut coverage can be visualized using command SHOW KEPLER_VELO.

21.3.23 KEPLER_VDISK

KEPLER_VDISK is a variable that indicates the disk systemic velocity (in km/s). The value is always used to define the velocity field (see KEPLER_VELO result variable).

However, for the radial profile, the usage depends on the /VSYSTEM option. When /VSYSTEM is not present, the radial profile is taken as the maximum brightness of the integrated, velocity aligned, spectrum for at each radius. When /VSYSTEM option is present, the radial profile is taken at the velocity KEPLER_VDISK.

21.3.24 KEPLER_STRICT

KEPLER_STRICT is an optional (user-created) logical variable that specifies whether a spectrum is computed or not when the full velocity range is not covered. Default is NO, i.e. spectra not covering the full range are not flagged.

21.3.25 Results:

The KEPLER command produces 2 SIC Tables, KEPLER_SPECTRUM, and KEPLER_PROFILE, and two SIC 2-D Images, KEPLER_PV and KEPLER_VELO that are available as SIC variables for further writing or plotting.

Command SHOW KEPLER Name (where KEPLER_Name is any of the above variable names) can be used to graphically display these variables. SHOW KEPLER ALL will display a combined plot. The SHOW KEPLER command behaviour is controlled by the variables in structure KEPLER_SHOW%.

SHOW KEPLER SPECTRUM and SHOW KEPLER ALL also perform a Gaussian fit into the integrated line profile to derive the integrated flux, the disk systemic velocity (if the option /VSYSTEM was not present in the KEPLER command) and the line width.

21.3.26 KEPLER_PROFILE

Radial profile of the peak brightness temperature. The peak brightness temperature is that appearing at KEPLER_VDISK if /VDISK option was present, or at the velocity defined by the maximum of the integrated spectrum. This is a 3 column table containing the radii (in au), the brightness (in K) and an estimate of its error.

21.3.27 KEPLER_PV

Velocity-aligned spectra as a function of radius. This is a 2-D image (equivalent to a Nrad column table) containing the spectra (averaged brightness temperature) for each of the Nrad radii defined by the sampling KEPLER_RINT, KEPLER_RMAX and KEPLER_STEP.

The velocities are defined in KEPLER_SPECTRUM[1], but also derived from the axis description of this 2-D image.

21.3.28 KEPLER_SPECTRUM

Velocity-aligned, disk-integrated spectrum, over the region defined by KEPLER_RINT, KEPLER_ROUT and KEPLER_THETA. This is a 3 column table containing the velocities (km/s) in column 1, the flux in column 2, and an estimate of the flux error in column 3.

21.3.29 KEPLER_VELO

Velocity field in the region retained by the KEPLER command. This contains the line-of-sight projected velocity offset from the (assumed) disk systemic velocity KEPLER_VDISK.

21.3.30 Display:

The results of the KEPLER command can be displayed in several ways by command KEPLER SHOW. The command behaviour is controlled by variables in the structure KEPLER_SHOW%.

21.3.31 KEPLER_SHOW

KEPLER_SHOW% is a Structure variable controlling how the display of the KEPLER SHOW (or SHOW KEPLER) command is handled. It contains the following variables, that can be listed by KEPLER SHOW ?

```
Show Mean Spectrum     KEPLER_SHOW%SPEC       [ NO ]
Show Radial Profile    KEPLER_SHOW%PROF       [ NO ]
Show PV diagram        KEPLER_SHOW%PV         [ NO ]
Select compact layout  KEPLER_SHOW%LAYOUT     [ NO ]

Velocity    Range      KEPLER_SHOW%V          [ 0 0 ]
Radius      Range      KEPLER_SHOW%R          [ 0 0 ]
Temperature Range      KEPLER_SHOW%T          [ 0 0 ]
Flux        Range      KEPLER_SHOW%F          [ 0 0 ]
```

21.3.32 KEPLER_SHOW%V

Velocity range for the displays (R-V diagram and Spectrum), in km/s.

21.3.33 KEPLER_SHOW%R

Radius range for the displays (R-V diagram and Radial Profile), in au.

21.3.34 KEPLER_SHOW%T

Temperature range for the display (R-V diagram), in K

21.3.35 KEPLER_SHOW%F

Flux range for the displays (Spectrum), in Jy.

21.3.36 KEPLER_SHOW%LAYOUT

If YES, use a compact layout. If NO use layout with equal panel sizes. This only applies to KEPLER SHOW ALL.

21.4 SELFCAL

SELFCAL [?|AMPLI|APPLY|PHASE|SUMMARY|SHOW [Last [First]] [/WIDGET]

Command to perform Self Calibration (even on spectral line data). The solution is computed, saved and/or applied.

The arguments control the action.

```
SELFCAL ?         Lists the self calibration parameters
SELFCAL AMPLI     Compute an Amplitude only self calibration
SELFCAL APPLY     Apply the computed solution
SELFCAL FLAG []   Flag data according to self calibration correction
SELFCAL PHASE     Compute a Phase only self calibration
SELFCAL SHOW []   Show the computed corrections
SELFCAL SAVE      Save self calibration parameters in selfcal.last
SELFCAL SUMMARY   Display the improvements in Noise & Dynamic range
                  and calibration status
```

21.4.1 SELFCAL /WIDGET

SELFCAL /WIDGET

Activates the widget interface to perform self calibration (even on spectral line data set). A continuum data set is extracted from the specified channel range, with all selected channels averaged to provide improved sensitivity to find a solution.

The buttons control the action.

AMPLI	Perform an Amplitude only self calibration
PHASE	Perform a Phase only self calibration
Continue	Continue execution when the script is in Pause
APPLY	Apply the solution
FLAG	Flag data according to self calibration correction
INPUT	List parameters (as in SELFCAL ?)
SAVE	Save the parameters in selfcal.last
SHOW	Show the computed corrections
SUMMARY	Display the improvements in Noise & Dynamic range

21.4.2 SELFCAL Arguments:

The arguments control the action.

SELFCAL ?	Lists the self calibration parameters
SELFCAL AMPLI	Perform an Amplitude only self calibration
SELFCAL APPLY	Apply the solution
SELFCAL FLAG []	Flag data according to self calibration correction
SELFCAL PHASE	Perform a Phase only self calibration
SELFCAL SAVE	Save the parameters in selfcal.last
SELFCAL SHOW []	Show the computed corrections
SELFCAL SUMMARY	Display the improvements in Noise & Dynamic range

21.4.3 AMPLITUDE

SELFCAL AMPLI

Compute an Amplitude only self-calibration. The integration times, SELF_TIME, should in general be significantly larger than for a Phase only self-calibration.

SELFCAL automatically adjusts the gains so that their mean is 1.0, to avoid changing the flux scale.

21.4.4 APPLY

SELFCAL APPLY [Type [Gain]]

Apply the self calibration solution. This is done only if the return status from the previous computation, SELF_STATUS, is greater than 0. SELFCAL APPLY automatically saves the parameters and results in the "selfcal-AMPLI.last" or "selfcal-PHASE.last" file, depending on the Type of solution applied.

Type is the type of solution to apply. The default is 'SELF_MODE', i.e. PHASE or AMPLI depending on the last type of self-calibration computed. Type can also take the DELAY value, where the "phase" corrections are interpreted as atmospheric path changes and scale as Frequency.

Gain is an optional gain factor (default is 1.) on the correction.

21.4.5 FLAG

```
SELFCAL FLAG [Threshold]
```

Flag data which have no valid correction, or a correction above the specified Threshold (in degrees, > 0 for PHASE, no units, > 1 for AMPLI). The mode (AMPLI, PHASE or DELAY) is taken from the default mode 'SELF_MODE'. No correction is applied however.

With the Widget interface, no Threshold can be specified.

21.4.6 INPUT

```
SELFCAL INPUT  or SELFCAL ?
```

Display SELFCAL control variables

21.4.7 PHASE

```
SELFCAL PHASE
```

Compute a Phase-only self calibration. The integration time should be short enough to correct for atmospheric errors, but large enough to obtain significant signal to noise for most antennas.

21.4.8 SAVE

```
SELFCAL SAVE
```

Save the parameters and results in the "selfcal.last" file.

21.4.9 SHOW

```
SELFCAL SHOW [Last [First]]
```

Shows the correction computed by self calibration. By default, the difference between the last two iterations is displayed: phase should be around 0, and amplitude around 1 if the solution is converged.

If Last and First are present, it shows the difference between these two specified iteration. If Last only is present, it shows the total correction between the original data and that iteration.

The displayed ranges are controlled by SELF_ARANGE[2] (limits of Amplitude gain), SELF_PRANGE[2] (limits of Phase correction) and SELF_TRANGE[2] (limits for time axis).

21.4.10 SUMMARY

SELFCAL SUMMARY

Display a summary of the Self-calibration process: type of calibration, rms and dynamic range at each iteration, as well as the number of flagged or uncalibrated visibilities.

21.4.11 Results:

SELFCAL returns results in several variables, creates a CGAINS array containing the Gain values, and one file to display the computed correction. The file name is specified by SELF_SNAME.

The result variables are:

SELF_APPLIED
  Indicates whether the solution has been applied
SELF_STATUS
  Indicates if a solution has been computed
SELF_DYNAMIC[Self_Loop+1]
  The dynamic range at each iteration
SELF_RMSCLEAN[Self_Loop+1]
  The rms noise at each iteration

21.4.12 SELF_APPLIED

Variable SELF_APLPLIED indicates whether a solution has been applied (#0) or not (0). The value indicates the type and quality of solution, as for SELF_STATUS

21.4.13 SELF_STATUS

```
Variable SELF_STATUS indicates if a solution has been computed
      0 no solution
      +/- 1   Phase solution
      +/- 2   Gain solution
    >0 means a good solution, <0 a poor one.
```

21.4.14 SELF_DYNAMIC

SELF_DYNAMIC is a variable length array of size Self_Loop+1, containing the dynamic range at each iteration.

21.4.15 SELF_RMSCLEAN

SELF_RMSCLEAN is a variable length array of size Self_Loop+1, containing the Clean map rms noise at each iteration.

21.4.16 Variables:

The SELFCAL behaviour can be adjusted through control variables named SELF_... (see EXA SELF_ for a list), in addition to the control variables of UV_MAP (MAP_...) and CLEAN (CLEAN_...)

The most important variable is SELF_TIMES, a variable length array which controls the integration time at each loop. SELF_NITER and SELF_MINFLUX also control the number of Clean components and minimum flux retained in each loop. The size of these arrays define the number of loops.

See HELP SELFCAL SELF_LOOP to find out how to control the number of loops.

21.4.17 SELF_CHANNEL

```
SELF_CHANNEL[2]
```

First and last channel to define the range to compute the UV table for the self-calibration solution. For example, if you have used UV\PLUGC to plug a continuum data into the last channel, this could be set to the number of channels. 0 0 means all channels are averaged to compute the "continuum" image.

21.4.18 SELF_COLOR

SELF_COLOR

Controls the LUT color range at each self-calibration cycle if non Zero. Since the dynamic range evolves, this can be a useful way to highlight the gain. If non Zero, SELF_COLOR is passed as argument to a COLOR command executed at each cycle after display.

Pratical values for SELF_COLOR are -8 ("bright" version, highlights the noise level) or +8 ("dark" version, hides the noise), but lower absolute values may be needed for higher dynamic ranges.

See HELP COLOR for details.

21.4.19 SELF_LOOP

Number of self-iteration loops. It is a ReadOnly variable that is automatically computed from the size of the SELF_TIMES, SELF_NITER and SELF_MINFLUX arrays.

These variable length arrays can be resized using the LET /RESIZE command. For example
LET SELF_TIMES 40 20 10 /RESIZE
will lead to an array of 3 elements, SELF_TIMES[3].

SELF_TIMES, SELF_NITER and SELF_MINFLUX must be of equal size. To simplify the process, constant arrays are assumed of arbitrary length in this determination. If all 3 arrays have constant values, SELF_TIMES determines the number of loops.

Constant arrays are assumed of arbitrary length in this determination. If all 3 arrays have constant values, SELF_TIMES determines the number of loops.

21.4.20 SELF_NITER

Variable length array specifying the number of Clean components retained for each loop of the self-calibration process. Default is 0, meaning all Clean components found by CLEAN.

For simple structures and Phase calibration, 10 may be enough. For more complex ones, be sure to include enough Clean components in the model. More components can be taken at each step, although the better knowledge of phase errors often allows the source to be represented with a smaller

number of components after self-calibration than before. The default is a reasonably good compromise, although faster convergence may be obtained with smaller number of components.

This variable length array can be resized using the LET /RESIZE command. For example
    LET SELF_NITER 10 0 0 /RESIZE
will lead to 3 self-calibration loops (if SELF_TIMES and SELF_MINFLUX are also of size 3), the first one selecting only 10 components, the two others all components.

21.4.21 SELF_TIMES

Variable length array specifying the integration time (in seconds) for each loop of the self-calibration process.

At NOEMA, 45 sec is the normal minimum integration time. Depending on Signal to Noise, you may need to have this longer, e.g. 120 sec. For several loops, start with a longer value, and decrease only at the end.

At ALMA, the minimum integration time is 6 sec. At VLA, this may be as small as 1 sec.

It is recommended to use the same integration time for the last two iterations, to simplify the interpretation of the results and of the SELFCAL SHOW display.

This variable length array can be resized using the LET /RESIZE command. For example
    LET SELF_TIMES 180 90 45 /RESIZE
will lead 3 self-calibration loops (if SELF_TIMES and SELF_MINFLUX are also of size 3) with decreasing integration times.

21.4.22 SELF_MINFLUX

Variable length array specifying the minimum flux density (in Jy/beam) to be considered in the Clean image for each loop of the self-calibration process. Note that this is the brightness of a pixel, not the flux of a Clean component.

This variable length array can be resized using the LET /RESIZE command. For example
    LET SELF_MINFLUX 0.001 0 0 /RESIZE
will lead 3 self-calibration loops (if SELF_TIMES and SELF_NITER are al-

so of size 3) selecting on regions brighter than 1mJy/beam in the first one, and all regions in the last 2 ones.

21.4.23 SELF_REFANT

The reference antenna number. If 0, the program will peak the one with the shortest average baselines.

21.4.24 SELF_FLUX

Maximum value in the FLUX window. If 0, the FlUX window of Clean is not displayed

21.4.25 SELF_PRECISION

Tolerance to test for self-calibration convergence. The default is 0.01. SELFCAL SUMMARY will write a message when no more selfcal iteration improves the solution (noise and dynamic range) at this precision level.

21.4.26 SELF_RESTORE

Use UV_SELF /RESTORE after Cleaning. This avoids signal aliasing at image edges, and leads to a better estimate of the noise level.

It also allows to use smaller images, in practice,

21.4.27 SELF_DISPLAY

If YES, display Clean image before each calibration loop, and prompt for user input. If YES, Cleaning at each step will use the number of iterations specified by NITER, while if set to NO, Cleaning will stop at SELF_NITER, saving time.

The dynamic range progress report is accurate only if SELF_DISPLAY is set.

21.4.28 SELF_FLAG

If SELF_FLAG is YES, SELFCAL will flag data with no solution. If NO, it will keep the data as it was before.

21.4.29 SELF_SNR

Minimum Signal to Noise ratio on the antenna gain to consider a solution to be valid for an antenna. 6 is a good value, 3 a lower limit. Beware that this SNR makes sense only if the a priori estimate of the noise from the UV weights is correct: see SELF_SNOISE.

21.4.30 SELF_SNOISE

Noise scaling factor. This should be 1, but some noise estimates need corrections. Continuum data from ALMA often requires sqrt(2) instead, for example. Command UV_PREVIEW may help you checking the noise scale.

21.4.31 CLEAN_ARES

Maximum absolute residual to stop Cleaning

21.4.32 CLEAN_FRES

Maximum fractional residual to stop Cleaning

21.4.33 CLEAN_NITER

Maximum number of Clean components. If all of CLEAN_ARES, CLEAN_FRES and CLEAN_NITER are 0, Clean stops by checking the stability of the cleaned flux over CLEAN_NKEEP iterations. See HELP CLEAN for details.

21.5 SPECTRAL_CLEAN

[BUNDLES\]SPECTRAL_CLEAN MODE [Control]

Perform a Clean with different spectral scales.

MODE can be
FFT Clean the Fourier Transform of the spectra
WAVE Clean a Wavelet Transform of the spectra
DUAL Ns Clean at two different spectral resolutions
MULTI Ns Clean at two (or more) spectral resolutions.

Instead of Cleaning separately individual channels, SPECTRAL_CLEAN cleans either Spectral Transform of the data cube (along the velocity axis), and performs the inverse transformation to restore the channel-based Clean components.

The SPECTRAL_CLEAN method can improve deconvolution of spectrally well resolved faint line wings, yielding more accurate integrated line fluxes, at the cost of some extra computations. However, SPECTRAL_CLEAN can be hampered by strong velocity gradient, that require supports that differ widely among channels.

Standard CLEAN control parameters apply. However the METHOD must allow retrieving Clean Componentes: MRC is thus excluded. SPECTRAL_CLEAN can also work on Mosaics.

21.5.1 SPECTRAL_CLEAN DUAL

[BUNDLES\]SPECTRAL_CLEAN DUAL NSmooth

In DUAL mode, SPECTRAL_CLEAN cleans a (spectrally) smoothed version of the data cube, and the difference between this smooth version and the original ones. The two Clean component tables are then merged to restore the final image using UV_RESTORE.

NSmooth is the number of channels being smoothed together.

The DUAL methods involves Cleaning two data cubes, so is a somewhat slower than a simple CLEAN. SPCLEAN%MEMORY indicates whether intermediate results are kept in memory (faster) or on temporary files (slower, but allowing to work on larger files).

Mode MULTI should in general be preferred, but does not include the final UV_RESTORE.

21.5.2 SPECTRAL_CLEAN FFT

[BUNDLES\]SPECTRAL_CLEAN FFT

Clean the Fourier Transform along the spectral axis of the DIRTY cube, and transform back the Clean components to retrieve the per-channel Clean components. The final Clean image is produced using UV_RESTORE.

The usual Clean convergence parameters apply, although CLEAN_NKEEP may need small adjustment because of the non-positive nature of the signal.

The Fourier Transform leads to a Complex valued data set, whose Hermiticity is not considered. Real and Imaginary parts are cleaned separately, so the method is about twice slower than a standard Clean.

21.5.3 SPECTRAL_CLEAN MULTI

[BUNDLES\]SPECTRAL_CLEAN MULTI Ns1 [Ns2 ...]

Clean a (spectrally) smoothed version of the data cube, then start from this result to continue Cleaning with the original spectral resolution.

To be implemented: the method can be generalized to more than 1 smoothing scale, by performing the smoothing in descending order (Ns1 > Ns2 > ... > 1). This may allow to catch signals that have widely different widths.

21.5.4 SPECTRAL_CLEAN WAVE

[BUNDLES\]SPECTRAL_CLEAN WAVE

Clean a Wavelet Transform along the spectral axis of the DIRTY cube, and transform back the Clean components to retrieve the per-channel Clean components. The final Clean image is produced using UV_RESTORE.

The noise level is not the same for the wavelet components of the data cube: this is taken into account by the method to Clean each wavelet image plane to its proper noise level.

Apart for the time required by the Wavelet transform, the method has a comparable speed to a standard Clean.

21.6 UV_DETECT

[BUNDLES\]UV_DETECT [Result] [/FILE UVData ImageData]

[BUNDLES\]UV_DETECT [Result]

Apply a matched filter defined by the CLEAN image to the current UV

data and save results as above. It is recommended that the continuum emission has been removed before, e.g. using UV_BASELINE.

Result.uvt will be a pseudo UV table with only the (0,0) visibility, which can be viewed by UV_PREVIEW.

Result.dat is a column data file, useable in GreG by command COLUMN. Column 1 is the velocity, 2 the frequency, 3 the filter intensity.

The default for argument Result is gag_scratch:detect.

21.6.1 UV_DETECT /FILE

[BUNDLES\]UV_DETECT [Result] /FILE UVData[.uvt] ImageData[.lmv-clean]

Apply a matched filter defined by the image model available in file ImageData (default extension .lmv-clean) to the UV data set available in file UVData (default extension .uvt), and save the filtered result into files Result.uvt and Result.dat. Continuum emission is automatically removed using UV_BASELINE, with line windows defined by UV_PREVIEW.

If argument Result is not present, the intermediate files are stored in
gag_scratch::detect.uvt and gag_scratch:detect.dat

21.6.2 UV_DETECT Algorithm

The UV_DETECT algorithm is basically that described by Loomis et al 2018 (2018AJ....155..182L, see https://arxiv.org/pdf/1803.04987.pdf). Here, instead of being implemented as a Python / CASA script, it is coded in Fortran and embedded into IMAGER. UV_DETECT calls an imager script that reads the data set, computes a UV_MODEL from the ImageData using the UV coverage found in the UV data set, resamples this UV_MODEL at the same velocity resolution than the UV data, and ultimately calls command UV_CORRELATE to apply the UV plane filter.

As described in the original paper, the UV data set and source model available in ImageModel must have the same centering. The ImageModel must also cover a sufficient width around its specified source velocity.

The command will complain if the smoothing kernel described by the ImageModel is insufficient.

22 IMAGER Language Internal Help

22.1 Language

```
CLARK       Shortcup to CLEAN with CLARK Method
HOGBOM      Shortcup to CLEAN with HOGBOM Method
MRC         Shortcup to CLEAN with MX Method
MULTISCALE  Shortcup to CLEAN with MULTISCALE Method
PIPELINE    Pipeline processing of all UV tables
SDI         Shortcup to CLEAN with SDI Method
TIPS        Give Random tip about IMAGER
```

22.2 TIPS

```
  IMAGER\TIPS [Number]
      Without argument, print out a random tip about IMAGER.
With an argument, give the corresponding tip number.
```

22.3 PIPELINE

```
      IMAGER\PIPELINE [Arg1 [Arg2]] [/MODE Type] [/WIDGET]

   PIPELINE  is a command activating the Imaging Pipeline. The pipeline
contains all processing steps for high fidelity  imaging  (Self-calibra-
tion, continuum extraction, line identification, etc...)

   With  no  /WIDGET  option,  PIPELINE  run  in  unattended  mode  (no
SIC\PAUSE statement), one step after the other.
   PIPELINE ?      will display the Pipeline input parameters
   PIPELINE *      will run the whole Pipeline with the current input param
   PIPELINE NEXT   runs the next following step
   PIPELINE LAST   will repeat the last step.
PIPELINE with no argument stands for PIPELINE  LAST.  The  /MODE  option
controls the operating mode (default: last selected mode).
```

22.3.1 PIPELINE /MODE

```
      IMAGER\PIPELINE [Arg1 [Arg2]] /MODE All|Continuum|Split|Survey

Specifdy in which mode the Pipeline will run. 4 modes are possible

    CONTINUUM
  Only  produce  continuum images, no data cube. In this mode, the emis-
sion is assumed to be purely continuum, with no significant line contam-
ination.
```

SURVEY

Only produce line+continuum data cubes. No attempt to separate line emission from continuum emission is made. The complete spectral coverage is imaged, at the velocity resolution specified by the user. This mode is especially adapted for extra-galactic work, or sources with lines close to the spectral confusion limit.

ALL

Produce data cubes that contains line and continuum emission together. Spectral line identification will be performed if a Catalog is present, with the user-specified velocity range and spectral resolution around each spectral line in the band. If no Catalog is present, the complete spectral coverage is imaged, at the velocity resolution specified by the user.

SPLIT

Similar to ALL, but produce data cubes that contains line and continuum emission separately. The separation is based on filtering performed by UV_PREVIEW.

22.3.2 PIPELINE /WIDGET

PIPELINE /WIDGET

Activates the widget that simplifies the interactive control of the Pipeline steps.

Each step is controlled by a separate button. In this mode, the Self-Calibration related button requires user interaction. Some buttons are duplicated on top of the widget, which also contains a SELF-CALIBRATION button that can execute all Self-Calibration related steps in sequence.

The PIPELINE button (as the GO button) will run the full Pipeline without any further user interaction, like the PIPELINE * command.

22.3.3 PIPELINE Arguments:

PIPELINE supports the following arguments when not used in Widget mode, given here in sequential order of actions

ORGANIZE
FIND
SELECT
CHECK

```
  COMPUTE  (or SELF)
  COLLECT
  APPLY
  TIME
  TABLES
  IMAGE
  SHOW
  SAVE
```

In addition argument SKY can be used to apply primary beam correction to existing images.

22.3.4 ORGANIZE

IMAGER\PIPELINE ORGANIZE

Automatically re-arrange the set of UV tables or UVFITS files in the current directory to a directory tree that is suitable for the Pipeline / Widget use.

This is Step 1, NEXT step is FIND

22.3.5 FIND

IMAGER\PIPELINE FIND

Finds the UV tables in the directory tree and scan their properties for further use

This is Step 2, NEXT step is SELECT

22.3.6 SELECT

IMAGER\PIPELINE SELECT

Select the UV tables to be processed according to the Filter (all%filter) and identify the Wide bands and Narrow bands

This is Step 3, NEXT step is CHECK

22.3.7 CHECK

IMAGER\PIPELINE CHECK

Check whether the continuum is strong enough to require and allow self-calibration.

This is Step 4, NEXT step is COMPUTE (or its alias SELF)

22.3.8 COMPUTE

IMAGER\PIPELINE COMPUTE or IMAGER\PIPELINE SELF

Compute the Self-calibration solution from the Wide bands UV tables

This is Step 5, NEXT step is COLLECT

22.3.9 COLLECT

IMAGER\PIPELINE COLLECT

Improve self-calibration solution by interpolating correction delays between different wide bands. This can increase S/N of self-calibration on weaker sources, but may degrade solutions with very high S/N per band on strong sources.

This is Step 6, NEXT step is APPLY

22.3.10 APPLY

IMAGER\PIPELINE APPLY

Apply self-calibration solution derived previously to all Wide and Narrow band UV tables. The selected solution is from the Wide band that is closest in Frequency, using the DELAY mode for the solution.

This is Step 7, NEXT step is TIME

22.3.11 TIME

IMAGER\PIPELINE TIME

Time average the self-calibrated UV tables to save space and further time processing

This is Step 8, NEXT step is TABLES

22.3.12 TABLES

IMAGER\PIPELINE TABLES

Extract UV tables around each spectral line available in the CATALOG and covered by the frequency setup. Continuum emission is (empirically) separated from Line emission using the UV_PREVIEW, UV_BASELINE / UV_FILTER mechanism.

This step is only needed when direct image analysis in the UV plane is required.

This is Step 9, NEXT step is IMAGE

22.3.13 IMAGE

IMAGER\PIPELINE IMAGE

Produce deconvolved images around each spectral line available in the CATALOG and covered by the frequency setup. Continuum emission is (empirically) separated from Line emission using the UV_PREVIEW, UV_BASELINE / UV_FILTER mechanism, and the continuum images are also produced.

This step does not use the result of the PIPELINE TABLES step, but only the self-calibrated UV tables produced by PIPELINE APPLY and PIPELINE TIME.

This is Step 10, NEXT step is SHOW

22.3.14 SHOW

IMAGER\PIPELINE SHOW

Show some plots about Self-calibration quality

This is Step 11, NEXT step is SAVE

22.3.15 SAVE

IMAGER\PIPELINE SAVE

Save PIPELINE input parameters into ./all-memory.ima, so that PIPELINE * will redo the whole processing with the these parameters. The script can be edited if any correction is required (e.g. changing image size, robust parameter, etc...)

An implicit SAVE is done after PIPELINE IMAGE

22.3.16 SKY

IMAGER\PIPELINE SKY Apply primary beam correction to the Clean images in the all%maps (default MAPS) directory to produce the Sky brightness images (.lmv-sky). This steps does nothing when processing Mosaics, since Clean directly restores Sky brightness images for Mosaics.

22.3.17 NEXT

IMAGER\PIPELINE NEXT

Run the next step of the Pipeline.

22.3.18 LAST

IMAGER\PIPELINE LAST

Repeat the last executed step of the Pipeline (presumably after changing some control parameters)

22.3.19 Variables

Apart from the usual imaging parameters, PIPELINE is controlled by a few variables.

```
ALL%CATALOG   indicates the spectral line catalog (set by the CATALOG comm
ALL%DROP              Number of edge channels drop at each edge
ALL%FILTER    indicates the files to be processed
ALL%MINFRES   is a threshold in spectral resolution to separate continuum-
ALL%RANGE     gives the velocity range and resolution
```

In addition, the self-calibration step is controlled by ALL%PHASE_NITER, ALL%PHASE_TIMES, ALL%AMPLI_NITER and ALL%AMPLI_TIMES that specify the number of Clean components and integration times for Phase and Amplitude respectively

22.3.20 ALL%CATALOG

Name of spectral line catalog. This is a Read-Only variable, defined using the CATALOG command. See HELP CATALOG for details.

If a line catalog is selected, the Pipeline scripts will produce one image for each spectral line falling into the observed "high resolution" spectral windows, over the specified velocity range ALL%RANGE.

If no line catalog is selected, all channels of all "high resolution" spectral windows will be imaged. No velocity range is used in this case.

"Low resolution" spectral windows will be used to produce continuum only images. The distinction between "Low" and "High" resolution spectral windows is made from variable ALL%MINFRES

22.3.21 ALL%DROP

INTEGER ALL%DROP[2] This is an integer array giving the number of edge channels ignored when reading a UV table. Edge channels may have different weights than the other ones, especially for ALMA data because of the off-line Doppler correction.

Such low weights edge channels are automatically dropped for Single fields, so you can use ALL%DROP = 0, but not yet for Mosaics, where ALL%DROP = 5 is more appropriate.

22.3.22 ALL%FILTER

CHARACTER*256 ALL%FILTER

File filter to be used to select the relevant UV tables.

This filter MUST NOT contain the .uvt extension. The default filter is *, but a specific file name can be given to process only one UV table instead of all of them (or all tables for a given source for example).

22.3.23 ALL%AMPLI_NITER

ALL%AMPLI_NITER is a size-variable integer array of rank 1.

It indicates the number of clean components for each iteration of Amplitude self-calibration. The number of iterations is defined by its number of elements. The size of this array can be changed using the LET /RESIZE option.

0 means to select all Clean components.

22.3.24 ALL%AMPLI_TIMES

ALL%AMPLITIMES is a size-variable real array tof rank 1.

It indicates the integration time for each iteration of Phase self-calibration. The number of iterations is defined by its number of elements. The size of this array can be changed using the LET /RESIZE option.

Typical values are 90 to 180 seconds for NOEMA, 60 seconds for ALMA.

22.3.25 ALL%COLLECT

Real ALL%COLLECT

Stability Threshold for flagging data after Self-Calibration.

For Self-calibration solutions with more than one iteration, the solution is flagged if the difference between the last two iterations exceeds Threshold times the expected phase error due to noise.

The default is 0, a convention indicating to ignore this flagging step.

When the noise level is appropriately estimated, a Threshold of 3 is a good value to suppress unstable solutions that reflect some issues in the self-calibration model.

22.3.26 ALL%COMBINE

Logical ALL%COMBINE

Indicates whether the COLLECT step (that derives an atmospheric delay correction from the per-band phase correction) should be performed or not for the Self-Calibration in PIPELINE mode.

22.3.27 ALL%ITIME

Real ALL%ITIME

Integration time used for the TIME compression step. 0 means that IMAGER will derive the best time given the baseline lengths and field of view.

22.3.28 ALL%MINFRES

REAL ALL%MINFRES

Maximum frequency resolution under which a UV Table is considered as a spectral line data. UV Tables with coarser spectral resolution are treated as continuum only data: the line emission is filtered out, and a pure continuum image is generated using bandwidth synthesis in this case.

22.3.29 ALL%PHASE_NITER

ALL%PHASE_NITER is a size-variable integer array of rank 1.

It indicates the number of clean components for each iteration of Phase self-calibration. The number of iterations is defined by its number of elements. The size of this array can be changed using the LET /RESIZE option.

22.3.30 ALL%PHASETIMES

ALL%PHASE_TIMES is a size-variable real array of rank 1.

It indicates the integration time for each iteration of Phase self-calibration. The number of iterations is defined by its number of elements. The size of this array can be changed using the LET /RESIZE option.

Typical values are 45 seconds for NOEMA data, 6 to 24 sec for ALMA data.

22.3.31 ALL%RANGE

REAL ALL%RANGE[3]

Velocity sampling around each spectral line (in km/s). ALL%RANGE[1:2] give the Min and Max velocities ([0,0] indicates to take the whole coverage of the sub-band).

ALL%RANGE[3] gives the desired spectral resolution (0 indicates to use the spectral resolution of the sub-band).

22.3.32 ALL%SPLIT

LOGICAL ALL%SPLIT

```
  Separate (Split) Line emission from Continuum emission in  data  Cubes
or extracted uv Tables, or keep Line+Continuum data together.
```

22.4 HOGBOM

```
      IMAGER\HOGBOM [First Last] [/FLUX Min Max]

is a short cut for
    LET METHOD = HOGBOM
    CLEAN [First Last] [/FLUX Min Max]
```

22.5 CLARK

```
      IMAGER\CLARK [First Last] [/FLUX Min Max] [

is a short cut for
    LET METHOD = CLARK
    CLEAN [First Last] [/FLUX Min Max]
```

22.6 MRC

```
      IMAGER\MRC [First Last]

is a short cut for
    LET METHOD = MRC
    CLEAN [First Last]
```

22.7 MULTISCALE

```
      IMAGER\MULTISCALE [First Last] [/FLUX Min Max]

is a short cut for
    LET METHOD = MULTI
    CLEAN [First Last] [/FLUX Min Max]
```

22.8 SDI

```
      IMAGER\SDI [First Last] [/FLUX Min Max]

is a short cut for
    LET METHOD = SDI
    CLEAN [First Last] [/FLUX Min Max]
```

23 Bugs and Release Notes

This section lists known bugs (or strange features) that are known at the date of release of the documentation, and the list of changes since previous releases. A more complete list is available in the IMAGER Bugs & Release note documentation here for the PDF version or there for the Web version

Notes since Documentation release of Version 3.2, dated Jan, 30th, 2022

23.1 Recent changes in repository

This section gives the changes made since the last documentation release (as dated above). The changes are available in the working repository, but may not be distributed in any standard Gildas release yet.

- 15-Nov-2022 Add the /MODE option to the PIPELINE command, with 4 modes: CONTINUUM, SPLIT, ALL or SURVEY.
- 14-Nov-2022 Debug use of MAP_CENTER variable in various commands (for Mosaics, UV_SELF , UV_SHIFT).
- 08-Nov-2022 Reorganize commands in languages:EXTRACT, SLICE and POPUP are now part of the DISPLAY\ language
- 05-Sep-2022 Add the /ASYMMETRIC option of MAP_SMOOTH and UV_SMOOTH
- 05-Jul-2022 Add the MAP_POLAR command This corresponds to revision 3.4-1 of the Documentation.
- 29-Jun-2022 Ensure READ handles properly Random Frequency Axis in UVFITS format
- 29-Jun-2022 Ensure UV_EXTRACT works with Random Frequency Axis
- 22-Apr-2022 Improve UV_PREVIEW command
- 12-Apr-2022 Support for command STOKES on internal buffers.
- 23-Mar-2022 Introduce the IMAGER\ language, and the PIPELINE command. This corresponds to revision 3.3 of the Documentation.

23.2 Known Bugs

This section contains the list of bugs known as of April 1, 2023.

- Discovered 15-Nov-2022 UV_RESTORE in Mosaic mode leads to subsequent crashes if the channel range is restricted by UV_MAP because a single beam cannot be applied to all channels.
  This is often the case for ALMA data, because of the shift implied by the Doppler correction when creating the UVFITS files. A work around is to manually drop the edge channels by reading a coherent range of channels, and over-writing the UV data.
- Corrected 08-Sep-2022 Bug correction in EXTRACT command.
- Corrected 29-Jun-2022 STOKES command had incorrect derivation from H-V polarization

- Corrected 09-Jun-2022 Debug UV_FLAG command - Allow to use without Display.
- Corrected 13-Apr-2022 MAP_COMBINE and COMBINE were not supporting big files.
- Corrected 12-Apr-2022 UV_RESIDUAL was not working on a non-sorted Mosaic data set. This could only happen by reading the Clean Component Table after reading the UV data, without any imaging step.
- Corrected 09-Mar-2022 CCT_MERGE Concatenation was improper
- Corrected 07-Mar-2022 Beam size derivation was incorrect when an ANGLE was specified
- Corrected 08-Feb-2022 UV_RESTORE was not using fixed beam size.
- Corrected 18-Jan-2022 SINGLEDISH data was incorrectly read, probably since 20-Sep-2021 (v1.53 of read.f90). Improper values were thus used by the UV_SHORT command (Discovered 17-Jan-2022 only).
- Corrected 12-Jan-2022 Mosaic offsets were improperly computed by command UV_FIELDS since 29-Jul-2021 (bug reported 10-Jan-2022). The script @ fits_to_uvt was thus leading to wrong pointing offsets.
- Corrected 10-Jan-2022 A improper interaction between scripts was causing strange behaviour when switching between EXPLORE and INSPECT_3D commands.
- Corrected 06-Jan-2022 The SIC command DEFINE IMAGE failed on large FITS files. This prevented SHOW and VIEW to work directly from these files.
- Corrected 18-Dec-2021 UV_TIME produced incorrect results on tables with different weights per channel (e.g. those produced by UV_MERGE /MODE CONCATENATE).
- Corrected 13-Dec-2021 Remove occasional infinite loop in MAP_CONTINUUM with method GAUSS (discovered 01-Dec-2021)
- Corrected 18-Nov-2021 UV_REWEIGHT syntax bug

23.3 Functional changes from previous versions

Changes in Version 3.3

- 23-Mar-2022 Introduce the IMAGER\ language. Implement the PIPELINE command there, and some miscellaneous (previously user-defined) commands like TIPS and POPUP

Changes in Version 3.2

- 18-Mar-2022 Add the CLEAN /RESTART capability
- 10-Mar-2022 Improve dirty beam related commands behaviour: replace MAP_BEAM_STEP by BEAM_STEP, use BEAM_SIZE to specify beam shape.
- 07-Mar-2022 Allow WRITE BEAM /APPEND in all cases
- 08-Feb-2022 Implement the CLEAN_STOP syntax in CLEAN
- 10-Jan-2022 Implement the LOAD command in VIEWER.

- 07-Jan-2022 Implement SHOWPV and improve transposed data display.
- 06-Jan-2022 Allow direct READ of FITS data cubes.
- 03-Jan-2022 Improve syntax of COMBINE
- 17-Dec-2021 Improve UV_CHECK BEAM and FIT commands.
- 16-Dec-2021 Add the UV_MERGE /MODE CONCATENATE command to stitch adjacent or overlapping spectral windows. Imaging is in general only possible using BEAM_STEP = 1 after this, so far.
- 13-Dec-2021 Implement a fast, reliable version of "statcont" through the MAP_CONTINUUM command (run time $<$ 40 secs for a 28 Gbyte data set on a 64-core machine)
- 13-Dec-2021 Globally revised the Robust weighting scheme. Speed up by huge amount for large (¿ 10 MegaVis) number of visibilities. UV data sorting not required anymore.
- 08-Dec-2021 Change the CASA - GILDAS interface. Use pure-Python scripts and put them in the `$HOME/.casa` directory to have a CASA-version independent solution.
- 07-Dec-2021 A missing initialization was causing UV_RESTORE to use the slow mode on some compiler versions.
- 06-Dec-2021 Speed up (by a factor 5-10) the Natural weighting scheme when there is only 1 channel.
- 19-Nov-2021 Create the VIEWER program, that only contains the DISPLAY\ language.
- 19-Nov-2021 Expand the COMMAND ? Key mechanism to many commands.
- 18-Nov-2021 Add the STATISTIC /UPDATE andFIT /JVM_FACTOR commands

A IMAGER versus CASA

A.1 Imaging Philosophy and Data Architecture

CASA intends to solve the *Measurement Equation*, whatever the complexity of this process. It is a all-in-one package for this purpose, where calibration and imaging are deeply intermixed and use a unified data format. As a result, a CASA Measurement Set is a complex architecture encompassing relations between many components stored as Tables in a directory-like tree. It can handle calibrated data, calibration tables, multisource data sets, raw data and final images in the same architecture, allowing to retain all information to process complex images, such as multi-frequency synthesis of polarized emission observed in a mosaic of fields.

On the contrary, GILDAS is designed to break the process in totally independent steps. For IRAM data, calibration is done in one program (CLIC for interferometry with NOEMA or CLASS for single-dish with the 30-m), and imaging in another (here IMAGER), with a clear intermediate step to change from the calibration data format (using the CLASSIC container) to UV Tables or CLASS Tables. In particular, IMAGER does not handle polarization transparently at the current time: polarization states must be imaged independently. The Gildas Data Format stores a limited number of informations in a single binary data file with a compact binary header, and is only suited for calibrated data. UV Tables are little more elaborate than simple images, but still limited in complexity.

A.2 Frequency and Velocity scales

In the ALMA use of CASA, all observations are kept in the Observatory frequency frame, and only converted to a celestial reference frame (such as the Local Standard of Rest, `LSRK`) at later stages, during imaging if requested.

GILDAS is more analysis oriented, and contains a dual interpretation of the frequency axis. This axis can either be interpreted with a Velocity scale, usually in the `LSRK` frame relative to a spectral line rest frequency, or as a Rest Frequency, with the astronomical source velocity specified. The choice of representation depends on the astronomer's science objective: astronomical object study (in which case the Velocity representation is more appropriate) or chemical composition study (in which case the Rest Frequency representation is preferred).

A.3 Transfering UV data from CASA

The basic idea of data transfer at this stage is to transfer a whole spectral window to GILDAS, and use the simpler and faster tools available in IMAGER for extracting channels, resampling, subtracting continuum, etc... rather than doing that in CASA.

A.4 In CASA

The steps in CASA involve

1. Separating spectral windows and different sources into independent (temporary) MS
2. Getting rid of flagged data
3. Setting the velocity reference frame and correcting for Doppler motions
4. Exporting to UVFITS

5. Removing the intermediate MS

The intermediate MS is created using *mstransform*, with `keepflags=False, regridms=True, outframe='LSRK'`. Source and spectral window identification is done using keywords `spw` and `field`, and an appropriate rest frequency is specified (in MHz) using `restfreq`. Finally, *exportuvfits* is used on the simple intermediate MS. IMAGER provides to CASA a tool named *casagildas()* that scans the content of the Measurement Set (using *listobs()* to automatically do this on all combinations of spectral windows and sources.

A.5 In IMAGER

The script `@ fits_to_uvt` handles the conversion from UVFITS to *uv* table format. The steps in GILDAS involve

1. Converting UVFITS to UVT
2. Collapsing the polarization information
3. Adjusting the weights to properly estimate the noise
4. Identify and flag bad data
5. Setting the frequency and velocity references

It also handles some nasty details, like the source coordinates being hidden in different places depending on the CASA version.

- Step 1
  `fits 'name'.uvfits to 'name'.uvt`
  will create a GILDAS UV table from the UVFITS file. The signal is assumed to be unpolarized at this stage. With the /STOKES option, polarization information would be carried on properly at this stage.
- Step 2 This step is only needed with the /STOKES option. It could be done manually by using command `STOKES` that handles Stokes parameter conversion. The desired polarization state should be set to `NONE` to optimize signal to noise ratio for unpolarized sources, to `I` if polarization is a concern.
- Step 3 This step is only carried on if the /WEIGHT option is present. Depending on CASA version, the weights are not handled in the same way. In Casa 3.4, they are approximately correct. In Casa 4.1, they are off by a large factor (perhaps the number of channels...). Task `uv_noise` utilizes the many channels available (3840 per spectral baseband in FDM mode) to compute a statistic per visibility, and adjust the weight through a median scaling factor. Task `uv_noise` will also flag data with "unusual" weights (deviating from the median by more than some factor (user specified, default 3)).
- Step 4 Task `uv_trim` will remove the flagged data from the UV table, saving space.
- In practice, the optional Steps 2 to 4 are handled by a single task called `uv_casa`, which saves 2 intermediate files (and thus 2 read and 2 writes of large files).

- Step 5
  At this stage, one could start usual imaging using the standard imaging commands `READ UV 'name'; UV_MAP`. Commands `SPECIFY FREQUENCY Value` and `SPECIFY VELOCITY Value` will set the desired correspondence between Velocity and Frequency axis.

A.6 Transfering Image data

In the Quality Assessment step 2 (QA2), the ALMA ARC staff usually provides one or more deconvolved images as FITS files. These FITS files can readily be converted into GILDAS images with the SIC command FITS:
`FITS 'name'.fits TO 'name'.gdf`
They will however have a frequency axis labelled in FREQUENCY, while GILDAS usually works with this axis labelled in VELOCITY. They can also be accessed as SIC Image variables using command `DEFINE IMAGE`

In IMAGER, direct display of these FITS files is normally possible with commands `SHOW` and `VIEW`, or other `DISPLAY\` commands.

B Properties of the Fourier Transform

Let us name $f(x)$ a function, and $F(X)$ its Fourier transform. We use here the simple, non-unitary convention

- Definition: $F(X) = \int_{-\infty}^{+\infty} f(x) e^{-2i\pi x X} dx$
- Linearity: $h(x) = af(x) + bg(x)$, then $H(X) = aF(X) + bG(X)$
- Translation: $h(x) = f(x - x_0)$, then $H(X) = e^{-2i\pi x_0 X} F(X)$
- Shifting: $h(x) = e^{2i\pi x X_0} f(x)$, then $H(X) = F(X - X_0)$
- Scaling: $h(x) = f(ax)$, then $H(X) = \frac{1}{|a|} F(\frac{X}{a})$ (the so-called *time reversal* property is obtained with $a = 1$
- Conjugation: if $h(x) = \bar{f}(x)$, then $H(X) = \bar{F}(-X)$
- Integration: With $X = 0$ in the definition
  $F(0) = \int_{-\infty}^{+\infty} f(x) dx$
- Convolution: $h(x) = f(x) * g(x)$ then $H(X) = F(X)G((X)$.
  The Fourier Transform of a product of two functions is the convolution of the Fourier Transforms of the functions.
- Uncertainty principle: the more concentrated $f(x)$ is, the more spread out its Fourier transform $F(X)$ must be. In particular, the scaling property of the Fourier transform may be seen as saying: if we squeeze a function in x, its Fourier transform stretches out in X. It is not possible to arbitrarily concentrate both a function and its Fourier transform.

The definition, illustrated here with scalars, also holds for x, X being 2-D vectors in Euclidean space, the product in the definition being a standard dot product of these vectors.

C Revision History

This section contains a verbatim revision history taken from the IMAGER code. Except for minor typos or cosmetic changes, each modification in the code is documented by 1 line of code that includes three informations:

- The language revision number of the affected command
- the date of the revision
- a short description of the change

IMAGER lists the revision numbers of all its languages at start of the program. This information can be useful for users encountering a suspected bug or strange behaviour, allowing him/her to see whether the problem has been handled since the version of IMAGER they use.

C.1 DISPLAY Language

C.2 CLEAN Language

C.3 ADVANCED Language

C.4 CALIBRATE Language

C.5 BUNDLES Language

C.6 IMAGER Language

References

Clark, B. G. 1980, A&A, 89, 377

Guilloteau, S. 2000, in IRAM Millimeter Interferometry Summer School, ed. A. Dutrey

Högbom, J. A. 1974, A&A suppl., 15, 417

Jorsater, S. & van Moorsel, G. A. 1995, AJ, 110, 2037

Pety, J., Gueth, F., & Guilloteau, S. 2001a, ALMA+ACA Simulation Results, Alma memo 387, IRAM

Pety, J., Gueth, F., & Guilloteau, S. 2001b, ALMA+ACA Simulation Tools, Alma memo 386, IRAM

Pety, J., Gueth, F., & Guilloteau, S. 2001c, Impact of ACA on the Wide-Field Imaging Capabilities of ALMA, Alma memo 398, IRAM

Schwab, F. R. 1984, AJ, 89, 1076

Steer, D. G., Dewdney, P. E., & Ito, M. R. 1984, A&A, 137, 159

Wakker, B. P. & Schwarz, U. J. 1988, A&A, 200, 312

Index

ALMA, 115
APPLY, 51, 52, 191
    /FLAG, 55, 192
    AMPLI, 191
    DELAY, 55, 192
    PHASE, 192

BUFFERS, 115
    AGAINS, 116
    BEAM, 116
    CCT, 116
    CGAINS, 116
    CLEAN, 116
    CLIPPED, 116
    CONTINUUM, 117
    DIRTY, 117
    EXTRACTED, 117
    FIELDS, 117
    M_AREA, 117
    M_PEAK, 117
    M_VELO, 118
    M_WIDTH, 118
    MASK, 117
    PRIMARY, 118
    RESIDUAL, 118
    SHORT, 118
    SINGLE, 118
    SKY, 118
    SLICED, 119
    SPECTRUM, 119
    UV, 119
    UV_FIT, 119
    UVCONT, 119
    UVRADIAL, 119
    WEIGHT, 119

casagildas, 18
CATALOG, 6, 62, 67, 71, 75, 79, 86
    Astro, 87
    Default, 87
    Linedb, 87
    LINEDB%ENERGY, 88
CCT_CLEAN, 119
CCT_CONVERT, 120
CCT_MERGE, 120
CLARK, 9, 27, 266
CLEAN, 5, 9, 33, 36, 51, 55, 58, 59, 120
    /ARES, 122
    /FLUX, 121
    /NITER, 122
    /PLOT, 121
    /QUERY, 121
    /RESTART, 122, 268
    ANGLE, 131
    BEAM_PATCH, 132
    BEAM_SIZE, 125
    BLC, 131
    CLARK, 123
    CLEAN_ARES, 126
    CLEAN_FRES, 126
    CLEAN_GAIN, 127
    CLEAN_INFLATE, 127
    CLEAN_NCYCLE, 127
    CLEAN_NGOAL, 127
    CLEAN_NITER, 127
    CLEAN_NKEEP, 128
    CLEAN_POSITIVE, 128
    CLEAN_RESIDUAL, 128
    CLEAN_SEARCH, 129
    CLEAN_SIDELOBE, 129
    CLEAN_SMOOTH, 129
    CLEAN_SPEEDY, 129
    CLEAN_STOP, 129
    CLEAN_WORRY, 130
    HOGBOM, 123
    MAJOR, 131
    METHOD, 130
    Methods:, 123
    MINOR, 131
    MRC, 123
    MULTI, 124
    Old_Names:, 131
    SDI, 124
    TRC, 131
    Variables:, 125
COLLECT, 192
    /FLAG, 193
COLOR, 88
COLUMN, 15
COMBINE, 85, 237

/BLANKING, 238
/FACTOR, 238
/THRESHOLD, 238
CODE, 237
Command, 12

DEFINE
IMAGE, 272
DERIVE, 193
Example, 193
DISCARD, 132
DUMP, 132

EXPLORE, 70, 88
/ADD, 89
/NOPAUSE, 89
CENTER, 89
DO_BIT, 90
DO_CONTOUR, 90
DO_GREY, 90
DO_MASK, 90
RANGE, 90
SCALE, 91
SIZE, 91
SPACE_TYPE, 91
SPACING, 91
SPAN, 92
Variables:, 89
EXTRACT, 92

FEATHER, 45, 46, 201
/FILE, 202
/REPROJECT, 202
Algorithm, 202
FEATHER_EXPO, 203
FEATHER_RADIUS, 203
FEATHER_RANGE, 203
FEATHER_RATIO, 203
Variables:, 203
FIND, 62, 92
/SPECIES, 93
FIT, 132, 269
/JVM_FACTOR, 33, 41, 133, 269
/THRESHOLD, 134
BEAM_SIZE, 134
CLEAN_SIDELOBE, 135
FixedValues, 133
Results, 133
FITS, 18, 272
fits_to_uvt, 18
FLUX, 204
Limitations, 204
Results, 204

GO, 6
BIT, 6
MAP, 6
NICE, 6
PLOT, 6
UVSHOW, 6

HEADER, 15
HELP, 5, 6, 10
command subtopic, 11
SHOW, 6, 71
VIEW, 6
HOGBOM, 9, 27, 266
HOW_TO, 5, 204

INPUT, 6
INSPECT_3D, 69, 93
History, 93
SPAN, 94
Variables:, 94

JvM factor, 33

KEPLER, 85, 238
/HFS, 239
/MASK, 239
/RESET, 240
/VELOCITY, 85, 240
/VSYSTEM, 240
/WIDGET, 241
Display:, 244
INIT, 239
KEPLER_AZIMUT, 242
KEPLER_DIST, 241
KEPLER_INCLI, 241
KEPLER_PROFILE, 243
KEPLER_PV, 244
KEPLER_RINT, 242
KEPLER_RMAX, 242
KEPLER_RMIN, 241
KEPLER_ROTA, 241
KEPLER_ROUT, 242
KEPLER_SHOW, 244
KEPLER_SHOW%F, 245

KEPLER_SHOW%LAYOUT, 245
KEPLER_SHOW%R, 245
KEPLER_SHOW%T, 245
KEPLER_SHOW%V, 245
KEPLER_SPECTRUM, 244
KEPLER_STEP, 242
KEPLER_STRICT, 243
KEPLER_THETA, 242
KEPLER_VDISK, 243
KEPLER_VELO, 244
KEPLER_VMASS, 241
KEPLER_X0, 241
KEPLER_Y0, 241
Results:, 243
SHOW, 71, 239
Variables:, 241

Language, 86, 114, 191, 201, 237, 257
LOAD, 94, 268
/FREQUENCY, 94
/PLANES, 94
/RANGE, 95

MAP_COMBINE, 135
/BLANKING, 136
/FACTOR, 136
/THRESHOLD, 136
CODE, 136
MAP_COMPRESS, 137
Output, 137
MAP_CONTINUUM, 61, 79, 205
/METHOD, 205
C-SCM, 207
EGM, 207
GAUSS, 206
GLOBAL, 205
SCM, 206
MAP_INTEGRATE, 137
Output, 138
MAP_POLAR, 83, 84, 207, 267
/BACKGROUND, 208
/COMPUTE, 208
/STEP, 208
MAP_REPROJECT, 138
/BLANKING, 139
/LIKE, 139
/PROJECTION, 139
/SYSTEM, 140
/X_AXIS, 140
/Y_AXIS, 140
MAP_RESAMPLE, 141
/LIKE, 141
MAP_SMOOTH, 141
/ASYMMETRIC, 142
MASK, 32, 209
/THRESHOLD, 82
ADD, 210
APPLY, 210
CHECK, 210
INITIALIZE, 210
INTERACTIVE, 210
OVERLAY, 211
READ, 211
REMOVE, 211
SHOW, 211
THRESHOLD, 32, 55, 212
Tricks, 209
USE, 32, 213
WRITE, 213
MFS, 213
MODEL, 58, 195
/MINVAL, 195
/MODE, 196
MOMENTS, 213
/CUTS, 214
/MASK, 214
/METHOD, 214
/RANGE, 214
/THRESHOLD, 215
MOSAIC, 143
MRC, 9, 27, 266
MULTI, 9, 27
MULTISCALE, 266
MX, 27, 33, 143
Variables:, 144

PIPELINE, 7, 14, 75, 76, 80, 257, 267, 268
/MODE, 75, 257
/WIDGET, 75, 76, 258
ALL%AMPLI_NITER, 263
ALL%AMPLI_TIMES, 263
ALL%CATALOG, 262
ALL%COLLECT, 264
ALL%COMBINE, 264
ALL%DROP, 263
ALL%FILTER, 263

ALL%ITIME, 264
ALL%MINFRES, 264
ALL%PHASE_NITER, 265
ALL%PHASETIMES, 265
ALL%RANGE, 265
ALL%SPLIT, 265
APPLY, 260
Arguments:, 258
CHECK, 259
COLLECT, 260
COMPUTE, 260
FIND, 82, 259
IMAGE, 261
LAST, 262
NEXT, 262
ORGANIZE, 259
SAVE, 261
SELECT, 82, 259
SHOW, 261
SKY, 262
TABLES, 260
TIME, 260
Variables, 262
POPUP, 95, 268
PRIMARY, 145
/TRUNCATE, 146
PROPER_MOTION, 215
/FILE, 215

READ, 4, 5, 146
/COMPACT, 148
/FREQUENCY, 19, 148
/NOTRAIL, 148
/PLANES, 148
/RANGE, 19, 81, 149
BEAM, 31
Buffers, 147
CGAINS, 55
CONTINUUM, 61
DIRTY, 31
FITS, 147
MASK, 32
MODEL, 57, 58
Optimisation, 148
SINGLE, 9, 20, 47, 149
UV, 19, 24, 46, 55, 57

SCALE_FLUX, 54, 194
SDI, 9, 27, 266
SELFCAL, 49, 51, 52, 54, 245
/WIDGET, 52, 245
AMPLI, 54
AMPLITUDE, 246
APPLY, 52, 55, 246
Arguments:, 246
CLEAN_ARES, 253
CLEAN_FRES, 253
CLEAN_NITER, 253
FLAG, 55, 247
INPUT, 247
PHASE, 52, 54, 247
Results:, 248
SAVE, 52, 247
SELF_APPLIED, 248
SELF_CHANNEL, 249
SELF_COLOR, 250
SELF_DISPLAY, 252
SELF_DYNAMIC, 249
SELF_FLAG, 253
SELF_FLUX, 252
SELF_LOOP, 250
SELF_MINFLUX, 251
SELF_NITER, 250
SELF_PRECISION, 252
SELF_REFANT, 252
SELF_RESTORE, 252
SELF_RMSCLEAN, 249
SELF_SNOISE, 253
SELF_SNR, 253
SELF_STATUS, 249
SELF_TIMES, 251
SHOW, 52–54, 63, 73, 74, 247
SUMMARY, 52, 54, 248
Variables:, 249
SET, 95
ANGLE_UNIT, 96
OtherKeyword, 96
SHOW, 6, 42, 62, 71, 96, 272
/SIDE, 62, 97
BEAM, 98
BOX_LIMITS, 104
CCT, 36, 63, 64, 67, 98
CENTER, 106
CLEAN, 9, 62
COMPOSITE, 98

CONTINUUM, 99
COVERAGE, 19, 64, 99
CROSS, 106
DO_BIT, 104
DO_CONTOUR, 104
DO_COVERAGE, 104
DO_DIRTY, 104
DO_FIELDS, 105
DO_GREY, 105
DO_HEADER, 105
DO_LABEL, 105
DO_MASK, 105
DO_NICE, 105
DO_WEDGE, 106
FIELDS, 63, 99
FLUX, 99
History, 97
KEPLER, 71, 85, 100
Keywords:, 97
MARK, 106
MOMENTS, 100
NOISE, 34, 36, 63, 65, 100
PRIMARY, 100
PV, 101
RANGE, 106
SCALE, 106
SED, 101
SELFCAL, 63, 101
SHOW_SIDE, 108
SIZE, 107
SNR, 102
SOURCES, 102
SPACE_TYPE, 107
SPACING, 107
SPAN, 108
SPECTRA, 102
UV, 19, 57, 58, 63
UV_DATA, 102
UV_FIT, 20, 57, 58, 63, 66, 103
Variables:, 103
SLICE, 108
SOLVE, 51, 52, 196
/MODE, 196
SPECIFY, 142
BLANKING, 142
FREQUENCY, 71, 142
LINENAME, 143
TELESCOPE, 143
VELOCITY, 143
SPECTRAL_CLEAN, 253
DUAL, 254
FFT, 254
MULTI, 255
WAVE, 255
SPECTRUM, 108
/CORNER, 109
/MEAN, 109
/PLANE, 109
/SUM, 109
STATISTIC, 36, 51, 52, 62, 110
/EDGE, 110
/NOISE, 110
STOKES, 82, 83, 215, 271
/FILE, 216
SUPPORT, 32, 149
/CURSOR, 150
/MASK, 32, 150
/PLOT, 151
/RESET, 151
/THRESHOLD, 151
/VARIABLE, 151

TABLE, 47
TIP, 5
TIPS, 257, 268
TRANSFORM, 197
FFT, 197
WAVE, 198

UV_..., 20
UV_ADD, 216
/FILE, 84, 216
UV_BASELINE, 20, 81, 151
/CHANNELS, 71, 152
/FILE, 81, 152
/FREQUENCY, 152
/RANGE, 153
/VELOCITY, 153
/WIDTH, 153
uv_baseline, 60
UV_CHECK, 20, 153
/FILE, 154
BEAM_RANGES, 154
UV_CIRCLE, 20, 57, 58, 216
/SAMPLING, 217

/ZERO, 217
UV_COMPRESS, 19, 154
/CONTINUUM, 155
/FILE, 81, 155
UV_CONTINUUM, 20, 57, 155
/INDEX, 156
/RANGE, 156
UV_CORRELATE, 217
/FILE, 218
UV_DEPROJECT, 20, 57, 58, 218
UV_DETECT, 85, 255
/FILE, 256
Algorithm, 256
UV_EXTRACT, 156
/FILE, 81, 157
/RANGE, 157
UV_FIELDS, 157
/CENTER, 158
/FILE, 158
UV_FILTER, 20, 59, 81, 82, 159
/CHANNELS, 71, 159
/FILE, 81, 159
/FREQUENCY, 160
/RANGE, 160
/RESET, 160
/VELOCITY, 160
/WIDTH, 161
/ZERO, 161
uv_filter, 60
UV_FIT, 20, 57, 58, 63, 218
/QUIET, 219
/SAVE, 219
/WIDGET, 220
History, 220
r, 63
ResultTable, 220
ResultValues, 221
t, 58
UV_FLAG, 19, 36, 161
/ANTENNA, 162
/DATE, 162
/FILE, 163
/RESET, 163
CHANNEL, 163
UV_HANNING
/FILE, 81
UV_MAP, 5, 9, 11, 20, 21, 24, 26, 31, 36, 40, 57, 82, 163
/CONT, 59, 164
/FIELDS, 165
/INDEX, 165
/RANGE, 165
/SELF, 51
/TRUNCATE, 166
?, 24
BEAM_STEP, 166
convolution, 171
map_angle, 172
map_beam_step, 172
MAP_CELL, 167
MAP_CENTER, 167
MAP_CONVOLUTION, 167
map_dec, 172
MAP_FIELD, 168
MAP_POWER, 168
MAP_PRECIS, 168
map_ra, 172
MAP_ROBUST, 168
MAP_ROUNDING, 169
MAP_SIZE, 169
MAP_TAPEREXPO, 170
MAP_TRUNCATE, 170
MAP_UVCELL, 170
MAP_UVTAPER, 170
MAP_VERSION, 170
mcol, 172
Mosaics, 164
Old_Names:, 171
taper_expo, 173
uv_cell, 173
uv_shift, 173
uv_taper, 173
Variables:, 166
wcol, 173
weight_mode, 173
UV_MAP , 26
UV_MERGE, 221
/FILE, 20, 81
/FILES, 221
/MODE, 222
/MODE STACK, 85
/SCALES, 222
/WEIGHTS, 223

UV_MOSAIC, 223
MERGE, 223
SPLIT, 224
UV_PREVIEW, 6, 7, 19, 20, 54, 59, 71, 72, 82, 224
/BROWSE, 224
/FILE, 81, 225
Algorithm, 225
Limitations, 226
Output, 226
uv_preview, 60
UV_RADIAL, 20, 57, 58, 227
/SAMPLING, 227
/ZERO, 227
UV_RESAMPLE, 6, 19, 82, 174
/FILE, 81, 174
/LIKE, 174
UV_RESIDUAL, 20, 57, 58, 175
Note, 175
UV_RESTORE, 13, 20, 33, 41, 59, 175
/COPY, 176
/SELF, 51
/SPEED, 176
UV_REWEIGHT, 20, 54, 176
/FILE, 178
/FLAG, 178
/RANGE, 178
APPLY, 177
DO, 177
ESTIMATE, 177
TIME, 178
UV_SELECT, 57, 58, 199
UV_SELF, 20, 198
/RANGE, 198
/RESTORE, 199
UV_SHIFT, 20, 179
/FILE, 179
UV_SHORT, 20, 37, 42, 45–47, 228
/CHECK, 228
/REMOVE, 47, 228
Algorithm, 228
SHORT_DO_PRIMARY, 231
SHORT_DO_SINGLE, 231
SHORT_FTOLE, 231
SHORT_IP_BEAM, 231
SHORT_IP_DIAM, 232
SHORT_MCOL, 232
SHORT_MIN_WEIGHT, 232
SHORT_MODE, 232
SHORT_SD_BEAM, 233
SHORT_SD_DIAM, 233
SHORT_SD_FACTOR, 233
SHORT_SD_WEIGHT, 234
SHORT_TOLE, 234
SHORT_UV_MAX, 234
SHORT_UV_MIN, 235
SHORT_WCOL, 235
SHORT_WEIGHT_MODE, 235
SHORT_XCOL, 235
SHORT_YCOL, 235
Step_1, 229
Step_2, 230
Variables:, 231
Zero_Spacing, 229
UV_SMOOTH, 179
/ASYMMETRIC, 180
/FILE, 180
UV_SORT, 81, 199
/FILE, 81, 200
UV_SPLIT, 180
/CHANNELS, 181
/FILE, 81, 181
/FREQUENCY, 182
/RANGE, 182
/VELOCITY, 182
/WIDTH, 183
UV_STAT, 20, 24, 26, 82, 183
ADVISE, 184
ALL, 185
Arguments:, 184
BEAM, 185
BRIGGS, 185
Casa, 183
CELL, 185
DEFAULT, 186
HEADER, 186
Mosaics, 184
RESET, 186
SETUP, 57, 186
TAPER, 186
WEIGHT, 186
UV_TIME, 6, 20, 82, 187
/FILE, 81, 187
/WEIGHT, 187

UV_TRIM, 187
    /FILE, 188
UV_TRUNCATE, 20, 188

VIEW, 5, 6, 42, 67, 111, 272
    /NOPAUSE, 111
    /OVERLAY, 67, 85, 112
    BEAM, 113
    CCT, 36, 67
    Keys, 112
    Keywords:, 113
    NOISE, 67
    PRIMARY, 114
    Scripts, 113
    Variables, 113

WRITE, 4–6, 47, 188
    /APPEND, 81, 189
    /RANGE, 189
    /REPLACE, 81, 189
    /TRIM, 190
    CGAINS, 55
    CONTINUUM, 61
    Format, 189
    MASK, 32
    SUPPORT, 32
    UV, 19

XY_SHORT, 47