Using the DCR Tool

Jim Braatz, NRAO

25 Feb, 2002

Contents

1 Purpose

This document describes the usage of the DCR tool, an AIPS++ tool used to analyze continuum data taken with the Digital Continuum Receiver (DCR) backend on the Green Bank Telescope (GBT). The DCR tool is the primary package used for continuum analysis during commissioning of the GBT. It allows access to raw data from the DCR, it handles pointing and focus analysis, and it includes a number of functions which make up the toolkit used by the astronomers commissioning the GBT. Continuum imaging and other science-related continuum observations are handled elsewhere in the AIPS++ system.

This documentation is written specifically for the staff and visitors to the Green Bank site. Minor changes may be required when used at other sites.

2 Sample DCR Data Reduction Session

```
% . /home/aips++/gbt/aipsinit.sh
% gbtmsfiller project=/home/gbtdata/pnt_lowgreg_25 minscan=110 maxscan=120
 backend=DCR
% aips++ -l dcr.g

- md := dcr('pnt_lowgreg_25_DCR')
- scan110 := md.getscan(110,0)
- md.point1(114,0,archive=T)
```

3 Filling DCR Data

Prior to accessing any GBT data with AIPS++ analysis tools, the data must first be converted from the native FITS files written by the GBT Monitor and Control system into an AIPS++ Measurement Set. The gbtmsfiller application is used to do this conversion. To run the filler, first issue the following UNIX command from the bash shell:

```
. /home/aips++/gbt/aipsinit.sh
```

or the following command, if using csh:

```
source /home/aips++/gbt/aipsinit.csh
```

The filler can then be run as follows:

```
gbtmsfiller project=/home/gbtdata/pnt_prime_01 backend=DCR minscan=1 maxscan=2
```

Other options are available in the gbtmsfiller application. The program will prompt for inputs if it is run as follows:

```
gbtmsfiller help=prompt
```

For complete documentation on the gbtmsfiller, see the User Reference Manual.

4 Starting the DCR Tool

To start AIPS++, issue the aipsinit.sh command as shown above (if not already issued) and enter the UNIX command:

```
aips++ -l dcr.g
```

This will start the AIPS++ system and open a pgplotter window to be used for showing DCR results. An instance of the DCR tool can then be started from the Glish prompt as follows:

```
- mydcr := dcr('pnt_prime_01_DCR')
```

You see a GUI which includes quick access to a number of common functions. Note that this particular instance of the DCR tool is attached to the specified data set. Use the complete path to specify that data set if it is not in the directory from which AIPS++ was started. To look at data from another data set within the same AIPS++ session, simply open another DCR tool as follows:

```
- mydcr2 := dcr('pnt_prime_02_DCR')
```

When access to one of the instances of the tool is no longer needed, it can be closed down by:

```
- mydcr2.done()
```

5 Using the DCR Tool GUI

When the DCR tool GUI is started, it shows the available scans in the "Scan" listbox. When a scan is selected with the mouse, the "Receiver" and "Phase" listboxes are filled in with the relevant choices, and the text area is filled with some header information. The Cal value can be set in the given entry box. The TPnC Cal entry value is not yet used. Raw counts can be plotted for a given phase by selecting a Receiver and Phase and then clicking the "Phase" button in the "Plot v. Time' row. Antenna temperature and source temperature can be plotted likewise, when appropriate.

The "Plot" row offers four buttons to plot the data counts vs. RA, data counts vs. Dec, antenna temperature vs. RA, and antenna temperature vs. Dec, respectively. The "Procedures" row allows easy access to the automated routines for reducing pointing and focus scans. The "Point1" routine plots a single scan's data as a function of RA or Dec offset, fits a baseline and gaussian to the cross scan, and returns information about the Az and El offsets determined from the scan. "Point2" and "Point4" work similarly on sequences of 2 or 4 scans in a pointing observation. Select the first scan of the sequence in the Scan listbox, then choose the desired pointing function.

Along the bottom of the DCR tool GUI are several more control buttons. The "Update Scans" button is used to update the scan list when the filler and DCR tool are being used in the online mode. The "Clear Text" button is self-explaining, and the "Summary" button lists some information on the scans in the scan listbox. The "Ready" button is a status indicator, and the "Dismiss" button shuts down this particular instance of the DCR tool.

6 Using the DCR Tool from the Command Line

The heart of the DCR Tool is the getscan function. It is used to mine the Measurement Set and return the data for a given scan into a simple Glish record, which can then be used for analysis using the large suite of functions and tools available. The getscan function has only two parameters, the scan number and (optionally) a focus parameter. Focus values may take up to several seconds to extract from the data, so they are not returned by default. The syntax for simple use of getscan (without focus data) is as follows:

```
- scan1 := mydcr.getscan(1)
```

This creates a Glish record structured as follows:

```
double     scan1.time
record     scan1.GO_header
double     scan1.GO_header.TIME
double     scan1.GO_header.INTERVAL
string     scan1.GO_header.SCAN
string     scan1.GO_header.PROJID
string     scan1.GO_header.SOURCE
string     scan1.GO_header.SCANID
string     scan1.GO_header.FITSVER
string     scan1.GO_header.OBSERVER
string     scan1.GO_header.PROCNAME
string     scan1.GO_header.PROCTYPE
integer    scan1.GO_header.PROCSEQN
integer    scan1.GO_header.PROCSIZE
string     scan1.GO_header.RADECSYS
double     scan1.GO_header.RAJ2000
double     scan1.GO_header.DECJ2000
double     scan1.GO_header.EQUINOX
double     scan1.GO_header.CNTRFREQ
string     scan1.GO_header.SWTCHSIG
integer    scan1.scan
float      scan1.data
float      scan1.lpc_az1
float      scan1.lpc_az2
float      scan1.lpc_el
record     scan1.pmodel
string     scan1.pmodel.FITSVER
float      scan1.pmodel.DELTAUTC
float      scan1.pmodel.IERSPMX
float      scan1.pmodel.IERSPMY
float      scan1.pmodel.MNTOFFAZ
float      scan1.pmodel.MNTOFFEL
float      scan1.pmodel.LPC_AZ1
float      scan1.pmodel.LPC_AZ2
float      scan1.pmodel.LPC_EL
string     scan1.pmodel.PNTMODEL
float      scan1.pmodel.PNTAZD00
float      scan1.pmodel.PNTAZB01
float      scan1.pmodel.PNTAZD01
float      scan1.pmodel.PNTAZA11
```

```
float       scan1.pmodel.PNTAZB11
float       scan1.pmodel.PNTAZC21
float       scan1.pmodel.PNTAZD21
float       scan1.pmodel.PNTELD00
float       scan1.pmodel.PNTELC10
float       scan1.pmodel.PNTELD10
float       scan1.pmodel.PNTELB01
float       scan1.pmodel.PNTELD01
string      scan1.pmodel.DIABMODE
string      scan1.pmodel.POLARMTN
string      scan1.pmodel.COSVMODE
float       scan1.pmodel.AMBTEMP
float       scan1.pmodel.AMBPRESS
float       scan1.pmodel.AMBHUMID
string      scan1.pmodel.OPTICSMD
string      scan1.pmodel.FOCTRMOD
string      scan1.pmodel.FOCPATM
double      scan1.az
double      scan1.el
double      scan1.ra
double      scan1.dec
```

In fields where it is appropriate, the values are stored in arrays. For example, scan1.time is an array containing the time for each of the samples in the scan. The data are in scan1.data, which is a 3-D array with the first index pointing to receiver, the second to phase, and the third to the sample number. The third sample from receiver 0 and phase 1 would then be accesses as Glish value scan1.data[0,1,3].

The full syntax of the getscan function is:

```
scan1 := mydcr.getscan(scan,getFocus=F)
```

The getFocus parameter can be omitted to pass the default F value. If set to T, the relevant focus arrays will be included into the returned record. For prime focus observations, these are:

```
string      scan1.focusparam
string      scan1.optics
float       scan1.PF_FOCUS
float       scan1.PF_ROTATION
float       scan1.PF_X
```

and for gregorian focus observations, these are:

```
string     scan1.focusparam
string     scan1.optics
float      scan1.SR_XP
float      scan1.SR_YP
float      scan1.SR_ZP
float      scan1.SR_XT
float      scan1.SR_YT
float      scan1.SR_ZT
```

The focusparam field in the returned record is an indicator to which focus values are available in the given record. For the cases above, scan1.focusparam has the value 'all'. It is possible to load only one focus parameter into the returned record by sending the desired parameter in the getscan function. For example:

```
scan3 := md.getscan(3,'PF_FOCUS')
```

This getscan call produces scan3.focusparam equal to 'PF_FOCUS' and a single entry in the record for focus:

```
float      scan3.FOCUS
```

The following commands, then, show a way to plot the focus values separately from the canned focus procedures in the DCR tool:

```
- md := dcr('pnt_prime_12_DCR')
- scan2030 := md.getscan(2030,'PF_FOCUS')
- dcrpg.ploty(scan2030.FOCUS)
```

7 General Notes on DCR Tool Usage

There are several useful shortcuts and hints that can be taken advantage of when using the DCR tool. For example, the `dcrpg` tool listed in the previous example is the plotter tool, which is open and available while using the dcr tool, and is accessed by the name `dcrpg`. So one can pass commands to the tool at any time. For example, clear the plotter with:

```
- dcrpg.clear()
```

Once a scan is selected, either via the GUI or by the command line, the data and functions both are available from the tool name. Data from the most recently accessed scan is available under a subrecord called currentscan, e.g.:

```
- md := dcr('pnt_lowgreg_23_DCR')
- scan21 := md.getscan(21,T)
- dcrpg.ploty(md.currentscan.data[1,1,])
```

In this example of CLI use, md.currentscan has the same structure and values as the record returned by the getscan function, scan21. However, when a scan is selected in the GUI, the md.currentscan values are updated again to reflect the selection. In this way, scans can be selected from the GUI and their data subsequently manipulated in the CLI.

Also available is a pointer the the measurement set table, as a subrecord called maintable. So for example, one might browse the MS as follows:

```
- md := dcr('pnt_lowgreg_23_DCR')
- md.maintable.browse()
```

8 Functions in the DCR Tool

The following functions are available in the DCR tool. The syntax provided in this guide is such that if a parameter has no assignment then there is no default value and it must be supplied by the user. If an assignment is shown, then the default value is listed. So for example, a call to the baselinefit function may omit the last parameter, plotflag, and it will be read as T, but the four preceding parameters must be supplied by the user.

8.1 baselinefit

```
baselinefit(xarray,yarray,ord,range,plotflag=T)
Example: md.baselinefit(chan,tant,3,[10,100,350,512])
Returns: Baseline-subtracted array
```

Fit a baseline. xarray and yarray are the data to be fit (for example, the channel number and antenna temperature arrays), ord is the order of the polynomial to fit, and range is the range of channels to fit, in the same units as the xarray data. If plotflag is T, the residuals will be plotted.

8.2 done

```
done()
Example: md.done()
```

End the given instance of the DCR tool.

8.3 focus

```
focus(tablename)
Example: md.focus('focusvals.txt')

  -- sample focusvals.txt --
  1.0 6.7
  1.4 6.9
  1.8 7.4
  2.2 7.3
  2.6 7.0
  3.0 6.4
  3.4 6.0
  --      end       --
```

Focus reduction. This routine is still under development. Focus observations are expected to happen as a sequence of pointing scans, with the focus being incremented between each pointing scan. The result is a table of focus positions and peak brightness values. This routine simply reads in an ASCII table with those values and fits a parabola to them, reporting the peak. For now, the ASCII table must be created by hand and it must have no blank rows or information other than the focus position (Column 1) and peak brightness (Column 2).

8.4 focusScan

```
focusScan(scan,receiver=0,cal_value=1,param='SR_XP',order=2,archive=F)
Example: md.focusScan(114,cal_value=3.2,param='PF_FOCUS',order=2,archive=T)
```

Focus scan reduction. This routine reduces and displays the data from a classical focus scan, in which a source is tracked and the signal strength is measured as the focus value is scanned through a range of values. Any of the

9 focus parameters may be reduced with this routine. Specify which focus to use via the param value, which can be SR_XP, SR_YP, SR_ZP, SR_XT, SR_YT, SR_ZT, PF_X, PF_ROTATION, or PF_FOCUS. The order parameter specifies the order of the polynomial to fit, and the archive parameter specifies whether the results will be appended to an ASCII file on disk, called focus.dat.

8.5 gauss

```
gauss(xarray,yarray,height,width,center,plotflag=1)
Example: md.gauss(RA_array,Tant,10,3.5,214.5)
```

Fit a gaussian. The parameters xarray and yarray specify the x- and y-data arrays to be fit. The next three parameters are initial guesses for the height, width, and center of the gaussian, in the same units as the xarray and yarray entries (as appropriate). The plotflag parameter specifies whether the result will be drawn over the existing plot.

8.6 get_tant

```
get_tant(scan,receiver=1,cal_value=1)
Example: md.get_tant(115,cal_value=3.2)
```

Return array of antenna temperatures.

8.7 getGO

```
getGO(scan)
Example: md.getGO(114)
```

Return GO header. Many of the setup parameters are listed in the GO header. This function returns a record containing those fields.

8.8 getscan

```
getscan(scan,getFocus=F)
Example: md.getscan(114)
```

Returns data associated with the given scan. See the discussion above.

8.9 guessmode

```
guessmode(scan)
Example: md.guessmode(114)
```

Guess observing mode. By looking at the number of phases and receivers, this function attempts to determine the switching scheme. This is outdated since an entry in the GO header now gives the scheme directly.

8.10 listscans

```
listscans()
Example: md.listscans()
```

List scannumbers.

8.11 plot_focus_time

```
plot_focus_time(scan,param='SR_XP')
Example: md.plot_focus_time(122,'PF_FOCUS')
```

Plot focus value against time.

8.12 plot_gain_time

```
plot_gain_time(scan,receiver)
Example: md.plot_gain_time(114,0)
```

Plot gain against time.

8.13 plot_phase_dec

```
plot_phase_dec(scan,receiver,phase)
Example: md.plot_phase_dec(114,0,1)
```

Plot data in counts, for a given receiver and phase, against declination.

8.14 plot_phase_ra

```
plot_phase_ra(scan,receiver,phase)
Example: md.plot_phase_ra(114,0,1)
```

Plot data in counts, for a given receiver and phase, against RA.

8.15 plot_phase_time

```
plot_phase_time(scan,receiver,phase)
Example: md.plot_phase_time(114,0,1)
```

Plot data in counts, for a given receiver and phase, against time.

8.16 plot_RA_Dec

```
plot_RA_Dec(scan)
Example: md.plot_RA_Dec(112)
```

Plot declination as a function of RA for each sample in the scan.

8.17 plot_sidelobe

```
plot_sidelobe(scan,receiver=1,basepct=10,bottom=-70)
Example: md.plot_sidelobe(92,0,15)
```

Plot sidelobes on the standard axes. This function plots sidelobes in dB, in the standard manner, for a given scan. A strong source gives the best results. The basepct parameter is a fraction of the total number of samples, measured from each edge, to be used in determining a baseline. For example, in a scan with 600 samples, a basepct=10 would use 60 samples at each edge to determine the baseline. The bottom parameter is the lower limit, in dB, to be used on the plot. Experiment with several values in the range, say, [-80,-40] to get a satisfactory result.

8.18 plot_tant_Dec

```
plot_tant_Dec(scan,receiver=1,cal_value=1)
Example: md.plot_tant_Dec(112, cal_value=3.2)
```

Plot antenna temperature as a function of declination.

8.19 plot_tant_RA

```
plot_tant_RA(scan,receiver=1,cal_value=1)
Example: md.plot_tant_RA(112, cal_value=3.2)
```

Plot antenna temperature as a function of RA.

8.20 plot_tant_time

```
plot_tant_time(scan,receiver=1,cal_value=1)
Example: md.plot_tant_time(112)
```

Plot antenna temperature as a function of time.

8.21 plot_tsrc_time

```
plot_tsrc_time(scan,receiver,cal_value=1)
Example: md.plot_tsrc_time(112)
```

Plot source temperature as a function of time.

8.22 plotscans

```
plotscans(bscan,escan,rcvr=0,phase=0)
Example: md.plotscans(112,114)
```

Plot raw data for a given phase and receiver, concatenated over several scans. The bscan and escan parameters specify the beginning and ending scan numbers for the sequence.

8.23 point1

```
point1(scan,receiver,xaxis=0,cal_value=1,basepct=10,plotflag=1,archive=F)
Example: md.point1(14,1,archive=T)
```

Reduce a pointing scan. This function determines the antenna temperature for the given scan, removes a baseline, fits a gaussian, and determines the Az and El offsets required to peak up the pointing. The parameter xaxis tells whether the scan is in RA (xaxis=1) or Dec (xaxis=2). If left to the default value of xaxis=0, the function will attempt to determine which is the applicable one. It is almost always safe to use xaxis=0. The basepct parameter indicates the fraction of samples to use, from each edge, in determining the baseline. A value of 10, for example, indicates that 10% of the scan should be used from each edge, for a total of 20% of the samples used to determine the baseline. If plotflag is F, the gaussian fit will not be shown. If archive is T, a listing of the results will be saved to the pointing.dat archive file on disk.

8.24 point2

```
point2(scan,receiver,cal_value=1,plotflag=1)
Example: md.point2(101,0)
```

Reduce a pair of pointing scans. Pointing scans generally come in pairs, with the telescope first moving in one direction, then back. This function reduces each of the two scans using point1, then plots both and reports the required Az and El corrections from the average. If plotflag is F, no plotting is done.

8.25 point4

```
point4(scan,receiver,cal_value=1,plotflag=1)
Example: md.point2(101,0)
```

Reduce a group of four pointing scans. A complete pointing sequence often includes a pair of RA scans followed by a pair of Dec scans. This function reduces each of the four scans using point1, then plots them and reports the required Az and El corrections from the averages. If plotflag is F, no plotting is done.

8.26 radec_to_azel

```
radec_to_azel(mjd,ra2000='0h0m0s',dec2000='0d0m0s')
Example: md.radec_to_azel(52138.5633,'12h00m00s','10d0m0s')
```

Converts RA and Dec to Az and El for the position of the GBT. The mjd parameter is the mean Julian Date, in days. RA and Dec are in J2000 coordinates. The returned value is an AIPS++ measure giving Az and El. The m0.value return value is RA, and m1.value is Dec.

8.27 scanSummary

```
scanSummary()
Example: md.scanSummary()
```

Summarize the current scan. This function prints header information for the scan currently selected, or most recently read with the getscan function.

8.28 test_srp

```
test_srp(scan,receiver,phase)
Example: md.test_srp(114,0,1)
```

Test for the existence of a given scan, receiver, phase combination. This function is mostly used internally, but may be of use for external analysis.

8.29 tsys

```
tsys(scan,receiver)
Example: md.tsys(114,0)
```

Calculate system temperature. Returns an array.