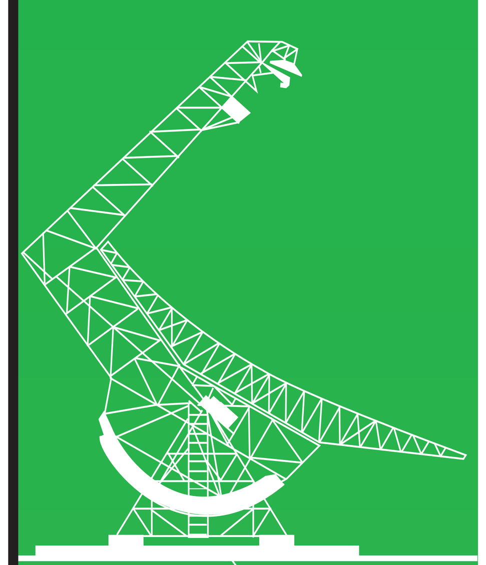
Update on Data Processing with the GBT

• Robert Garwood (NRAO-CV) • James A. Braatz (NRAO-CV) • Ronald J. Maddalena (NRAO-GB) • Paul Marganian (NRAO-GB) • Anthony H. Minter (NRAO-GB)

Abstract:

The Green Bank Telescope (GBT) is a general purpose instrument that operates at frequencies from 300 MHz to 50 GHz, and addresses a range of astronomical problems from mapping the Moon to studying the earliest structures in the universe. Data processing and analysis for the GBT is complex and relies on a combination of strategies, generally tied to the data recording device (backend) being used. Pulsar and radar observations are reduced and analyzed using software developed by individual research groups. All other GBT data analysis is supported by in-house efforts. We will outline the data processing paths and software applications used for each type of observation, including continuum imaging, spectral line on-the-fly mapping, and point source spectroscopy. The primary package for calibration and reduction of spectral line data is GBTIDL. We will describe flagging capabilities recently added to this package. GBTIDL is extensible and is used as a base by observers using specialized methods such as polarization measurements or pulsed spectral line experiments. The package also lends itself to incorporation of alternative calibration algorithms and techniques.



1. The Robert C. Byrd Green Bank Telescope (GBT)

- 100-m diameter single dish radio telescope
- Fully-steerable antenna; 85% coverage of the celestial sphere
- Unblocked aperture
- Active surface
- Frequency coverage: 300 MHz to 50 GHz
- Backends (data recording device)
  - Continuum: Digital Continuum Receiver (DCR), Caltech Continuum Backend (CCB)
  - Spectral Line: Spectrometer, Spectral Processor
  - Pulsar: Spigot Card, Spectral Processor
  - VLBA/VLBI
  - User-Provided Backends

2. Continuum Data Processing

The DCR is the GBT's general purpose continuum backend. It is used both for utility observations to characterize the properties of the telescope and the atmosphere (focus, pointing, opacity) as well as for extended source mapping (imaging).

Utility observations are reduced in near-real-time using GFM (GBT FITS Monitor). GFM is integrated into ASTRID (Astronomer's Integrated Desktop) along with the other tools that observers use to prepare and submit scheduling blocks and monitor the observations. The results of pointing and focus observations can be fed back to the telescope control system to adjust those properties of the telescope.

DCR mapping data can be calibrated and imaged in aips++. ***Figure 1*** shows an 8.4 GHz continuum image of the Rosette Nebula taken with the GBT and imaged using aips++ (Ghigo and Maddalena, 2003). ***Figure 2*** shows a continuum image of SN 1007 at 1370 MHz taken with the GBT and the VLA and imaged using aips++ (Dyer et. al.)

The Caltech Continuum Backend (CCB) is a wideband continnuum backend designed exlusively for use with the GBT Ka-band receiver.

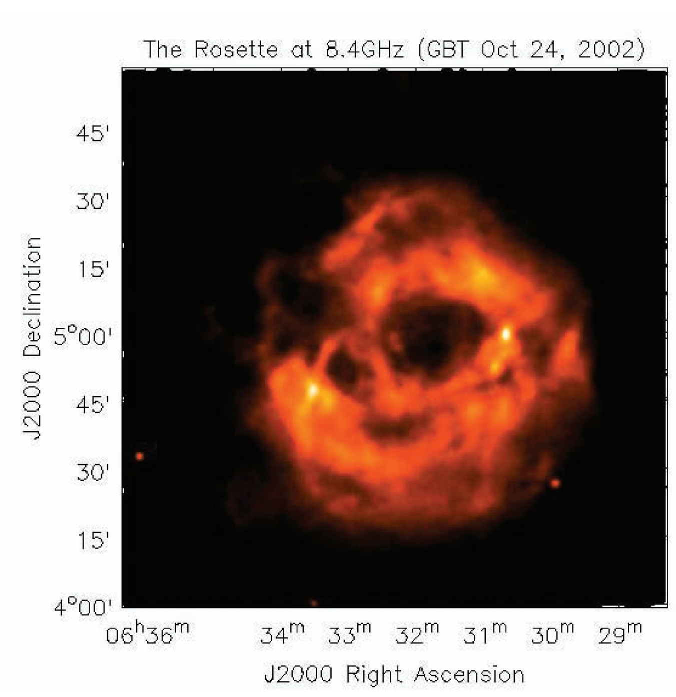


Figure 1: 8 GHZ continuum image of the Rosette Nebula using the GBT (Ghigo and Maddalena)

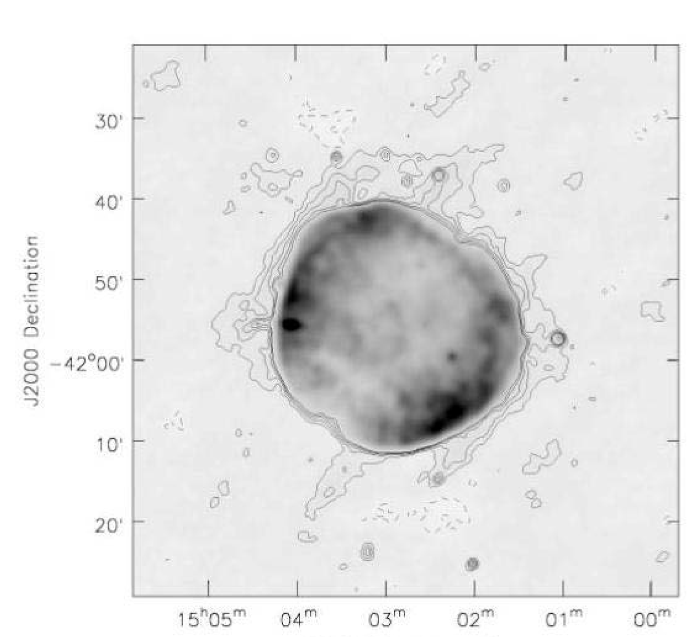


Figure 2: Continuum GBT and VLA image of SN1006. Image produced using aips++. (Dyer et. al.)

3. Spectral Line Data Processing

GBTIDL is the recommended package for processing single pointing (non-imaging) spectral line data from the GBT. A data flagging and blanking capability has recently been added to GBTIDL. The built-in flagging tools can be used directly to mark data inappropriate for averaging, or they can be incorporated into (user-developed) higher level tools to flag data based on statistical tests.

There are two supported routes for making images from GBT spectral line data.

- The data can be pre-processed using IDL and exported to classic AIPS. ***Figure 3*** shows GBT observations of Galactic 21cm emission that have been imaged using AIPS.
- Dish, as part of aips++, can be used to calibrate and image GBT spectral line data. Aips++ tools can also be used to combine GBT data with synthesis data. Dish development is frozen along with aips++ development.

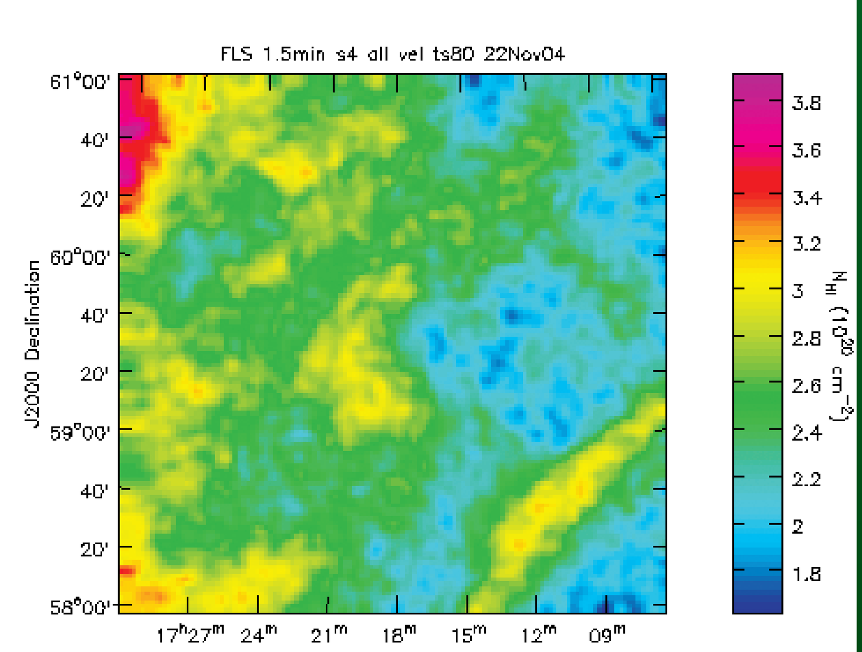


Figure 3: GBT HI image created using AIPS (Lockman and Condon)

4. GBTIDL

- Written entirely in Interactive Data Language (IDL), a commercial product of ITT Visual Information Solutions
- Can be used as a calculator that operates on spectra.
- Calibration procedures are provided for data taken in one of the GBT standard observing modes.
- Easy to explore alternative calibrations
- Data blanking and flagging (recent addition, described in more detail in section 5).
- Interactive plotter that can produce high-quality postscript files
- User-contributed code is encouraged and distributed with each release
- Can be run online, giving immediate access to the most recent data coming off the telescope.
- Data I/O is well isolated so additional data formats can be supported.

5. Flagging and Blanking in GBTIDL

Blanking is the process of replacing spectral intensities for a given set of channels with a special value recognized by the data analysis system (GBTIDL). Flagging is the process of establishing rules so that when data on disk is read into GBTIDL, the data described by those rules is blanked. In general, blanked values cannot be undone easily because they are replacement values, stored as data. The only way to undo a blanked value is to return to the original source of data and reload the (possibly uncalibrated) spectra. Flagging rules, on the other hand, can be undone or selectively applied and ignored. Since the data on disk are not changed by these rules, one simply has to re-read the data from disk after changing the flagging rules. Flagging rules can be specified by individual channel, channel range, individual spectrum, or by selection criteria to identify a group of spectra that that rule applies to. Rules are also identified by a user-supplied name so that rules having the same name can be selectively applied, ignored, or removed. Flagging is typically an iterative process:

1. Calibrate the raw data.
2. Examine the calibrated data and determine whether any flagging is required to improve calibration.
3. If necessary, flag the offending data and return to step 1.
4. Write a new data file with calibrated data. In general, the new data file should contain an entry for each integration that will be considered as a candidate for the average.
5. When all data are calibrated and written to disk, specify the calibrated data file as the new source of input.
6. Again examine the data and use the flagging procedures to mark residual bad data to exclude from the average.
7. Average the data.
8. Examine the average and, if necessary, return to step 1 or step 5 and modify the flagging commands as necessary.
9. Proceed with analysis of the averaged spectrum.

GBTIDL Flagging and Blanking Commands

Arguments in [] are optional. Arguments with leading "/" are toggles (on if set, otherwise off).

Command	Comments
replace, [bchan, echan, /blank]	Replace the data in channels bchan through echan with another value. If the /blank toggle is set, the data are replaced by the special blanked value. This can only be used on a spectrum already in memory.
flag, [scan, intnum, plnum, ifnum, fdnum, bchan, echan, chans, chanwidth, idstring, scanrange]	Create a flag rule based on the selection criteria (scan, integration, polarization, etc). If not specified, that criteria will match all data. The idstring can be used to identify the flag rule or groups or rules.
flagrec, record, [bchan, echan, chans, chanwidth, idstring]	Write a flag rule including some or all channels in a specific row (one spectrum).
listflags, [idstring, /summary]	List some or all of the flag rules associated with the current data file.
listids	List all of the unique idstrings in the flag rules.
unflag, id	Remove a flag rule or set of rules by number or idstring.
get, [useflag, skipflag, parameters]	Get some data and apply all or a limited set of flags rules (by exclusion or inclusion using their idstring values). Data that match the applied flag rules will be blanked. The parameters is a set of arguments that allow data selection. All other data retrival routines (not shown here) support this syntax for applying flags.

6. A bulk-flagging example

It is straightforward to write an IDL script that automatically generates flagging rules for an entire data set based on any criteria (e.g. statistics). We refer to this as "auto-flagging" to distinguish it from single rule command-line flagging where the user types in a specific flag command by hand.

This example IDL script illustrates this. The script flags channels in each spectrum based on flux deviation from the mean.

Example RFI Auto-flagging script

Syntax: flagbyrec, filename, edgedrop, sigmaflag, bufferflag, reason, startchan, lastchan

This generates flag rules by examining each spectrum (record) in the data file (filename). Data are flagged when the intensity is more than sigmaflag away from the mean. Extra channels can be flagged on either side of a flagged signal (bufferflag is true). The edges can be excluded from the mean and from any flags. The resulting set of flag rules all share the same idstring (reason).

The argument checking, comments, and error checking found in the procedure have been excluded from this example for clarity.

Command	Comments
filein, filename	Open the data file
totalcount = nrecords()	How many records are in the file?
freeze	Turn off plotter auto-updates after each record is retrieved from disk.
for recnum=0,totalcount-1 do begin	Start looping over all records
getrec, recnum, /skipflag	get the record (data plus header), do not apply any existing flags
a = getdata()	get the data array from the data container already in memory
; code to exclude edges in mean not shown here	
myrms = stddev(a)	Get the RMS
mymean = mean(a)	and the mean
up = mymean + sigmaflag*myrms	upper limit
down = mymean + sigmaflag*myrms	lower limit
bad = where((a lt down) or (a gt up), bandcount)	find the locations of bad data
if badcount gt 0 then begin	was any bad data found
; code not shown	It is useful to compact the set of bad channels to as small a set of ranges as possible so that when a future data retrieval command uses the flag rules that they can be applied efficiently with as many vector-like operations as possible. The code that does that here is not shown. It will eventually be encapsulated into a separate routine for use in other auto-flagging procedures.
;	e.g. brange=[40, 200, 500], erange=[50, 200, 600] indicates channels 40 through 50, 200 and 500 through 600 are to be flagged
flagrec, recnum, bchan=brange, echan=erange, idstring=reason	Finally, write out the flag rule
endif	End of section when badcount is positive
endfor	End of loop over all records
unfreeze	Turn the plotter auto-update back on

The plots below show calibrated and averaged GBT data before ***(Figure 4)*** and after ***(Figure 5)*** autoflagging.

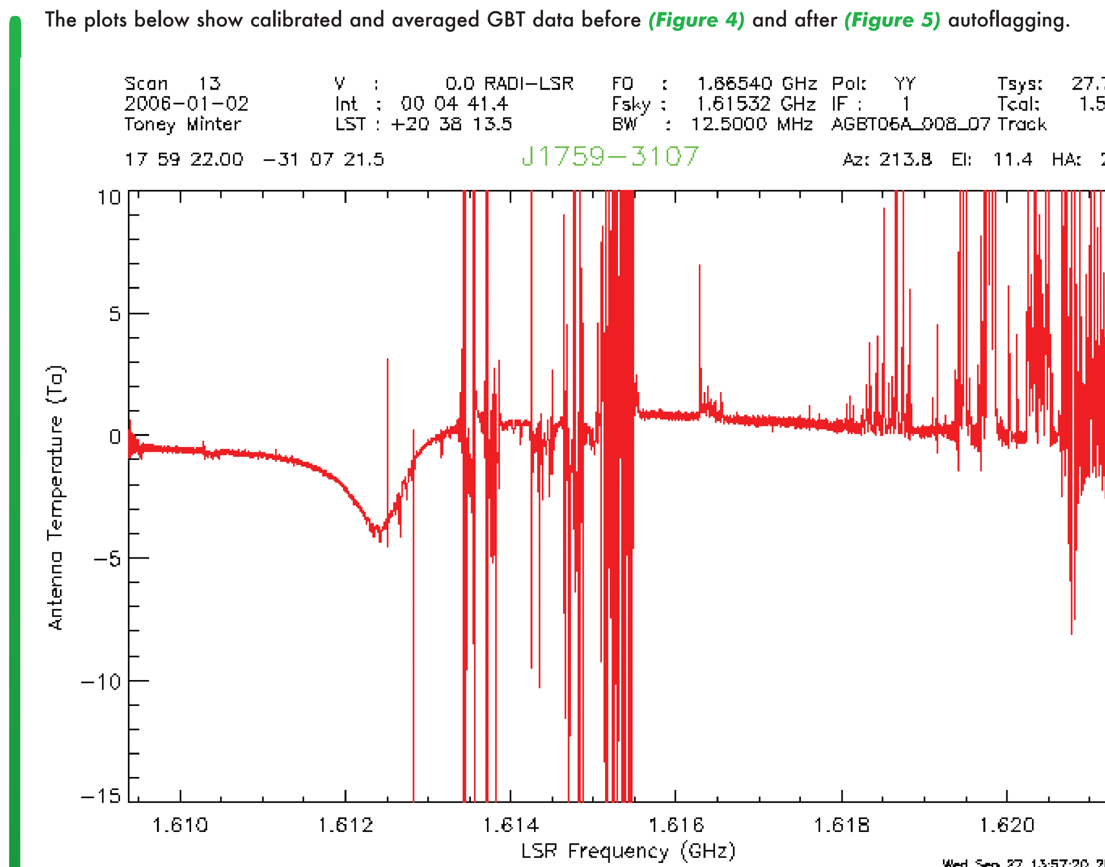


Figure 4: Unflagged, calibrated and averaged GBT data showing RFI.

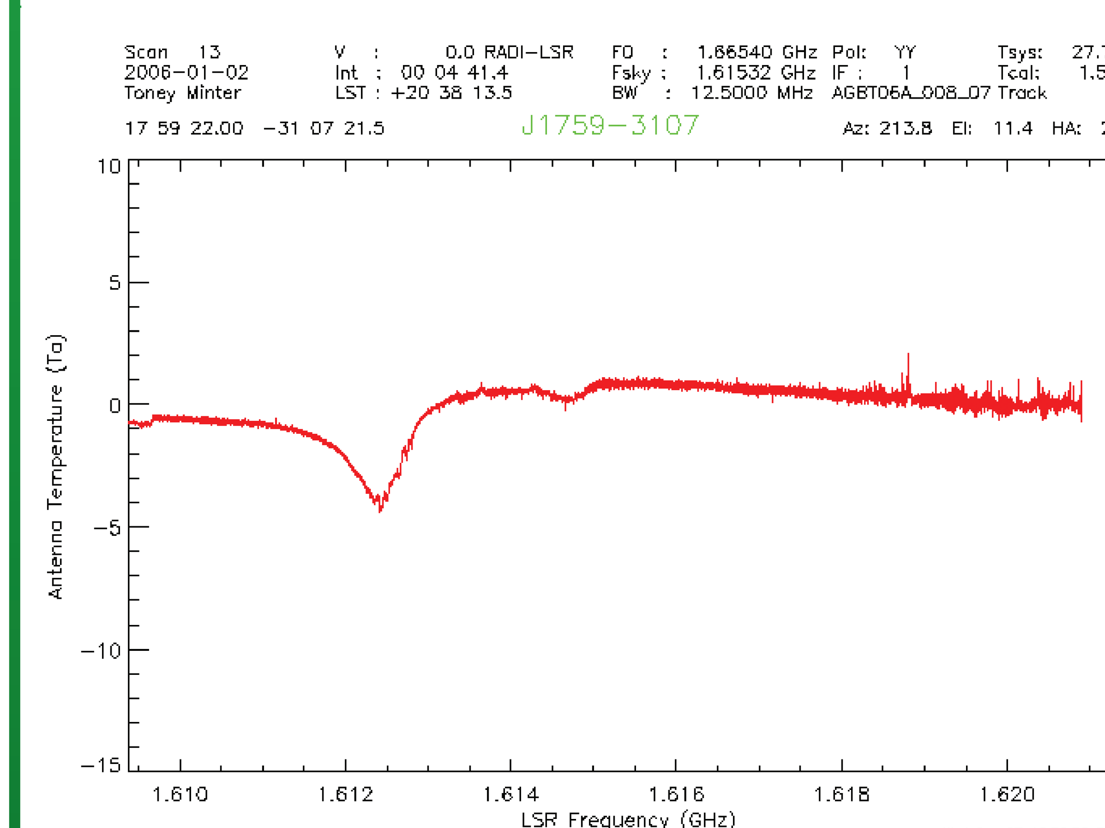


Figure 5: Calibrated and averaged GBT data after auto-flagging.

7. Other GBT Backends

The data processing software for these backends is not supported by in-house efforts.

- VLBA/VLBI
- Pulsar: Spigot, BCMP, Spectral Processor, GASP, CGSR2
- Radar

8. Future Data Processing Plans

- Improved wide-bandwidth calibration
- More automatic flagging tools in GBTIDL
- Visual interactive flagging from GBTIDL
- Ability to import and export data to Casa, the ALMA data analysis package and successor to aips++
- More flexible export from IDL to classic AIPS
- Ability to import CLASS binaries to GBTIDL

References

- GBT: http://www.gb.nrao.edu/GBT/
- GBTIDL: http://gbtidl.sourceforge.net
- aips++: http://aips2.nrao.edu
- casa: http://casa.nrao.edu
- Dyer, K.K, Cornwell, T.J, Maddalen, R.J., Bull. Am. Astr. Soc., 37, 1437.
- Ghigo, F., Maddalena, R.J, NRAO Newsletter, Jan. 2003.
- Lockman, F.J, and Condon, J.J., AJ 129(4), 1968-1977.

The National Radio Astronomy Observatory operated for the National Science Foundation (NSF) by Associated Universities, Inc. (AUI), under a cooperative agreement.