IRAM Memo 2005-1

CLASS evolution: I. Improved OTF support

P. Hily-Blant[1], J. Pety[1,2], S. Guilloteau[3]

1. IRAM (Grenoble)
2. LERMA, Observatoire de Paris
3. L3AB, Observatoire de Bordeaux

Dec, 20th 2005
Version 1.0

Change Record

Revision	Date	Section/ Page affected	Remarks
1	2005-12-20	All	Initial version

Contents

1 CLASS internal data format **4**
1.1 Pointed observation . . . 4
1.2 On–The–Fly observation . . . 4

2 Limitations, ease of use and efficiency **6**
2.1 Philosophy . . . 6
2.2 Implications . . . 6
2.2.1 New OTF data format . . . 6
2.2.2 Memory limitations . . . 6
2.3 Warning . . . 7

3 Large dataset processing **7**
3.1 Basic idea . . . 7
3.2 Listing and Table of Content . . . 7
3.3 Index consistency . . . 9
3.4 Visualization . . . 9
3.5 Sorting . . . 10
3.6 Baseline fitting . . . 10
3.7 Interactive data exploration . . . 12

4 Miscellaneous changes **15**
4.1 New defaults . . . 15
4.2 Command names . . . 15
4.3 Griding and spectra cube visualization . . . 15
4.4 `PLAIT` algorithm . . . 16

5 CLASS90 for beta testers **16**

Abstract

CLASS is a **GILDAS**[1] software for reduction and analysis of (sub)–millimeter spectroscopic data. It is daily used to reduce spectra acquired with the IRAM 30m telescope. **CLASS** is currently used in many other facilities (*e.g.* CSO, HHT, Effelsberg) and it is considered for use by Herschel/HIFI. **CLASS** history started in 1983. As a consequence, it was written in FORTRAN 77 and tailored to reduce pointed observations. On–The–Fly support was added in the 90s but it showed limitation as the quantity of OTF data increased quickly. One year ago, we decided to fully rewrite **CLASS** in FORTRAN 90 with the 3 following goals: 1) clarifying satisfying features with backward compatibility in mind, 2) improving code readability to simplify maintenance and 3) easing reduction of large OTF data sets. This memo is describing the *current* state of affair with a particular emphasis on changes in the program behavior. Future foreseen changes (linked to the increase of receiver instantaneous bandwidth like an improved conversion from frequency to velocity axis) will be described in a future memo.

1 CLASS internal data format

Observations with a single dish telescope may be divided in two categories:

1. Pointed observation: The telescope beam is pointed toward a fixed position of the source during the signal integration. In its simplest form, the scan is composed of a spectrum whose intensity is accumulated during the scan duration. Nothing prevents more complex scan definitions, *e.g.* a scan composed of several shorter integrations, all at the same position on the sky.

2. On-The-Fly (OTF) observation: The telescope beam continuously drifts on source during the integration to make a map of the source. The scan is then composed of set of spectra regularly dumped (typically every 1 second) during a contiguous portion of time (typically 10 minutes). Each dumped spectra corresponds to a different position on the sky.

1.1 Pointed observation

Only two minor changes happened into the data format of a scan containing a single spectra. The data is stored as a header (divided in independent sections) and the data. We have added: 1) the three parameters associated to the descriptive coordinate system (2 parameters for the system center + the system position angle) to be able to go from this descriptive coordinate system to a standard (*e.g.* equatorial) coordinate system and 2) a subscan number. This subscan number is foreseen to always be greater than 1. There is one exception: when **CLASS90** read data in old format, the subscan number is zero and a warning is issued.

1.2 On–The–Fly observation

In **CLASS77**, an OTF scan is stored as a header and a 2 dimensional array containing the intensities of dumped spectra as a function of time plus a given number of columns (named DAPS for Data Associated Parameters) containing header parameters whose values vary during the scan (*e.g.* the position on the sky, date from beginning of scan). Therefore, all the dumped spectra share the same header in **CLASS77**.

In **CLASS90**, each dumped spectra of an OTF scan is stored as a pointed observation with its own header and data. The concept of scan is kept, as all dumped spectra inside an OTF scan share the same scan number. They are tagged by a subscan number whose value is incremented for each new OTF line (both to enable easy selection of a single line inside one OTF scan, and to ensure consistency with the 30m numbering). The dumped spectra are also tagged by a unique observation number. This new OTF data format has several advantages:

[1] `http://www.iram.fr/GILDAS`

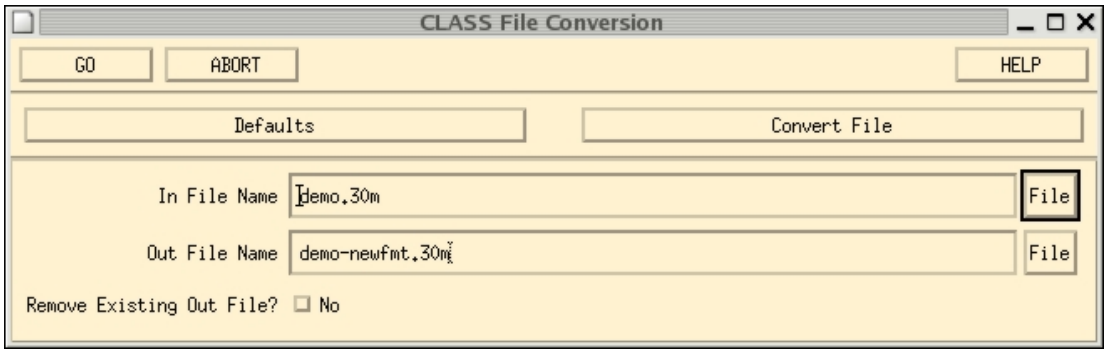

Figure 1: File conversion tool to convert from **CLASS77** to **CLASS90** data format. The action of this widget is equivalent to "`go class-convert infilename outfilename`" command at prompt level.

- It considerably simplifies the source code, as operations on OTF scan are not a special case anymore. While some operations need the information that a dumped spectra is part of an OTF scan, OTF scans may be seen as a collection of independent spectra for many basic operations.
- The granularity of operations is much finer. For instance, access and work on any dumped spectra is now obvious. Moreover, it is obvious in this framework to store the windows used for baseline fitting for every dumped spectra.
- The scan/subscan organization can be generalized to other kind of observation (rasters, cross scans, ...)

For backward compatibility, **CLASS90** will be able to read the old OTF data format. Nevertheless, it will then stubbornly refuse to do anything else with it than to rewrite the data in the new OTF data format. This is easily done with the following commands:

```
LAS90> file in  12co21-oldfmt.30m
LAS90> file out 12co21-newfmt.30m new
LAS90> set line 12co(2-1)
LAS90> find
LAS90> for ientry 1 to found
LAS90>    get n
LAS90>    write
LAS90> next
```

For the user's convenience, **CLASS90** proposes a widget that implements this conversion with several additional safeguards (as avoiding to convert again data already written in new format). Fig. 1 shows this widget available in the main **CLASS90** menu. This widget is just a front-end to a procedure launched by the "`go class-convert infilename outfilename`" command. This procedure can be used in scripts to automate conversion.

Since the advent of the New Control System (NCS) of the 30m in fall 2005, the reading of raw data and the chopper calibration is done by **MIRA** (developed by H. Wiesemeyer) which directly writes data in the **CLASS90** format. For data acquired previously (with the Old Control System), the conversion step is mandatory. Indeed, the reading of raw data format and the chopper calibration were done by **OTFCAL** (maintained by A. Sievers), which still writes OTF scans using the **CLASS77** data format.

2 Limitations, ease of use and efficiency

2.1 Philosophy

Several qualities are desirable for a reduction software: portability, stability, ease of maintenance, ease of use, short learning curve, good documentation, time and storage efficiency, best functionalities, no arbitrary limitations and last but not least backward compatibility. As we are manpower limited, it is impossible to get the perfect software. We thus have to make some compromises to get the closest possible to perfect. For instance, we rewrote **CLASS** in FORTRAN 90 although this may bring short-term instability because this clarifies the program structure and it thus eases future maintenance (key to long term stability). We also favor functionalities over efficiency with the idea that **CLASS** users will be happier to be able to do something a bit inefficiently than to be stuck. Obviously, we always keep storage and time efficiency in mind and we are willing to improve the situation when a widely used feature is too inefficient. Finally, if we make all our effort to have a 100% backward compatibility in data format so that users will be able to do something useful even with even very old data, we can not ensure full backward compatibility on defaults and command names. The easiest way to implement new features (required by improved instrumentation) is sometimes to change defaults and command names though we try to refrain from making those kind of changes without good reasons.

2.2 Implications

2.2.1 New OTF data format

One of the major change in **CLASS90** is the new way OTF data are stored. While advantages have already been described, the main inconvenient is an increase of the size of the data on disk and RAM memory by at most 20-30%, due to the new individual headers (Note that this increase becomes negligible for spectra with a large number of channels).

Data access time is less problematic as the data has always been buffered by **CLASS**. In fact, we choose this OTF data format to have a much more user-friendly approach to OTF processing. Indeed, it maybe (but still has to be proven...) that the CPU time will be a bit larger with **CLASS90** than with **CLASS77** for perfect data. However, our main goal is to decrease the human time spent on data reduction to deal with the unavailable problems.

2.2.2 Memory limitations

Almost all processing steps are available spectrum per spectrum because this is a very powerful way to work around the limitation of your computer RAM memory. There are two main exceptions:

- When opening a file, **CLASS** is buffering information (source name, line name, telescope name, scan number, offset, etc...) on each observations of the file to speed next `find` commands. This buffer has a fixed sized to avoid code complexity. This limits the number of observations that **CLASS90** can read/write during one session. The default maximum number of observations is currently set to 100 000 which amounts to about 5 MB of RAM memory. This number can be set to a larger value through the `CLASS_IDX_SIZE` logical variable in your `$HOME/.gag.dico` configuration file as it is probable that the default value is too small with the largest maps observed today.

- In **CLASS90**, it is also possible to load a whole index as a single 2D array for further visualization and processing (see below). There is no limitation on the number of dumped spectra loadable, apart from the previous limitation and the RAM memory of your computer.

Except from the index buffer, all other **CLASS90** buffers are now dynamically allocated, in particular the buffer R & T containing the spectra data. This means that the number of channels of a spectrum is now unlimited which is a useful feature for line surveys.

2.3 Warning

When processing a small number of spectra, **CLASS90** will be quick on whatever kind of computer. Now, if **CLASS90** users needs to process 300 000 dumped spectra, they should then be prepared to use a powerful computer (with lot's of RAM to avoid swapping) and to wait during processing, whichever software (in particular whichever version of **CLASS**) they are using.

Moreover, flexibility is favored in **CLASS90**. This means that the same things may be done in many different, more or less efficient ways. It will take time to the **CLASS90** community to learn what should be or not be done to ensure efficiency. For instance, loading 300 000 with the `LOAD` command (see below) on a laptop is probably going to take a while (if possible at all due to limitation in RAM memory). However, this possibility does not even exist within **CLASS77**.

3 Large dataset processing

It is today possible with the IRAM–30 m to map a square degree field in ^{12}CO (J=2–1). As an order of magnitude, this gives a final spectra cube of about 10^6 spectra with a slight oversampling of $4''$.

3.1 Basic idea

An observer who has just spent a few hours doing OTF observation of the same source may want to visualize all the dumped spectra at once even though they do not belong to the same scan. This was impossible in **CLASS77**. In **CLASS90**, it is now possible to load all the individual spectra currently in the index as a 2D array for future work, in particular visualization.

3.2 Listing and Table of Content

Before visualizing, the user needs to know what kind of data is available. The `LIST` command is commonly used to easily see the content of the current index. However, this command outputs one line per observation in **CLASS90** which is useless when dealing with thousands of observations. The `LIST /SCAN` command reintroduces the **CLASS77** way of listing an index, *i.e.* one line per scan and setup (*i.e.* unique combination of source, line and telescope names). `LIST /SCAN /BRIEF` lists the scan with the number of associated observations. These three possibilities respectively give

```
LAS90> list
I-LISTE, Current index:
 197; 3 B0355+508    12CO(1-0)    30M-V01-A100   -109.1   -100.0 Eq  9608; 1
 198; 3 B0355+508    12CO(1-0)    30M-V01-A100   -106.1   -100.0 Eq  9608; 1
 199; 3 B0355+508    12CO(1-0)    30M-V01-A100   -103.0   -100.0 Eq  9608; 1
 200; 3 B0355+508    12CO(1-0)    30M-V01-A100   -100.5   -100.0 Eq  9608; 1
 201; 3 B0355+508    12CO(1-0)    30M-V01-A100    -97.5   -100.0 Eq  9608; 1
 ...
 270; 3 B0355+508    12CO(1-0)    30M-V02-B100   -109.1   -100.0 Eq  9608; 1
 271; 3 B0355+508    12CO(1-0)    30M-V02-B100   -106.1   -100.0 Eq  9608; 1
 272; 3 B0355+508    12CO(1-0)    30M-V02-B100   -103.0   -100.0 Eq  9608; 1
 273; 3 B0355+508    12CO(1-0)    30M-V02-B100   -100.5   -100.0 Eq  9608; 1
 274; 3 B0355+508    12CO(1-0)    30M-V02-B100    -97.5   -100.0 Eq  9608; 1
 ...
 343; 3 B0355+508    12CO(2-1)    30M-V03-A230   -109.1   -100.0 Eq  9608; 1
 344; 3 B0355+508    12CO(2-1)    30M-V03-A230   -106.1   -100.0 Eq  9608; 1
 345; 3 B0355+508    12CO(2-1)    30M-V03-A230   -103.0   -100.0 Eq  9608; 1
 346; 3 B0355+508    12CO(2-1)    30M-V03-A230   -100.5   -100.0 Eq  9608; 1
 347; 3 B0355+508    12CO(2-1)    30M-V03-A230    -97.5   -100.0 Eq  9608; 1
```

```
 ...
 416; 3 B0355+508    12CO(2-1)    30M-V04-B230   -109.1   -100.0 Eq  9608; 1
 417; 3 B0355+508    12CO(2-1)    30M-V04-B230   -106.1   -100.0 Eq  9608; 1
 418; 3 B0355+508    12CO(2-1)    30M-V04-B230   -103.0   -100.0 Eq  9608; 1
 419; 3 B0355+508    12CO(2-1)    30M-V04-B230   -100.5   -100.0 Eq  9608; 1
 420; 3 B0355+508    12CO(2-1)    30M-V04-B230    -97.5   -100.0 Eq  9608; 1
 ...
 661; 3 B0355+508    12CO(1-0)    30M-V01-A100   -109.8    -90.0 Eq  9609; 1
 662; 3 B0355+508    12CO(1-0)    30M-V01-A100   -106.7    -90.0 Eq  9609; 1
 663; 3 B0355+508    12CO(1-0)    30M-V01-A100   -103.6    -90.0 Eq  9609; 1
 664; 3 B0355+508    12CO(1-0)    30M-V01-A100   -101.2    -90.0 Eq  9609; 1
 665; 3 B0355+508    12CO(1-0)    30M-V01-A100    -98.1    -90.0 Eq  9609; 1
 ...
 734; 3 B0355+508    12CO(1-0)    30M-V02-B100   -109.8    -90.0 Eq  9609; 1
 735; 3 B0355+508    12CO(1-0)    30M-V02-B100   -106.7    -90.0 Eq  9609; 1
 736; 3 B0355+508    12CO(1-0)    30M-V02-B100   -103.6    -90.0 Eq  9609; 1
 737; 3 B0355+508    12CO(1-0)    30M-V02-B100   -101.2    -90.0 Eq  9609; 1
 738; 3 B0355+508    12CO(1-0)    30M-V02-B100    -98.1    -90.0 Eq  9609; 1
 ...
 807; 3 B0355+508    12CO(2-1)    30M-V03-A230   -109.8    -90.0 Eq  9609; 1
 808; 3 B0355+508    12CO(2-1)    30M-V03-A230   -106.7    -90.0 Eq  9609; 1
 809; 3 B0355+508    12CO(2-1)    30M-V03-A230   -103.6    -90.0 Eq  9609; 1
 810; 3 B0355+508    12CO(2-1)    30M-V03-A230   -101.2    -90.0 Eq  9609; 1
 811; 3 B0355+508    12CO(2-1)    30M-V03-A230    -98.1    -90.0 Eq  9609; 1
 ...
 880; 3 B0355+508    12CO(2-1)    30M-V04-B230   -109.8    -90.0 Eq  9609; 1
 881; 3 B0355+508    12CO(2-1)    30M-V04-B230   -106.7    -90.0 Eq  9609; 1
 882; 3 B0355+508    12CO(2-1)    30M-V04-B230   -103.6    -90.0 Eq  9609; 1
 883; 3 B0355+508    12CO(2-1)    30M-V04-B230   -101.2    -90.0 Eq  9609; 1
 884; 3 B0355+508    12CO(2-1)    30M-V04-B230    -98.1    -90.0 Eq  9609; 1
LAS90> list /scan
I-LISTE, Current index:
 B0355+508    12CO(1-0)    30M-V01-A100   -109.1:+108.2   -100.0 Eq  9608; 73
 B0355+508    12CO(1-0)    30M-V02-B100   -109.1:+108.2   -100.0 Eq  9608; 55
 B0355+508    12CO(2-1)    30M-V03-A230   -109.1:+108.2   -100.0 Eq  9608; 73
 B0355+508    12CO(2-1)    30M-V04-B230   -109.1:+108.2   -100.0 Eq  9608; 73
 B0355+508    12CO(1-0)    30M-V01-A100   -109.8:+107.5    -90.0 Eq  9609; 73
 B0355+508    12CO(1-0)    30M-V02-B100   -109.8:+107.5    -90.0 Eq  9609; 73
 B0355+508    12CO(2-1)    30M-V03-A230   -109.8:+107.5    -90.0 Eq  9609; 73
 B0355+508    12CO(2-1)    30M-V04-B230   -109.8:+107.5    -90.0 Eq  9609; 73
LAS90> list /scan /brief
I-LISTE, Current index:
 9608; 274   9609; 292
```

It is also useful to get all the available setups of a file at a glance. This is available through the "`list in /toc`" command

```
LAS90> file in 12co21-newfmt.30m
LAS90> list in /toc
I-LIST, Input index:
 Number of sources......   1
     B0355+508         17210
 Number of Lines........   2
```

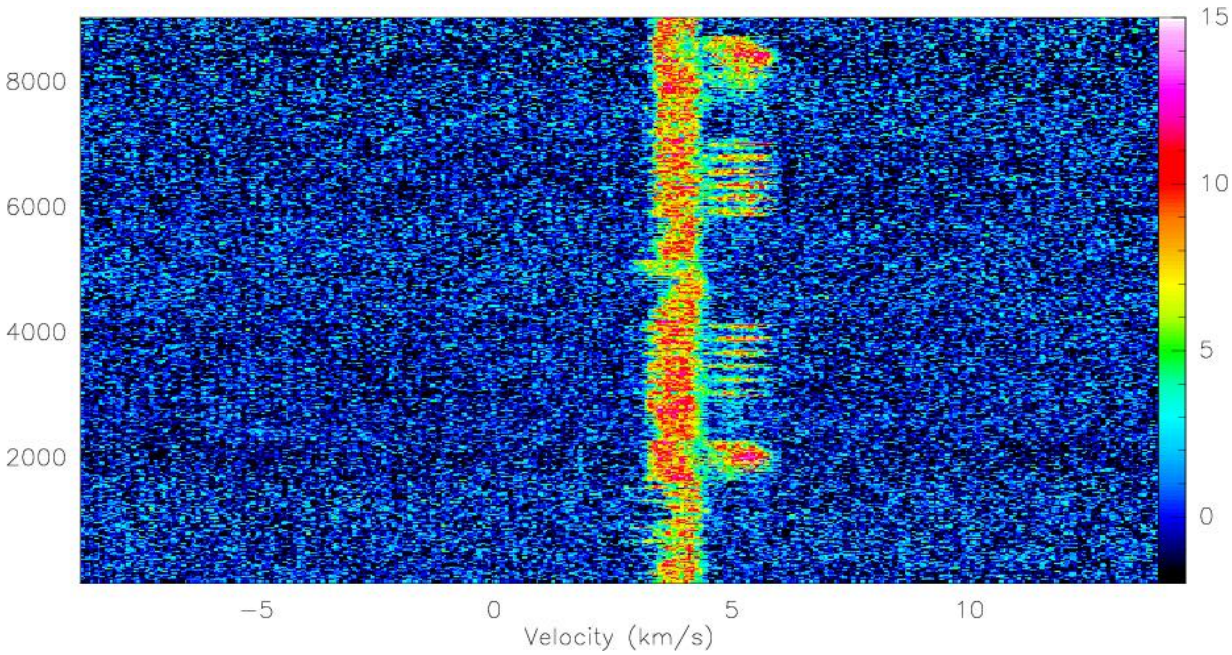

Figure 2: Time series of the intensity as a function of velocity of 9000 dumped spectra part of 200 OTF scans. Only 7 commands are needed to obtain this result, 4 of which are mandatory and 2 of which just there to improve the visual aspect.

```
    12CO(1-0)          8596
    12CO(2-1)          8614
Number of backends.....   4
    30M-V01-A100       4307
    30M-V02-B100       4289
    30M-V03-A230       4307
    30M-V04-B230       4307
Number of setups.......   4
    B0355+508    12CO(1-0)    30M-V01-A100       4307
    B0355+508    12CO(1-0)    30M-V02-B100       4289
    B0355+508    12CO(2-1)    30M-V03-A230       4307
    B0355+508    12CO(2-1)    30M-V04-B230       4307
```

3.3 Index consistency

Loading a whole index as a 2D array implies that all the spectra are consistent (basically the same coordinate system and the same frequency axis) to obtain meaningful results. A new command, named `LAS\CONSISTENCY`, was introduced to check the consistency of all the spectra of the current index. The source name, the source position, the line name and the frequency axis are checked by default. It is possible to avoid one or several of those checks with the `NOCHECK` option. The commands which work on the whole index (*e.g.* `LOAD` or `TABLE`) automatically trigger the check of the index consistency, if this check was not already done previously.

3.4 Visualization

The following list of commands is all you need to plot as a single image (see Fig. 2) 9000 dumped spectra part of 200 OTF scans observed in $\sim$ 4 hours.

```
LAS90> lut rainbow3
LAS90> file in 12co21-newfmt.30m
LAS90> find
LAS90> load
LAS90> set mode y -2 15
LAS90> plot /index
```

3.5 Sorting

In the previous visualization, the dumped spectra were order according to their observation number, which most often corresponds to the observing time sequence. However, it often happens that two sequential OTF scans belongs to 2 far-away part of the same source. It is thus desirable to be able to sort dumped spectra in the index by coordinates. In **CLASS90**, the "`set sort keyword`" command defines the sorting that will be applied when the index will be built by the next `find` command. In particular, the `keyword` may be `lambda` or `beta`. Fig. 3 shows the results of the following commands

```
LAS90> set mode y 0 8
LAS90> set angle second
LAS90> set sort beta
LAS90> for ilambda -470 to -260 by 30
LAS90>    find /range 'ilambda' 'ilambda+10' * *
LAS90>    if (found.ne.0) then
LAS90>       load
LAS90>       plot /index
LAS90>       g\draw text 0 1 'ilambda'"` < lambda < "'ilambda+10'"`" 5 /char 8
LAS90>       list
LAS90>       pause
LAS90>    endif
LAS90> next
```

which enables to view the dumped spectra with a good continuity in intensity.

3.6 Baseline fitting

Other operations than visualization may benefit from the definition of the 2D array. The first coming in mind is baseline fitting. It is well known that this requires the separation of the spectrum channels between signal and baseline, in other words the definition of baseline windows. When dealing with a small number of spectra, the windows are often defined separately on each spectrum. When dealing with thousands of spectra, we still need to define windows adapted to each individual spectrum as large velocity gradients (*e.g.* in galaxies) may quickly move in frequency the separation between signal and baseline from one spectrum to another. However, defining windows on each individual spectrum is both impractical and inefficient as this does not take into account spatial homogeneity (really useful in case of moderately weak lines). We thus enable the definition of spectral windows directly on the 2D plot with polygons.

The following set of commands are used to define the polygon and to fit the baselines

```
LAS90> file out 12co21-newfmt-base.30m new
LAS90> plot /index
LAS90> set window /polygon
LAS90> for iobs 1 to found
LAS90>    get next
LAS90>    base
LAS90>    write
LAS90> next
```

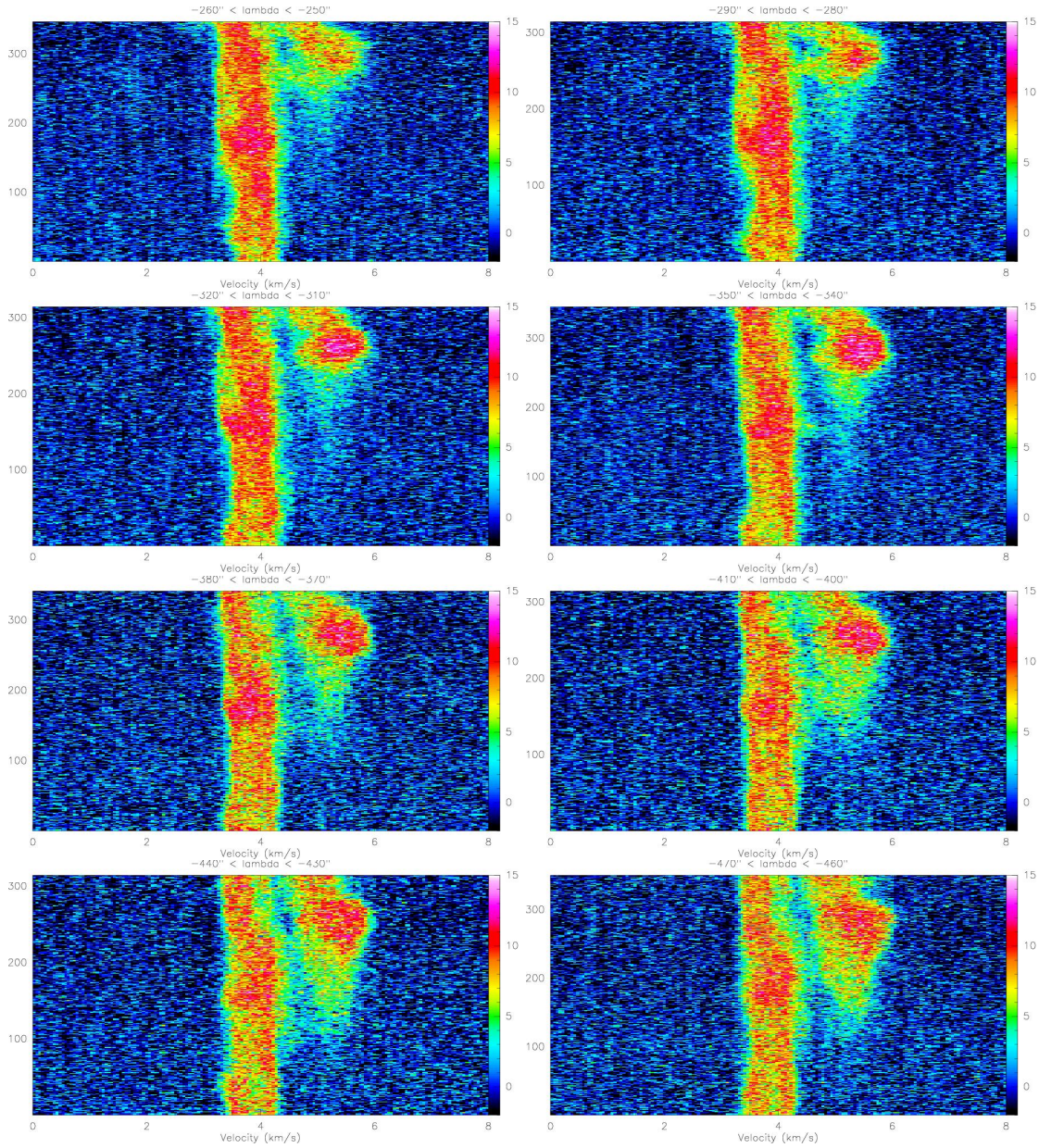

Figure 3: Beta sorted series of dumped spectra intensities as a function of the velocity. For each panel, only the spectra with a lambda offset belonging to the same spatial resolution bin is shown. This is the way to ensure the best possible intensity continuity from one to another dumped spectra.

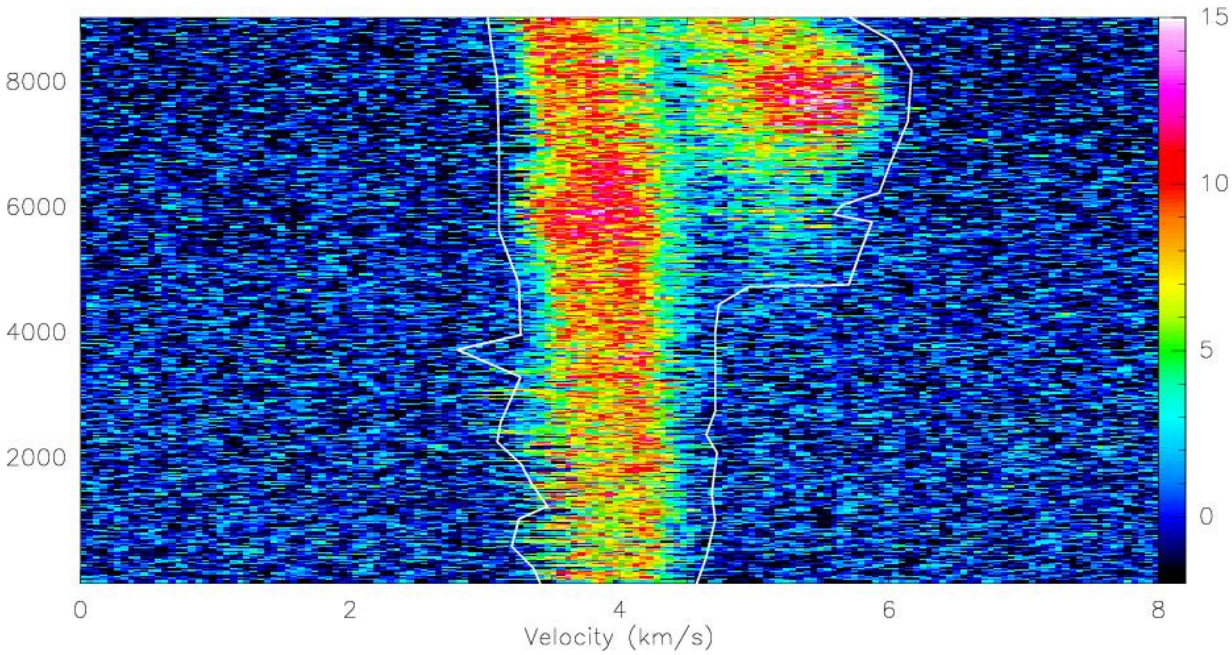

Figure 4: Beta sorted series of dumped spectra intensities as a function of the velocity. All the lambda offsets are shown here, implying the presence of horizontal stripes in the image. This representation of the data nevertheless enable a very quick (even though not optimal) first data reduction by the definition of baseline windows through a polygon drawing.

Up to 21 independent polygons may be defined. The polygon boundaries are directly converted to individual windows for each dumped spectra. Those windows are stored in the `set%window` **SIC** array which is then used by the `base` and `write` commands. In the end, each spectrum will have its own baseline window stored in its header. Each polygon is also stored in a separate file for the user's convenience.

3.7 Interactive data exploration

Scripting is important for memory and/or for automating data processing. However, repeatedly typing the same commands quickly becomes cumbersome in particular when discovering new data. Interactive exploration of data is now available through the following set of commands

```
LAS90> file in 12co21-newfmt-base.30m
LAS90> set source YOUR-SOURCE
LAS90> set line 12CO(2-1)
LAS90> set tele 30M-V03-A230
LAS90> go explore
```

"`go explore`" enables loop visualization of data scan by scan or by offset range (as in section 3.5). It is possible to visualize a 2D image or the average of the observations. Zoom are easily accessible as well as popups of spectra and drifts. An up-to-date summary of the possibility is available by typing `input explore`. "`go explore`" is best used in conjunction with the possibility to easily select a consistent setup. To do this (*i.e.* exploring a subset of its input data file), the user must select this setup through the `set` command because the "`go explore`" script issues many intermediate `find` commands. Both functionalities (selection and exploration) are merged in the `Explore Data File` widget of the **CLASS90** main menu. Examples are given in Figs. 5 to 7.

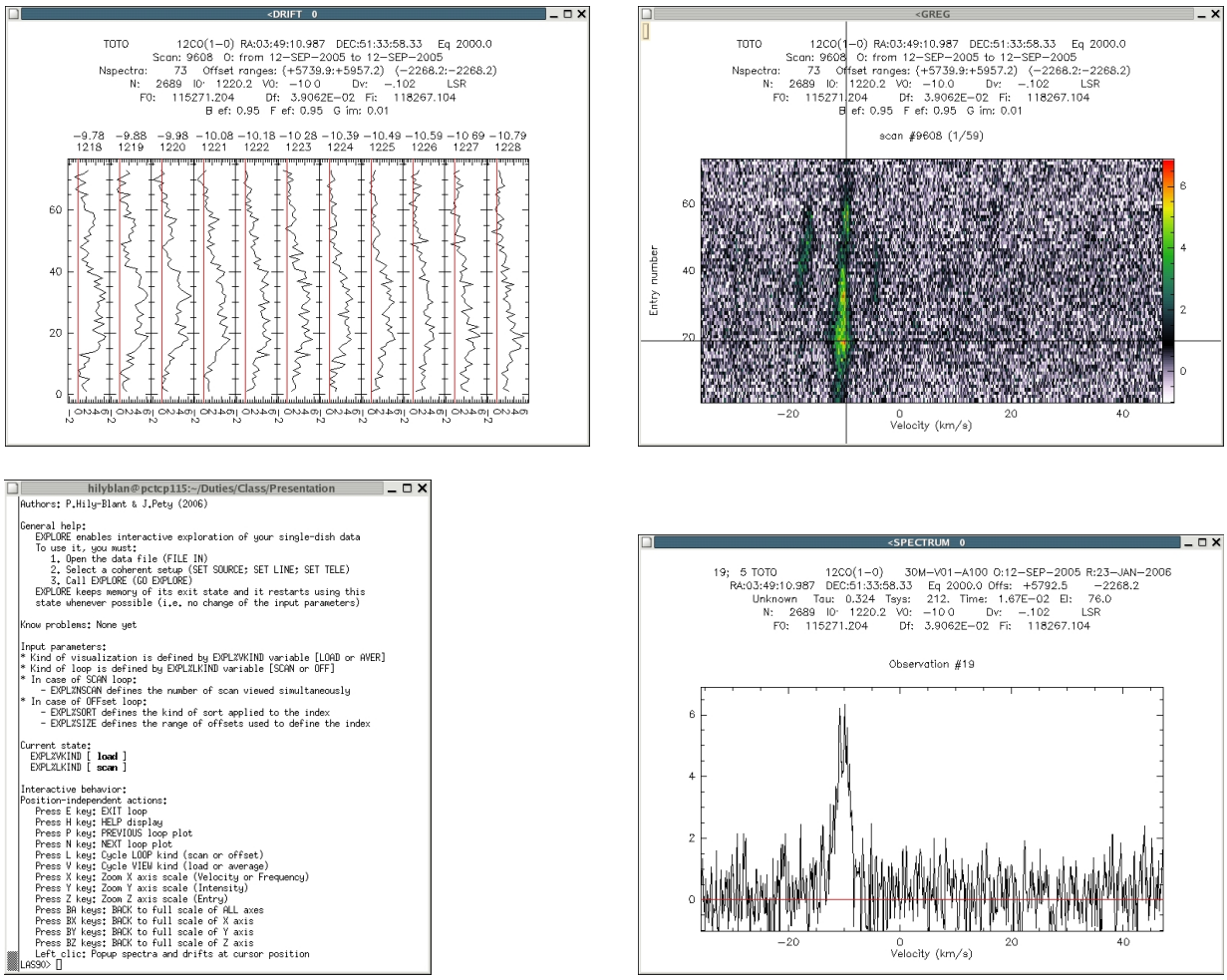

Figure 5: Data exploration by selecting `Explore Data File` from the **CLASS90** main menu or by typing `go explore` at the **CLASS90** prompt. Upper right: The 73 dumped spectra of scan 9608. Upper left: Intensity as a function of entry number, in ten contiguous velocity channels centered on the channel shown as the vertical line of the upper right panel. Bottom right: Spectrum corresponding to entry number 19 defined as the horizontal line of the upper right panel. Bottom left: On-line documentation of this interactive tool.

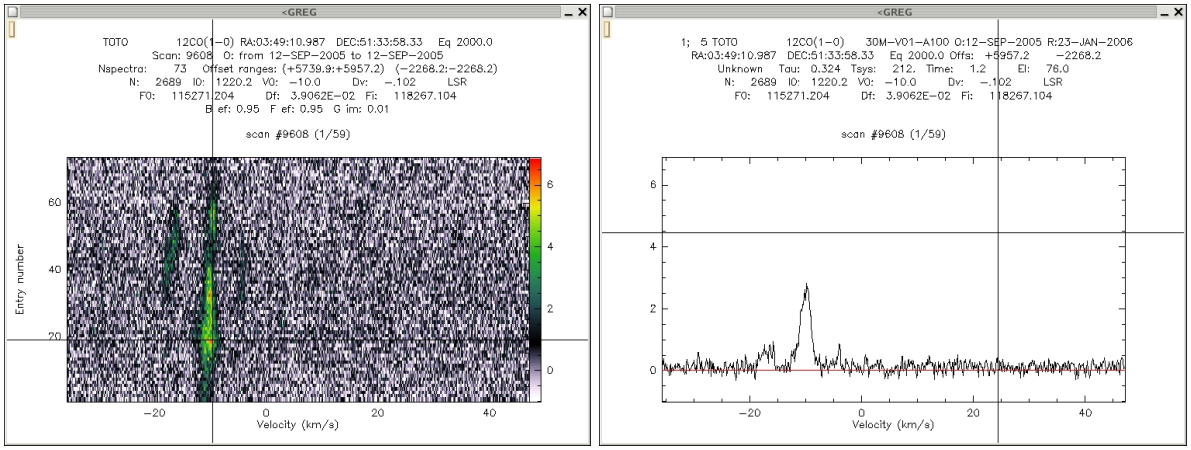

Figure 6: Data exploration by scan. Left: 2D image of all dumped spectra of the scan. Right: Averaged spectrum computed on all dumped spectra of the scan.

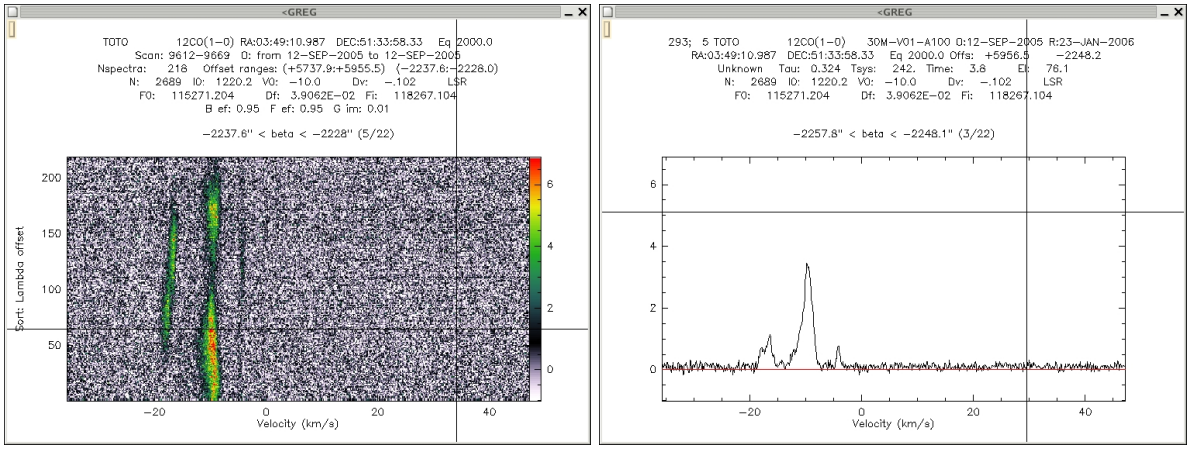

Figure 7: Same as Fig 6 except that the displayed data belong to a beta offset range.

4 Miscellaneous changes

4.1 New defaults

A few defaults has been changed **CLASS77** and **CLASS90**. They are summarized in Table 1. We are

Default kind	Old names	New names
Extension	.bur	.30m
Angular unit	arcminute	arcsecond
Epoch	1950.00	2000.00

Table 1: Correspondence between changed defaults in **CLASS77** and **CLASS90**.

distribution a **CLASS** procedure named `old-set-defaults.class` which reset the `set` default in the **CLASS77** way.

4.2 Command names

Due to its long history, **CLASS77** command name were sometimes awkward, *e.g.* the `GAUSS` command which was launching a fit even though the fitting function was not Gaussian! We made the minimum number of changed to avoid disturbing too much long standing habits of **CLASS** users. We are also furnishing a **CLASS** procedure named `old-command-names.class` which aliases new and old names through the use of **GILDAS** symbols. This is distributed for maximum backward compatibility, *e.g.* to enable use of **CLASS77** within **CLASS90** (this is not foolproof though). However, it is clear that old command names are obsolete and *we advise users against the use of the aliases.* In particular, users' new procedures should use new command names. Table 2 displays an exhaustive list of correspondence between old and new names. The fit–related commands have been gathered into a separate language named `FIT`. The `GAUSS` and `FIT` commands have been replaced by the more explicit `MINIMIZE` and `VISUALIZE` names. For logical reasons, `SUM` has been replaced by the `AVERAGE`. Due to the new way of dealing with OTF data, the `RECORD` command is now obsolete and will disappear soon. The `GRID` command is also obsolete as its functionalities have been redistributed and extended in the `TABLE` and `XY_MAP` commands (cf. section 4.3). Finally, the `CFITS` language has been replaced by a single `LAS\FITS` command. Indeed, most of the `CFITS` commands were customized for tapes which are an obsolescent storage medium.

Old names	New names
`ANALYSE\GRID`	Obsolete (replaced by `ANALYSE\TABLE` and `MAP\XY_MAP`)
`ANALYSE\DISPLAY`	`FIT\DISPLAY`
`ANALYSE\FIT`	`FIT\VISUALIZE`
`ANALYSE\GAUSS`	`FIT\MINIMIZE`
`ANALYSE\ITERATE`	`FIT\ITERATE`
`ANALYSE\KEEP`	`FIT\KEEP`
`ANALYSE\LINES`	`FIT\LINES`
`ANALYSE\METHOD`	`FIT\METHOD`
`ANALYSE\RESIDUAL`	`FIT\RESIDUAL`
`LAS\SUM`	`LAS\AVERAGE`
`LAS\RECORD`	Obsolete (not useful anymore)

Table 2: Correspondence between command names in **CLASS77** and **CLASS90**.

4.3 Griding and spectra cube visualization

The functionalities of the old `ANALYSE\GRID` command has been redistributed and extended in several commands:

- `ANALYSE\TABLE` creates a table containing the offsets, weights and intensities of all the dumped spectra. A check of the consistency of the observations in the current index is performed at this step, if not already done before.
- `MAP\XY_MAP` grids the dumped spectra from the table to an lmv cube. An image of the associated weights is also produced for further processing like optimal combination of several data cubes.
- Moreover, all the plotting capabilities of **GREG** program has been imported inside **CLASS** so that the user can directly visualize the data cube of the gridded spectra. For instance, Fig. 8 has been obtained with the following sequence of commands:

```
LAS90> file in 12co21-newfmt.30m
LAS90> find
LAS90> table 12co21 new
LAS90> xy_map 12co21
LAS90> let name 12co21
LAS90> let type lmv
LAS90> go view
```

  The "`go view`" scripts enables interactive visualization of a spectra cube. Channel maps may be produced through the `go bit` command and position-velocity diagrams through the `go xv` and `go vy` commands.

4.4 `PLAIT` algorithm

The `PLAIT` algorithm "combine" two spectra cubes resulting from OTF observation with orthogonal scanning directions into a single spectra cube. It works in the Fourier plane and it reduces the striping due to receiver and atmospheric instabilities which inevitably show up in OTF maps. A version of this algorithm has been implemented as a **GILDAS** task named `PLAIT` from a previous version by C.Nieten (itself elaborated on original ideas from the NOD2 package).

5 CLASS90 for beta testers

The current default version of **CLASS** is **CLASS77**. Soon enough, the default version will swap to **CLASS90**, *i.e.* users will have access directly to **CLASS90** when calling **CLASS** from the shell prompt. There will be a warning that this is a new **CLASS** with modified features and that the old **CLASS** is still available through the **CLASS77** name in case of a problem with **CLASS90**. **CLASS77** will stay about one year after the swap to ensure good stability of **CLASS90**. However, *no* support will be given anymore to **CLASS77**.

Before the swap happens, you can become beta testers of **CLASS90** if you are interested by the new features. Beta testers should be able to quickly get bug fixes in their version. We thus recommend they use anonymous CVS in the following way:

```
shell-prompt> export CVSROOT=:pserver:anonymous@netsrv1.iram.fr:/CVS/GILDAS
shell-prompt> cvs co -r feb06 -d gildas-src-feb06 gildas
shell-prompt> cd gildas-src-feb06
shell-prompt> source admin/gildas-env.sh
shell-prompt> make
shell-prompt> make install
shell-prompt> cd packages/class90
shell-prompt> cvs up -r class90-stable
shell-prompt> make clean
```

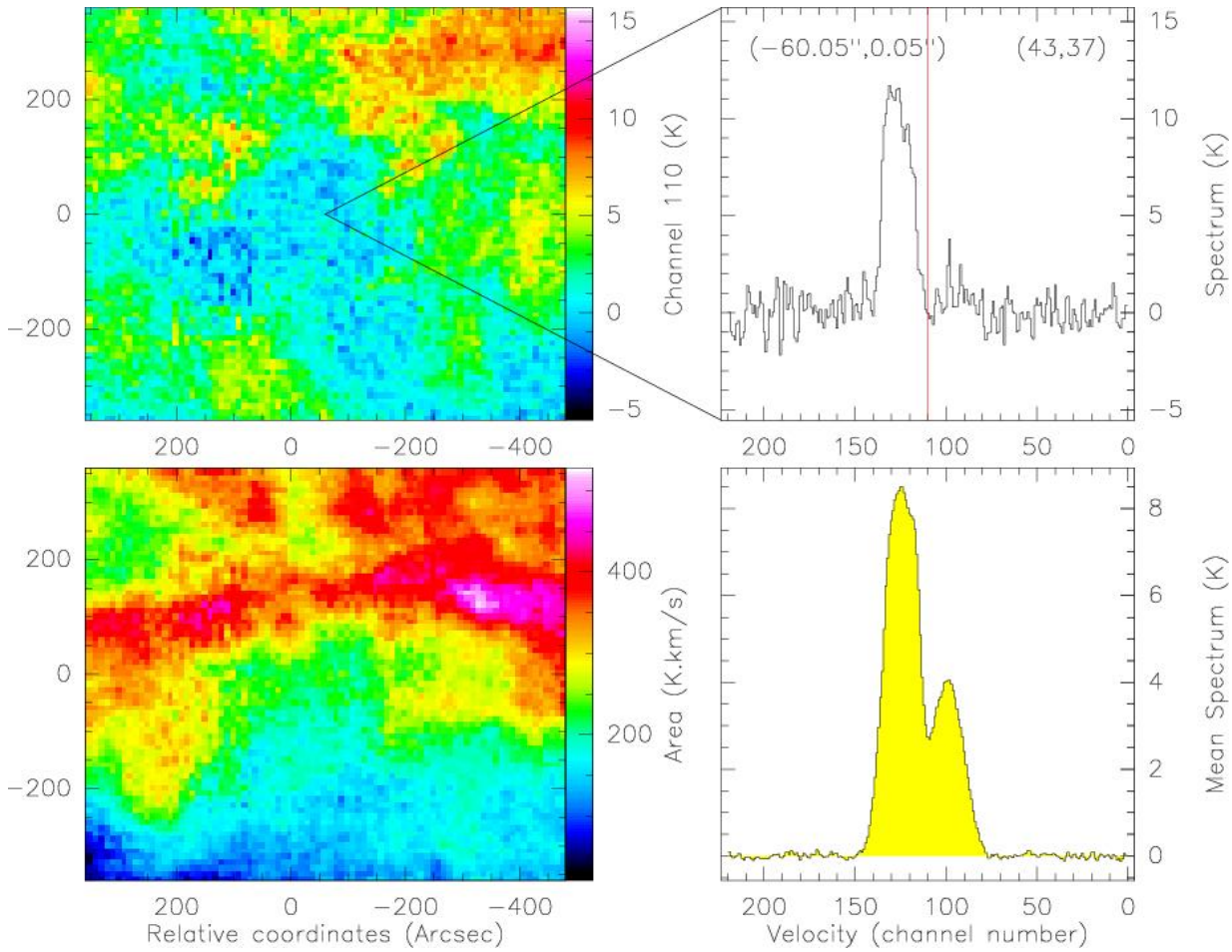

Figure 8: Screen-shot of result of the `go view` command. The top,left panel is the channel map corresponding to velocity shown as a vertical red line in the top,right panel. The top,right panel displays the spectrum at the position localized on the top,right panel. The bottom,left panel shows the line emission integrated over the velocity range appearing in yellow on the bottom,right panel. This last panel displays the spectrum averaged over the whole map. `go view` is an interactive command with many more features. Please, type `h` to display the complete help.

```
10 shell-prompt> make
11 shell-prompt> make install
```

Line 1 and 2 create a directory name `gildas-src-feb06` with the feb06 "stable" monthly release. Lines 3 to 6 are the standard way to install **GILDAS**. As **CLASS90** is evolving quickly, the **CLASS90** version shipped in a **GILDAS** monthly release may be unstable. We thus recommend that you go to the **CLASS90** directory (line 7), update it to a "stable" version (line 8), compile and install it (lines 9 to 11).

Beta testers also refer to the manual which is being fully updated. Bug reports and suggestions of improvements should be send to `gildas@iram.fr`. We will try to fix bugs quickly. We will consider the feasibility of all suggestions on longer timescales. When a bug is fixed, here are the steps to update **CLASS90**:

```
1 shell-prompt> export CVSROOT=:pserver:anonymous@netsrv1.iram.fr:/CVS/GILDAS
2 shell-prompt> cd gildas-src-feb06
3 shell-prompt> source admin/gildas-env.sh
4 shell-prompt> cd packages/class90
5 shell-prompt> cvs up -r class90-stable
6 shell-prompt> make
7 shell-prompt> make install
```

Line 5 is the line which will update the class90 directory. Be sure to type it only in the `gildas-src-feb06/packages/class90` directory to ensure that only **CLASS90** is updated because nothing ensure that other parts of **GILDAS** are "stable" between 2 monthly releases.

Have fun!