# IRAM Memo 2009-4

# Averaging spectra with **CLASS**

S. Bardeau[1], J. Pety[1,2]

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

April, 11th 2011
Version 1.1

**Abstract**

**CLASS90** (hereafter **CLASS**) provides a set of commands capable to average two or more spectra. They provide many averaging modes, presented hereafter in this document. The different modes often imply to perform internally and silently some important computations, namely resampling and weighted average. Combining these operations at the same time may imply some non-trivial effects described here.

On October 2008, it appeared that some particular combinations of the tunable modes were not behaving as expected (either in **CLASS90** and in **CLASS77**). A complete cleaning and factorization of the algorithm was performed, associated to exhaustive tests of all combinations. A test suite was also provided to check the output of each commands and modes.

Following this maintenance of the code, and the use of these capabilities to concatenate the new EMIR[1] spectra, it was decided to write this document in order to keep a trace of all the methods applied. This was also the occasion to investigate deeply in the code and to examine the effects which can occur during all the possible processings.

Keywords: `AVERAGE`, `ACCUMULATE`, resampling, weighting, concatenation
Related documents: IRAM memo 2005-1: **CLASS** *evolution: I. Improved OTF support*

[1] see `http://www.iram.es/IRAMES/mainWiki/EmirforAstronomers`

# Contents

**1 CLASS commands** **3**
1.1 Main commands . . . 3
1.1.1 ACCUMULATE . . . 3
1.1.2 AVERAGE . . . 3
1.2 Tuning of the addition . . . 4
1.2.1 Alignment . . . 4
1.2.2 Weighting type . . . 4
1.2.3 Bad channels . . . 4
1.2.4 Position matching . . . 4
1.2.5 Calibration matching . . . 5

**2 Status of the CLASS releases** **5**
2.1 Bug fixes . . . 5
2.1.1 `Dec08` . . . 5
2.1.2 `Jul09` . . . 5
2.2 Test suites . . . 6

**3 Algorithm** **6**
3.1 Setting the weights . . . 6
3.2 Definition of the output spectrum . . . 7
3.3 Aligned spectra . . . 8
3.4 Non-aligned spectra . . . 8
3.4.1 Resampling . . . 8
3.4.2 Channel weight of a resampled spectrum . . . 10
3.4.3 Output spectrum . . . 11
3.5 Data not available / bad channels . . . 12
3.6 Controls . . . 12
3.7 Output parameters . . . 12
3.7.1 `GENERAL` section . . . 13
3.7.2 `SPECTROSCOPIC` section . . . 13
3.7.3 `BASE` section . . . 14
3.7.4 `HISTORY` section . . . 14
3.8 Concatenation . . . 14

**4 Discussion** **14**

**5 Conclusion** **15**

# 1 CLASS commands

Here is a summary of the commands and their online helps which can be used in this context.

## 1.1 Main commands

Two main commands are able to add spectra in **CLASS**: `ACCUMULATE` and `AVERAGE`. `ACCUMULATE` works on the two $R$ and $T$ **CLASS** buffers, whereas `AVERAGE` can combine all the spectra in the current index.

### 1.1.1 ACCUMULATE

```
LAS\ACCUMULATE

ACCUMULATE is used to integrate step by step an ensemble of spectra.
The T spectrum is added to the R spectrum with the current weights. R
thus contains the current sum, and T the last sum. As command AVERAGE,
ACCUMULATE checks for the positional coincidence and calibration homo-
geneity according to SET MATCH and SET CALIBRATION respectively. Aligne-
ment of spectra is checked according to SET ALIGN command. Accumulated
spectra may have different spectral resolution. In this latter case, the
final spectrum has the coarsest resolution of the two input spectra, and
each channel will be the weighted average of the average values of each
spectrum in the resulting channel. It is however preferable to accumu-
late spectra which have the same resolution.

For EQUAL weights, ACCUMULATE computes the sum of the two spectra, al-
lowing addition of an ensemble of spectra after division by the total
number of spectra.

For other weightings, ACCUMULATE computes the average of the two spec-
tra.

ACCUMULATE also works on Continuum drift. The alignement may be Channel
or Position in this case.
```

Please consider that `ACCUMULATE` behaves differently if using an `EQUAL` weight or another one: this provides to the user a way to perform its own average by first adding the spectra and then dividing them at its will.

### 1.1.2 AVERAGE

```
LAS\AVERAGE [/NOMATCH]

Average all the spectra of the current index using the current weighting
function (see SET WEIGHT) and alignment mode (see SET ALIGN). The homo-
geneity of calibration is checked according to SET CALIBRATION. Bad
channels are handled according to SET BAD. A sum may be interrupted by
<^C>, but the results is then undefined.

The position coincidence of consecutive spectra will not be checked if
option /NOMATCH is present. Else, this check relies on the current SET
[NO]MATCH status. See SET MATCH to build an index with a given position-
ing tolerance.
```

## 1.2 Tuning of the addition

The behavior of the two `ACCUMULATE` and `AVERAGE` commands can be tuned with a set of associated parameters. The commands to change them are presented hereafter.

### 1.2.1 Alignment

```
LAS\SET ALIGN Type Range
```

```
Defines the way the spectra to be added are aligned (Type) and  combined
(Range). Possible combinations are:
Type: C[hannel]      for spectra and drift
      V[elocity]     for spectra only
      F[requency]    for spectra only
      P[osition]     for drift only
Range: I[ntersect] or C[omposite]
Default is SET TYPE C I. Note: This means that continuum drifts in oppo-
site directions are not properly added: ALIGN Position must be specified
for that.
```

### 1.2.2 Weighting type

```
LAS\SET WEIGHT type
```

```
Weighting to be used for summations: T[ime],  E[qual]  or  S[igma]  (for
1/sigma**2).  The  default  is TIME. Sigma is not recommended unless you
just made a baseline fit before. Equal weight behaves differently in AV-
ERAGE and ACCUMULATE (which produces the sum).
```

### 1.2.3 Bad channels

```
LAS\SET BAD Check
```

```
Specify  the way ACCUMULATE and AVERAGE behaves in the case of bad chan-
nels. Check may be OR or AND. Default is OR, i.e. the output channel  is
bad if any of the input channels is bad.
```

### 1.2.4 Position matching

```
LAS\SET MATCH Tol
```

```
Turn on the checking of position-matching in ACCUMULATE and AVERAGE. Tol
is  the  position tolerance in current units. The default is SET MATCH 2
(arcsec). The tolerance is also used in FIND, MAP and a few  other  com-
mands.
```

```
LAS\SET NOMATCH
```

```
Turn off checking of position-matching in ACCUMULATE and  AVERAGE.  This
does not change the tolerance for the other functions like FIND and MAP.
```

#### 1.2.5 Calibration matching

```
LAS\SET CALIBRATION [Beam_Tol [Gain_Tol]] or OFF

Specify the tolerance on the Beam efficiency (Beam_Tol) and  gain  image
ratio (Gain_Tol), on turn or calibration checking. A value of 0 means no
check. The default values are 0.02 and 0 respectively.

These values are used by commands ACCUMULATE and AVERAGE to verify  that
the calibration is consistant. They are also used by command WRITE which
writes the corresponding information in the output file only if Beam_Tol
is  non  zero. Note that because the beam efficiency is less than 1, you
could use Beam_Tol=1 to suppress  the  calibration  checking  but  still
write the information.
```

# 2 Status of the CLASS releases

## 2.1 Bug fixes

### 2.1.1 Dec08

Following a bug reported by a **CLASS** user, it appeared that the outputs from `AVERAGE` and `ACCUMULATE` in some exotic modes were incorrect. As of the `nov08` release of **GILDAS**, only the following configurations were fully correct. These include the default mode:

- `ALIGN CHANNEL INTERSECT`, `WEIGHT TIME`, `BAD OR` (default)
- `ALIGN CHANNEL INTERSECT`, `WEIGHT SIGMA|TIME`, any `BAD`
- `ALIGN CHAN|VELO|FREQ INTERSECT`, `WEIGHT SIGMA|TIME`, `BAD OR`

Other configurations (in particular the composite mode) may have produced correct output, but with no waranty since it was depending on the input spectra and the level of complexity of the combination. This bug was affecting the default **CLASS** (*i.e.* **CLASS90**), but **CLASS77** had a comparable level of bug for these commands.

This bug was fixed in the `dec08` release of **CLASS90**, by fully cleaning and factorizing the code. **CLASS77** is not fixed according to the policy concerning it: this package is obsolescent and is frozen in order to keep a comparison point with **CLASS90**.

### 2.1.2 Jul09

The `ACCUMULATE` and `AVERAGE` commands perform many implicit features, and perform silently non-trivial computations. When writing the current document, the exhaustive check of all the combinations led to detect two remaining bugs in the internal resampling loop.

These first configurations were incorrect:

- `ALIGN VELO|FREQ COMPOSITE|INTERSECT`, `WEIGHT EQUAL`, any `BAD`, on spectra with different resolutions

In this case a spectrum with a better resolution was overweighted during the resampling process by a factor $reso(resampled)/reso(input)$. This is not what user expects with an `EQUAL` weighting.

These other configurations were also incorrect:

- `ALIGN CHAN|VELO|FREQ COMPOSITE`, any `WEIGHT`, `BAD OR`

With this bug, *no data available in one spectrum* could be synonymous of *bad data in this spectrum*. As a consequence a piece of the output spectrum was wrongly flagged as bad with the `BAD OR` contamination rule. This behaviour was different from the other modes where *no data available in one spectrum* implies to *use data available in the other spectra*.

These two bugs were fixed in the `jul09` release of **CLASS90**.

### 2.2 Test suites

Consecutive to these bugs detection, it appeared that some attempts to fix bugs in the old versions of **CLASS77** and **CLASS90** had broken other parts of the averaging code. Again, this is because of all the different modes to take into account.

A set of procedures are now available as both demos and test suites. They are in particular aimed to checked the influence of any futur code modification:

- `LAS90> @ gag_demo:demo-accu [C|V|F|*] [I|C|*] [E|T|S|*] [A|O|*]`
- `LAS90> @ gag_demo:demo-aver [C|V|F|*] [I|C|*] [E|T|S|*] [A|O|*]`

These procedures invoke respectively the `ACCUMULATE` and `AVERAGE` commands, with Alignment, Range, Weight and Bad arguments. A star '*' will test all the possible values for the considered argument. Default is to check all the values and all the combinations.

These demos run on a set a spectra and try to combine them in different ways. The input spectra include: different intensities, blanked channels, different abscissae definitions (ranges, reference channels, resolutions), different parameters which influence the weighting. On return the demos try to do their best to compute themselves an average spectrum with can be compared to the output from `ACCUMULATE` or `AVERAGE`. In particular, they invoke the `RESAMPLE` command of **CLASS** to perform their own resampling.

## 3 Algorithm

The `AVERAGE` algorithm is based on a single run through the whole index. The spectra are averaged two by two, each output sum becoming an input spectrum when adding the next observation of the index.

This incremental method requires to associate a weight to the output/input sum, in concordance with the number and weights of the spectra already summed. Furthermore, this weight must be an array (one weight per channel), because the command allows to combine different spectra with different range of frequencies (*e.g.* `COMPOSITE` mode), *i.e.* not the same number of spectra may have contributed to each channel in the output sum.

The `ACCUMULATE` algorithm uses the same code but adds only two input spectra ($R$ and $T$).

The algorithm described here apply only on spectra. The average of continuum drifts rely on specific routines which are not considered here.

### 3.1 Setting the weights

The Fortran type `observation` in **CLASS** provides a weight array (`obs%dataw`) for each observation. It has the same dimension of the abscissa and data arrays. One must take care that this array is not part of the **CLASS** data format, *i.e.* it is not saved in the observation when it is written in the output file. As a consequence, before adding any new spectrum, `AVERAGE` and `ACCUMULATE` will fill this array according to the `SET WEIGHT` method:

- `EQUAL`: the weight array is filled with 1.0 values,
- `TIME`: the weight is set to $t \times f_{\rm res}/T_{\rm sys}^2$, where $t$ is the integration time in seconds, $f_{\rm res}$ the frequency resolution in Hertz, and $T_{\rm sys}$ the system temperature in Kelvin. If the integration time is null, an error is raised.
- `SIGMA`: the weight is set to $1/\sigma^2$, where $\sigma$ is the rms noise in Kelvin. If it is null or not set [2] in the observation header, an error is raised.

This value is unique for a single observation, and all the channels have the same weight. Any previous values set in `obs%dataw` are ignored and overwritten. On the other hand, the weight array of the ongoing sum is preserved. This ensures its correct ponderation in front of the input spectrum.

This method implies that all the weight arrays of the spectra of the index are recomputed from scratch before addition. If a sum (returned by a previous call to `AVERAGE` or `ACCUMULATE`) is used again as input, the memory of all spectra it comes from (their number and their different weights along all the channels) will be lost. One should take care that **`AVERAG`'ing the whole index leads into different results than `AVERAG`'ing it by part**! This is similar to first calculate two weighted means but then calculate a new mean from these without taking into account their weights.

## 3.2 Definition of the output spectrum

Let's denote $R$ and $T$ two spectra that one wants to add, and $S$ the output sum. With the `ACCUMULATE` command, $R$ and $T$ are precisely the corresponding buffers exposed to the user in **CLASS**. With the `AVERAGE` command, $R$ is the average spectrum returned by the $(N-1)^{th}$ loop, and $T$ the $N^{th}$ spectrum in index, with its weight array correctly set. We also call $r_R$ (resp. $r_T$) the resolution of $R$ (resp. $T$). This resolution is either in "number of channels"[3], frequency, or velocity per channel depending on the alignment mode (`SET ALIGN CHANNEL|FREQUENCY|VELOCITY` resp.).

From these, the output resolution is taken as the highest of the two (absolute) resolutions in input:

$$r_S = \max(|r_R|, |r_T|) \tag{1}$$

This ensures that we never oversample a spectrum which has a low resolution.

The range of abscissa in output spectrum depends on the combination mode:

- `COMPOSITE` mode:

$$\begin{aligned} x_{\min,S} &= \min(x_{\min,R}, x_{\min,T}) & (2)\\ x_{\max,S} &= \max(x_{\max,R}, x_{\max,T}) & (3) \end{aligned}$$

- `INTERSECT` mode:

$$\begin{aligned} x_{\min,S} &= \max(x_{\min,R}, x_{\min,T}) & (4)\\ x_{\max,S} &= \min(x_{\max,R}, x_{\max,T}) & (5) \end{aligned}$$

Given the output resolution and range, the number of channels $N$ can be deduced:

$$N_S = \text{int}(\frac{x_{\max,S} - x_{\min,S}}{r_S} + 1.5), \tag{6}$$

[2] `SIGMA` is usually set by the command `BASE`.
[3] in this case the resolution is set to *1 channel per channel*

where the 1.5 value adds 1 or 2 extra-channels to avoid the erosion of the spectra at the boundaries.

For the oncoming call of the addition routine, we also define the abscissa at the channel 0 used as reference:

$$x_{\mathrm{val},S} = x_{\mathrm{min},S} - 1. \times r_S, \tag{7}$$

$x_{\mathrm{min},S}$ being the abscissa value of the first channel of the output spectrum. Again, $x$ unit is either in "channels", velocity or frequency depending on the `SET ALIGN` mode.

This definition of the output spectrum is done only once with the `ACCUMULATE` command. On the other hand with the `AVERAGE` command, the above parameters are revised (if required) each time a new spectrum of the index is added. For example, the output range $[x_{\mathrm{min},S}, x_{\mathrm{max},S}]$ and number of channels $N_S$ may enlarge (`SET ALIGN * COMPOSITE`) or reduce (`SET ALIGN * INTERSECT`). Also, the resolution is progressively reduced to the coarsest of all the resolutions in the index (eq. 1).

## 3.3 Aligned spectra

In the simplest case, the spectra have all the same resolution, reference channel and reference value: they are aligned and the channel mode (`SET ALIGN CHANNEL`) can be used. The intensity (temperature) $T(i)$ and weight $w(i)$ of the output sum $S$ are derived from a simple weighted mean of the input spectra:

Averaged channel intensity and weight (aligned spectra):

$$T_S(i) = \frac{w_R(i) \times T_R(i) + w_T(i) \times T_T(i)}{w_R(i) + w_T(i)} \tag{8}$$

$$w_S(i) = w_R(i) + w_T(i) \tag{9}$$

On return, $S$ weight array can vary across the abscissae axis, *e.g.* in `COMPOSITE` mode the overlapping channels typically have a double weight compared to the side channels.

## 3.4 Non-aligned spectra

When aligning the spectra by velocity or frequency (`SET ALIGN VELOCITY|FREQUENCY`), the input channels may not be aligned with the output ones (because of different resolutions or different abscissae at reference channels). A resampling must be performed in these cases. Fig. 1 shows a generic example where the channels have different size and are not aligned.

### 3.4.1 Resampling

Let's consider the input spectrum $R$. We assume that the intensity $T_R$ at channel $i$ is a random variable which follows the normal distribution $N(\mu_R, {\sigma_R}^2)$. Its probability density function is:

$$\mathrm{pdf}_R(x) = \frac{1}{\sigma_R\sqrt{2\pi}} \exp\left(-\frac{(x-\mu_R)^2}{2{\sigma_R}^2}\right) \tag{10}$$

where $\mu_R$ is the mean and $\sigma_R$ the standard deviation. The variance of such a distribution is:

$$\mathrm{var}(T_R(i)) = \sigma_R(i)^2 \tag{11}$$

Choosing a weight $w_R(i)$ of the following form:

$$w_R(i) = \frac{1}{\sigma_R(i)^2} \tag{12}$$

ensure that the weighted mean is the maximum likelihood estimator of the mean, under the assumption of independent (*i.e.* uncorrelated at first order) and normally distributed channel intensities with the same

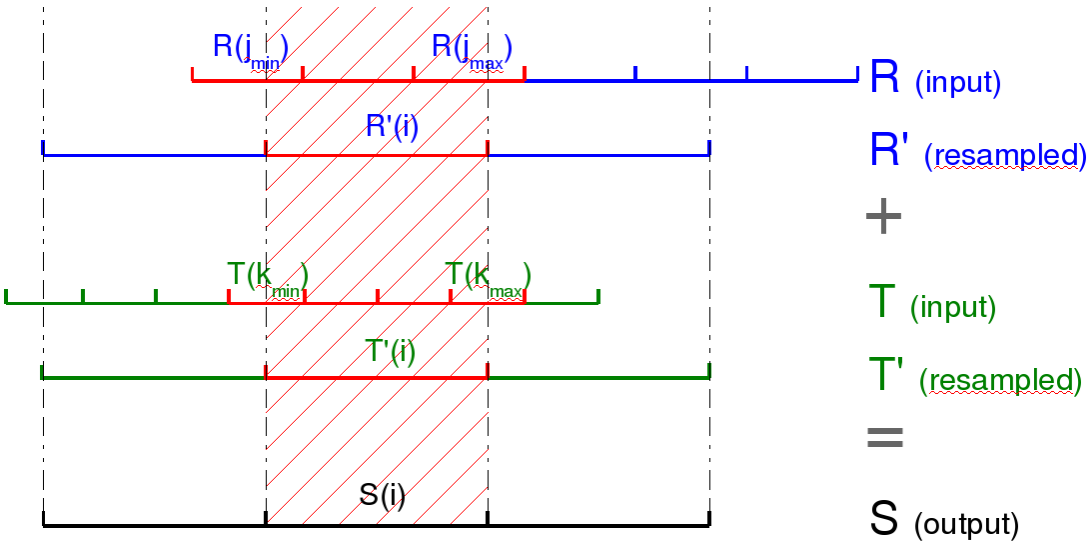



Figure 1: How the input spectra $R$ and $T$ are resampled into the intermediate spectra $R'$ and $T'$ resp. to match the desired output spectrum $S$ (general case). $R$ (resp. $T$) channels $j_{\min}$ to $j_{\max}$ (resp. $k_{\min}$ to $k_{\max}$) have a partial or total contribution to $S(i)$.

mean.

Let's call $R'$ the resampled version of $R$ such as it is aligned with the desired output spectrum $S$. The intensity of $R'$ at channel $i$ can be written (see fig. 1):

Resampled channel intensity (all weights):

$$T_{R'}(i) = \frac{\sum_{j=j_{\min}}^{j_{\max}} f_R(j) \times w_R(j) \times T_R(j)}{\sum_{j=j_{\min}}^{j_{\max}} f_R(j) \times w_R(j)} \tag{13}$$

where:

- $j_{\min}$ and $j_{\max}$ are the channel range of input spectrum $R$ which are overlapped at least partially by the output channel number $i$,
- $f(j)$ is the used fraction of the different input channels. It quantifies their contribution in the selected range:
  $f(j) = 0$ if the input channel is out of the range,
  $f(j) = 1$ if the output channel fully overlaps the input channel,
  $0 < f(j) < 1$ otherwise (proportionally to the overlap).

We can also define the normalization factor $\beta$ and the weight $\alpha(j)$:

$$\beta = \sum_{j=j_{\min}}^{j_{\max}} f_R(j) \times w_R(j) \tag{14}$$

$$\alpha_R(j) = \frac{f_R(j) \times w_R(j)}{\beta} \tag{15}$$

The normalization factor $\beta$ ensures that the total integrated intensity is preserved during the resampling

process. With these, $T_{R'}(i)$ can be written:

$$T_{R'}(i) = \sum_{j=j_{\min}}^{j_{\max}} \alpha_R(j) \times T_R(j) \tag{16}$$

### 3.4.2 Channel weight of a resampled spectrum

With the assumption of independent channel intensities, the weight $w_{R'}(i)$ of this resampled channel can be derived from its variance:

$$\begin{aligned} \mathrm{var}(T_{R'}(i)) &= \mathrm{var}\left(\sum_{j=j_{\min}}^{j_{\max}} \alpha_R(j) \times T_R(j)\right) && (17)\\ &= \sum_{j=j_{\min}}^{j_{\max}} \alpha_R(j)^2 \times \mathrm{var}\left(T_R(j)\right) && (18)\\ &= \frac{1}{\beta^2} \sum_{j=j_{\min}}^{j_{\max}} f_R(j)^2\, w_R(j) && (19) \end{aligned}$$

where eq. 18 is derived from the general variance property:

$$\mathrm{var}(aX + bY) = a^2\mathrm{var}(X) + b^2\mathrm{var}(Y) \tag{20}$$

where $a$ and $b$ are numerical constants and when $X$ and $Y$ are independent random variables.

Finally from eqs. 12 and 19, the resampled channel weight is[4]:

Resampled channel weight (weights `TIME` and SIGMA):

$$w_{R'}(i) = \frac{\left(\sum_{j=j_{\min}}^{j_{\max}} f_R(j) w_R(j)\right)^2}{\sum_{j=j_{\min}}^{j_{\max}} f_R(j)^2\, w_R(j)} \tag{21}$$

This relation has a non-intuitive effect on the resampled spectrum: its weight is, in the general case, different from the original one. Let's assume that the spectrum $R$ is resampled onto a spectrum $R'$ with the same resolution but with a shifted value $x_{\mathrm{val}}$ at reference channel (fig. 2). In such a case, the channel weight $w_{R'}$ we can deduce, and its associated $\sigma_{R'}$, are:

1. $x_{\mathrm{val}} \to x_{\mathrm{val}} + 0.00$: $w_{R'} = w_R$, $\sigma_{R'} = \sigma_R$
2. $x_{\mathrm{val}} \to x_{\mathrm{val}} + 0.25$: $w_{R'} = 1.6 \times w_R$, $\sigma_{R'} = \sigma_R/\sqrt{1.6}$
3. $x_{\mathrm{val}} \to x_{\mathrm{val}} + 0.50$: $w_{R'} = 2 \times w_R$, $\sigma_{R'} = \sigma_R/\sqrt{2}$

assuming all the $R$ channels have the same weight (*i.e.* same $\sigma_R(j)$). The extra factor $w_{R'}/w_R$ affects either the `SIGMA` and the `TIME` weightings. These examples show that, depending on the desired resampling, the weight of the recomputed spectrum may be different. In the two latter cases, a correlation has been introduced between contiguous channels. From the physical point of view, this can be explained from the fact that one resampled channel contains more information than one original channel, *e.g.* in case number 3 it has an equal contribution of the two original ones: the noise is reduced by a factor $\sqrt{2}$.

This should be kept in mind when averaging *e.g.* two $R$ and $T$ spectra with same resolution but shifted X-axis. If a `SIGMA` weighting is invoked, the average won't be found at the mean distance of the two spectra **even if their $\sigma$ are equal!**

[4]This is verified by numerical simulations on spectra with known (simulated) rms noise.

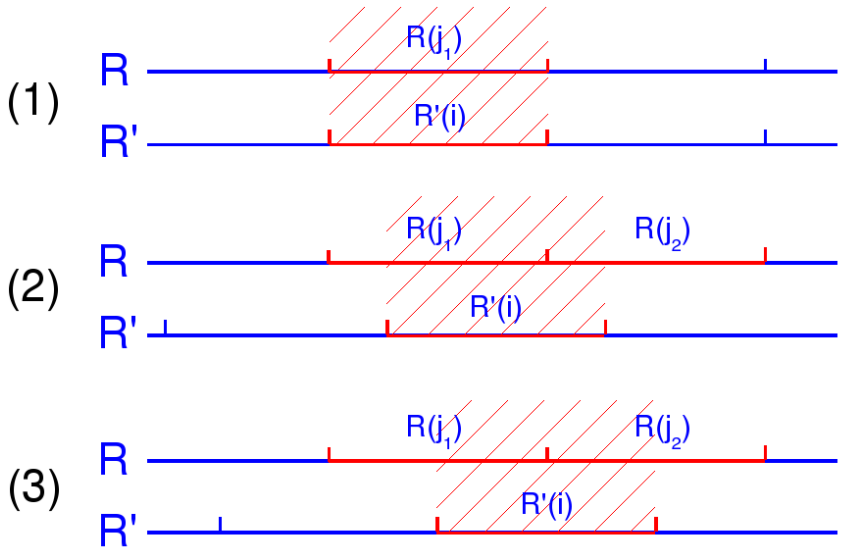



Figure 2: Resampling of $R$ spectrum into $R'$, when the resolution is preserved but the reference channel is shifted by 0.0 (1), 0.25 (2) and 0.5 (3) channel.

One should also take care that eq. 21 assumes **uncorrelated** input channels. Resampling a spectrum which was already resampled (*e.g.* $R' \rightarrow R''$) introduces a correlation between more contiguous channels. In particular this equation should not be used to compute the associated weights.

The weight at eq. 21 apply to the `TIME` and SIGMA weighting, where the computations above have a physical meaning and one can expect consistant values for integration time, channel width and $\sigma$. For the `EQUAL` weighting, user expects the two input spectra to have the same weight whatever their abscissa axis definition: this means that a re-`EQUAL`ization of the channels must come *after* the resampling. The *ad hoc* weighting is in this case:

Resampled channel weight (weight `EQUAL`):

$$w_{R'}(i) = \frac{\sum_{j=j_{\min}}^{j_{\max}} f_R(j) w_R(j)}{\sum_{j=j_{\min}}^{j_{\max}} f_R(j)} \tag{22}$$

where $w_R(j)$ is either 1.0 (see section 3.1) for a new input spectrum, or any (possibly not constant) value for the reentrant sum. This preserves a correct ponderation of the reentrant sum in front of a new input spectrum.

### 3.4.3 Output spectrum

The output intensity $T_S$ and weight $w_S$ at channel $i$ are set to:

Averaged channel intensity and weight (non-aligned spectra):

$$T_S(i) = \frac{w_{R'}(i) \times T_{R'}(i) + w_{T'}(i) \times T_{T'}(i)}{w_{R'}(i) + w_{T'}(i)} \tag{23}$$

$$w_S(i) = w_{R'}(i) + w_{T'}(i) \tag{24}$$

where the $R'$ (resp. $T'$) index refers to the resampled version of input spectrum $R$ (resp. $T$) computed from eqs. 13 and 21/22. The normalization in eq. 23 ensures that the integrated intensity is also preserved

over the whole spectrum when averaging these two spectra. These equations are the same as eqs. 8/9 (for aligned spectra) except that they operate on their resampled versions.

## 3.5 Data not available / bad channels

Equations 23 and 24 describe the general case when all the input channels which contribute to the output sum are available. Nevertheless there may be some situations were data is missing. In such cases, the general rule is to ignore the channels with no data (*i.e.* zero-weight them in eqs. 8/23), and to use data in the other spectra when it is available:

- In the `COMPOSITE` mode, if no data is available in an input spectrum (*i.e.* out of its boundaries, in a frequency range covered by other input spectra), its corresponding resampled channels are zero-weighted.
- In the `COMPOSITE` mode, if no data is available in **all** the input spectra (*e.g.* there is a gap covered by none of these), the corresponding channels in the output spectrum are flagged as bad.
- In all modes, any bad channel in an input spectrum is zero-weighted. Additionally, if it contributes to the output channel ($f \neq 0$) but bad channels should propagate (`SET BAD OR`), it contaminates the current output channel which is also set as bad.

## 3.6 Controls

There are several controls which ensures that user adds consistant spectra. The first ones raise errors which stop the averaging process:

- The position compatibility is checked according to the position matching rule (`SET [NO]MATCH`, or option `/NOMATCH` for command `AVERAGE`). Default is to match the positions with a $2''$ tolerance.
- The calibration compatibility is checked according to the beam efficiency and gain image ratio tolerances set by command `SET CALIBRATION`. Default is to check the beam efficiencies with a 0.02 tolerance, but not the gain image ratios.
- In the intersection mode (`SET ALIGN * INTERSECT`), an error is raised if the input abscissae ranges do not intersect.

There is also a serie of conditions under which user is warned about possible inconstent mix of spectra:

- for channel alignment of the input spectra (`SET ALIGN CHANNEL`), a difference in the reference channels, in the frequency resolutions, or in the sky frequencies raises a warning,
- for frequency alignment of the input spectra (`SET ALIGN FREQUENCY`), a difference in the velocity offsets raises a warning (this indicates a different matching between the frequency and velocity axes),
- for velocity alignment of the input spectra (`SET ALIGN VELOCITY`), a difference in the rest frequencies raises a warning (for the same reason).

## 3.7 Output parameters

The main sections of the header contain some fundamental parameters which are unique for all the spectrum. This is a problem for the output spectrum from `ACCUMULATE` or `AVERAGE` which may contain different part coming from a various number of spectra (e.g. in `COMPOSITE` mode). Actually, only the `INTERSECT` mode with `SET BAD OR` ensures that the same number of input spectra have been used to compute the non-bad channels of the output spectrum.

Nevertheless, typical values have to be given to these parameters. First of all, when adding a new spectrum *obs* with the previous sum $S_{\rm in}$, the header of the output sum $S_{\rm out}$ is completely copied from *obs* header. Then specific values of the different sections of the header are adjusted.

### 3.7.1 `GENERAL` section

The `GENERAL` section is then revised:

- the integration time $t_{S_{\rm out}}$ is redefined with the sum of the two input integration times:

$$t_{S_{\rm out}} = t_{S_{\rm in}} + t_{\rm obs} \tag{25}$$

- the system temperature $T_{{\rm sys},S_{\rm out}}$ is derived from the sum of $t \times |r|/T^2_{\rm sys}$ (where $r$ is here the frequency resolution):

$$\frac{t_{S_{\rm out}} \times |r_{S_{\rm out}}|}{T^2_{{\rm sys},S_{\rm out}}} = \frac{t_{S_{\rm in}} \times |r_{S_{\rm in}}|}{T^2_{{\rm sys},S_{\rm in}}} + \frac{t_{\rm obs} \times |r_{\rm obs}|}{T^2_{\rm sys,obs}} \tag{26}$$

- the output observation number, version number, and scan number are arbitrarily reset to the $S_{\rm in}$ ones, *i.e.* coming from the first observation in the index.

### 3.7.2 `SPECTROSCOPIC` section

The `SPECTROSCOPIC` section is revised as follows:

- from the first copy of all the input header of *obs* into $S_{\rm out}$, the spectroscopic section inherits the *obs* values, in particular the rest and image frequencies.
- the number of channels is set to the appropriate value (according to the `COMPOSITE` or `INTERSECT` mode) computed at section 3.2.

Then, depending on the alignment rule, another set of values can be set:

- in case of channel alignment (`SET ALIGN CHANNEL`), no more values of this section are redefined.
- in case of frequency alignment (`SET ALIGN FREQUENCY`), the frequency and associated velocity parameters are set:
  - the output frequency resolution $f_{{\rm res},S_{\rm out}}$ is set to the value computed at Eq. 1,
  - the output frequency offset $f_{{\rm off},S_{\rm out}}$ is set to $f_{{\rm off},S_{\rm in}}$,
  - the reference channel associated to $f_{\rm off}$ is set to $(f_{{\rm off},S_{\rm out}} - x_{{\rm val},S_{\rm out}})/f_{{\rm res},S_{\rm out}}$ where $x_{\rm val}$ is the frequency value at channel 0[5], computed at Eq. 7,
  - then the velocity resolution $v_{{\rm res},S_{\rm out}}$ is set to $-c \times f_{{\rm res},S_{\rm out}}/f_{{\rm rest},S_{\rm out}}$ where $c$ is the speed of light and $f_{\rm rest}$ the rest frequency,
  - and the velocity offset $v_{{\rm off},S_{\rm out}}$ is set to $v_{{\rm off},S_{\rm in}}$
- in case of velocity alignment `SET ALIGN VELOCITY`, the velocity and associated frequency parameters are set to:
  - the output velocity resolution $v_{{\rm res},S_{\rm out}}$ is set to the value computed at Eq. 1,
  - the output velocity offset $v_{{\rm off},S_{\rm out}}$ [6] is set to $v_{{\rm off},S_{\rm in}}$,
  - the reference channel is set to $(v_{{\rm off},S_{\rm out}} - x_{{\rm val},S_{\rm out}})/v_{{\rm res},S_{\rm out}}$ where $x_{\rm val}$ is the velocity value at channel 0, computed at Eq. 7,
  - then the frequency resolution $f_{{\rm res},S_{\rm out}}$ is set to $-f_{{\rm rest},S_{\rm out}} \times v_{{\rm res},S_{\rm out}}/c$,
  - and the frequency offset $f_{{\rm off},S_{\rm out}}$ is set to $f_{{\rm off},S_{\rm in}}$

[5] $f(i_{\rm chan}) = f_{\rm res} \times (i_{\rm chan} - r_{\rm chan}) + f_{\rm off}$ which, at $i_{\rm chan} = 0$, can be written $x_{\rm val} = f_{\rm res} \times (0 - r_{\rm chan}) + f_{\rm off}$

[6] even if $v_{\rm off}$ is always null in **CLASS**, the algorithm is as generic as possible.

#### 3.7.3 `BASE` section

In the `BASE` section, the $\sigma$ value is reset to 0.0 because, in most of the cases, it makes non-sense to provide a single value for channels which may be the sum of different number of inputs.

#### 3.7.4 `HISTORY` section

This section is updated such as it contains the history of the scan numbers of the input observations. This is available as a list of sequences of consecutive scan numbers.

### 3.8 Concatenation

The `AVERAGE` command can be used to concatenate spectra of different frequency ranges. This can be done by enabling the `COMPOSITE` mode (`SET ALIGN VELO|FREQ COMPOSITE`).

All the rules presented in the different sections above apply without exception:

- the output abscissa is computed as exposed in section 3.2. In particular, the output resolution is set to the coarsest resolution of all the spectra in the index (eq. 1), and the output range is the minimal which can contain all the input abscissae ranges,
- the resampling (and averaging in overlapping parts) rules presented at section 3.4 apply,
- the returned header is also set according to section 3.7. In particular, the output integration time is the sum of all the times of input spectra (eq. 25)[7].

## 4 Discussion

As raised in section 3.1, the **CLASS** data format does not provide a weight array, even if one is available in the Fortran type `observation` which is used to load in memory the observations read from file. The `observation` type is used only when processing the data *e.g.* in the current context of `AVERAGE` and `ACCUMULATE`.

For consistency reasons over all the commands in **CLASS** which can make use of the weight array, its content is assumed to be inexistant or unset when entering the command: the weight array is recomputed each time it is needed. This implies a unique value for all the channels, *e.g.* the $\sigma$ value computed by `BASE`.

The discussions in this document raise the idea to save the weight array in the **CLASS** data format. Indeed, we have seen that the single value which is recomputed from scratch can not reflect the real weight of each channel, *e.g.* when the spectrum is the `AVERAGE` of several others in `COMPOSITE` mode. But this has two major caveats:

- if it is introduced in the **CLASS** data format, the weight should properly be taken into account everywhere in **CLASS**, in each command which makes use of it, including `AVERAGE`, `ACCUMULATE`, and `BASE`.
- this implies an increased weight on disk for each spectrum: the amount of data written (data array + weight array) will typically double. This point has not been quantified in particular their relative weight compared to the observation header.

  For small amount of data (*e.g.* with long integration times, line surveys,...), this would not be a problem. But when dealing with large quantity of data *e.g.* multibeam OTF observations, the increase would become dramatic. We may need in this case to have the choice to save or not the weight arrays.

[7] User is free is redefine at its convenience the header sections by using *e.g.* `SET VAR SPECTRO WRITE`

For the time being, saving (always or not) the weight array in the **CLASS** data format remains an open question.

# 5 Conclusion

In this document we fully described the current status of the `ACCUMULATE` and `AVERAGE` commands of **CLASS**, applied on spectra. They are very often used, and for different purposes, *e.g.* averaging a set of spectra or concatenating them. User must be warned about several traps that he may encounter:

- averaging a set of spectra with successive calls to `AVERAGE` may lead into different results than averaging the whole set with a single call to the command,
- averaging non-aligned spectra must invoke at one point a resampling. User can do it himself or rely on the internal resampling loops of the commands. The resampling has non-trivial effect of the weights.

As a solution to the first problem, the question of saving a weight array (one weight per channel) in the **CLASS** data format has been raised. This would have consequences in particular in term of disk storage.