Spectral data doppler corrections for the 40 m radiotelescope

A. Díaz-Pulido, P. de Vicente

Informe Técnico IT-CDT/OAN 2014-10

Revision history

Version	Date	Author	Updates
1.0	13-04-2014	P. de Vicente A. Díaz-Pulido	First draft data analysis

Contents

1 Introduction

We have discovered that a full Doppler effect correction was not properly applied in the spectral data from the 40m radiotelescope. In this short report we explain the problem and how it was corrected.

2 The Doppler effect in the telescope

Spectral data require Doppler effect corrections to avoid smearing the spectra when integrating along time. The cause is simple, the observer moves relative to the source and as a consequence the observed frequency shifts towards the red or the blue during the observation time. This shift depends on the frequency of observation. The movement of the observer relative to the source has several contributions: topocentric (which ammounts for the rotation of the Earth), geocentric (rotation around the sun), and drift with respect to the Local Standard of Rest (heliocentric). The last two are the main contributors to the final velocity of the observer.

2.1 The Local Standard of Rest

The Local Standard of Rest (LSR) is a point in space that is moving on a circular orbit around the center of the galaxy at the Sun's galactocentric distance. Its velocity has been estimated to be $\simeq 220$ km/s. It can be determined by averaging the radial velocities and the proper motions of the stars in the vicinity of the Sun. When the Sun is compared to the nearby stars it is seen that, on average, stars move towards the Sun in one direction and away from the Sun in the opposite direction. This is due to the movement of the Sun with respect to the LSR with a velocity of $\simeq$ 20 km/s towards the solar apex (18:03:50.2, 30:00:16.8). The Sun moves faster than the LSR in its rotation around the center of the galaxy, it slightly moves towards the galactic center and northwards of the galactic plane.

When doing spectral observations the radial velocity of the sources is referred to the LSR. In order to compute the radial velocity of the source with respect to the antenna, it is necessary to compute the radial velocity of the observer with respect to the LSR and substract it from the velocity of the source referred to the LSR.

The velocity of the observer is computed by doing a scalar product between the velocity of the observer in the direction of the solar apex and an unitary vector in the direction of the source. This is acomplished in three steps: computing the velocity of the sun, the velocity of the Earth with respect to the sun and the velocity of the observatory with respect to the center of the Earth.

2.2 First order correction

Let v_{sun} be the velocity of the sun towards the solar apex and $\mathbf{u_{sa}}$ and $\mathbf{u_s}$ unitary vectors towards the direction of the solar apex and towards the source respectively, then the radial velocity of the sun (v_{rs}) towards the source will be:

$$v_{rs} = v_{sun}(\mathbf{u_{sa}} \cdot \mathbf{u_s})$$

Let α_{sa} and δ_{sa} be the right ascention and declination of the solar apex respectively and α and δ the right ascention and declination of the source, then we have that:

$$v_{rs} = v_{sun}(\cos\delta_{sa}\cos\alpha_{sa}, \cos\delta_{sa}\sin\alpha_{sa}, \sin\delta_{sa}) \cdot (\cos\delta\cos\alpha, \cos\delta\sin\alpha, \sin\delta)$$

$$\begin{aligned} v_{rs} &= v_{sun}\,(\cos\delta_{sa}\cos\alpha_{sa}\ \cos\delta\cos\alpha + \cos\delta_{sa}\sin\alpha_{sa}\ \cos\delta\sin\alpha + \sin\delta_{sa}\ \sin\delta) \\ &= v_{sun}\ (\cos\delta_{sa}\cos\delta(\cos\alpha_{sa}\cos\alpha + \sin\alpha_{sa}\sin\alpha) + \sin\delta_{sa}\ \sin\delta) \end{aligned}$$

and the radial heliocentric velocity:

$$v_h = v_{LSR} - v_{rs}$$

v_{rs} varies between -20 and 20 km/s depending on the source coordinates. It has no dependence on the time of the observation.

Fig. 1 depicts graphically the relation between the heliocentric velocity and the LSR velocity of the source.

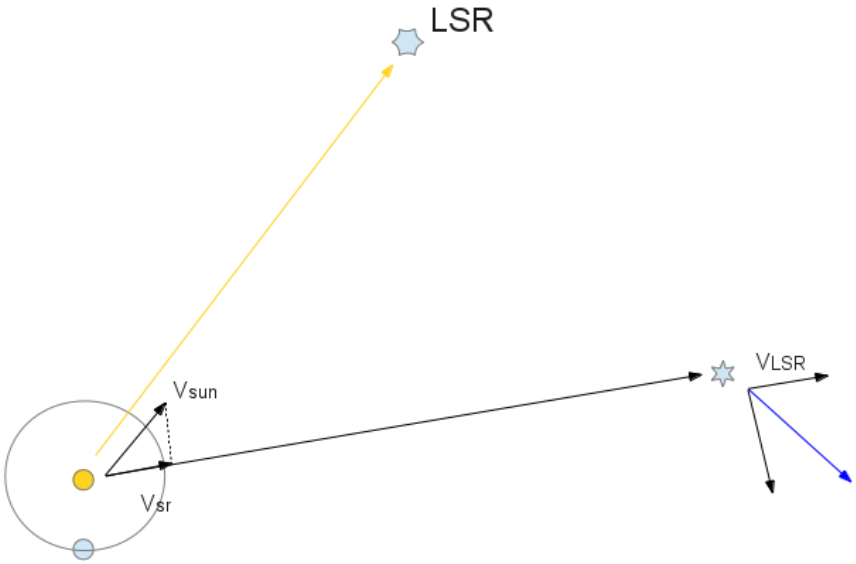

Figure 1: *Scheme which relates the velocity of the source with respect to the LSR and the heliocentric velocity.*

The geocentric velocity is computed in the same way as before. The velocity of the Earth ($\mathbf{v_e}$) is usually given in cartesian coordinates in an equatorial heliocentric system and the radial velocity towards the source (v_{re}) is obtained from the scalar product:

$$\begin{aligned} v_{re} &= \mathbf{v_e} \cdot \mathbf{u_s} \\ &= (v_x, v_y, v_z) \cdot (\cos\delta\cos\alpha, \cos\delta\sin\alpha, \sin\delta) \\ &= v_x\cos\delta\cos\alpha + v_y\cos\delta\sin\alpha + v_z\sin\delta \end{aligned}$$

and then the geocentric velocity v_g would be:

$$v_g = v_{LSR} - v_{rs} - v_{re} = v_h - v_{re}$$

The velocity of the Earth is approximately 30 km/s and its projection on the direction of the source depends, obviously on the coordinates of the source and on the time of the year. The radial velocity can then vary between -30 to 30 km/s with respect to the LSR.

Finally the topocentric velocity is computed from the rotation of the Earth, the location of the observatory and the time of the day. The same computation as before is applied. Let $\mathbf{v_o}$ be the velocity of the observatory in a geocentric equatorial system.

$$\begin{aligned} v_{ro} &= \mathbf{v_o} \cdot \mathbf{u_s} \\ &= (v_{xo}, v_{yo}, v_{zo}) \cdot (\cos\delta\cos\alpha, \cos\delta\sin\alpha, \sin\delta) \\ &= v_{xo}\cos\delta\cos\alpha + v_{yo}\cos\delta\sin\alpha + v_{zo}\sin\delta \end{aligned}$$

and the topocentric velocity, v_t would be:

$$v_t = v_{LSR} - v_{rs} - v_{re} - v_{ro} = v_h - v_{re} - v_{ro}$$

The velocity of the observer with respect to the center of the Earth depends on the latitude, the direction of the source and the time of the day. It may vary 0.6 km/s at most in 24 hours.

Therefore the radial velocity of the source referred to the observer may differ up to 50 km/s from the LSR one and it varies mainly with the observation date and strongly depends on its coordinates. The contribution due to the rotation of the Earth may ammount a 2% at most.

In the 40 m radiotelescope the Doppler effect is computed every second for the observing frequency and a correction is continuously applied to the local oscillator when performing single dish observations. Let the f_{obs} be the observing frequency, the local oscillator frequency will be computed from the IF frequency:

$$f_{OL} = f_{obs} - f_{IF}$$

The correction to be applied to the local oscillator is:

$$\delta f_{OL} = -f_{obs}\,\frac{v_t}{c}$$

and the final local oscillator frequency will be $f_{OL} + \delta f_{OL}$.

The Earth rotation, at 45 GHz , ammounts at most a frequency change during a whole day of 90 KHz approximately. At Yebes, at the time of the report we use as spectral backend an FFTS with 500 MHz bandwidth and 16384 channels. Each channel is 30 KHz wide, therefore the frequency change in a day due to the rotation of the Earth may go up to 3 channels approximately.

The corrections explained above are basic and very well known in all radiotelescopes of the world and have been applied since many years when doing spectral observations. However a more subtle correction is required, which is specially important when the observed band is wide.

2.3 Second order correction

When a spectral interval is observed, not only the whole interval shifts due to the relative velocity between the observer and the source, but it also gets compressed or expanded. That means that the shift, usually computed for the center of the interval is not correct for other frequencies in the observing band, and this difference is highest at both ends of the interval. If one observes a line in the upper end of an interval and later in the lower end of another frequency interval, when averaged, the spectra will not align properly. This effect is the one that we discovered as uncorrected and it is briefly described below.

Let the bandwith interval be Δf and f_0 the central frequency. The error made when correcting the Doppler effect in the lower end can be computed by substracting the correction required at the lower end from the correction at the center: According to the explanation above, the correction at the center of the observing band is:

$$\delta f_0 = -f_0 \frac{v_t}{c}$$

The correction at the lower end of the observing interval should be:

$$\delta f_{-1} = -(f_0 - \frac{\Delta f}{2})\frac{v_t}{c}$$

and at the upper end:

$$\delta f_1 = -(f_0 + \frac{\Delta f}{2})\frac{v_t}{c}$$

The frequency error at the lower end will be:

$$\begin{aligned} \delta f_{-1} - \delta f_0 &= -(f_0 - \frac{\Delta f}{2})\frac{v_t}{c} + f_0\frac{v_t}{c} \\ &= \frac{\Delta f}{2}\frac{v_t}{c} \end{aligned}$$

and at the upper end:

$$\begin{aligned} \delta f_1 - \delta f_0 &= -(f_0 + \frac{\Delta f}{2})\frac{v_t}{c} + f_0\frac{v_t}{c} \quad (1) \\ &= -\frac{\Delta f}{2}\frac{v_t}{c} \quad (2) \end{aligned}$$

That is, at both ends the errors have different signs. Therefore the Doppler effect causes a frequency shift and a frequency contraction or expansion. Fig 2 displays the behaviour graphically.

If the ratio v_t/c is known for the source and date and time of the observation, a correction can be applied to the observed spectrum. The corrections should have the opposite sign to what is depicted in Fig. 2. That is, for example a channel whose frequency is above the central frequency should be considered to hold data from a frequency which is the rest frequency plus the correction in 2. Obviously the correction gets more important with increasing observing bandwidths and should also be taken into account with small bandwidths but high frequency resolution.

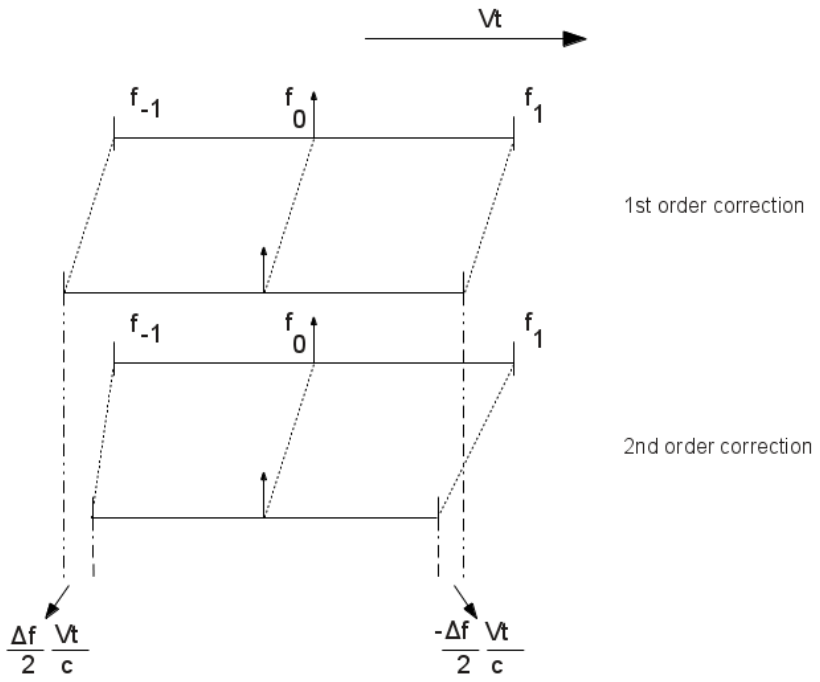

Figure 2: *Scheme which explains graphically the Doppler effect in a frequency interval: the interval gets shifted and contracted. The plot above corresponds to what is supposed to happen when a first order correction is taken into account. Δf is the instantaneous observing bandwidth. The plot below corresponds to the real behaviour.*

3 Effects on spectra when the second Doppler correction is not applied

Fig. 3 illustrates the problem described in the previous section when the second order correction was not applied. The figure depicts the maser emission from line ^{28}SiO v=1 J=1-0, towards IK Tau and TX CAM. Two spectra per source are depicted. The spectra were obtained with different central frequencies which were 400 MHz apart.

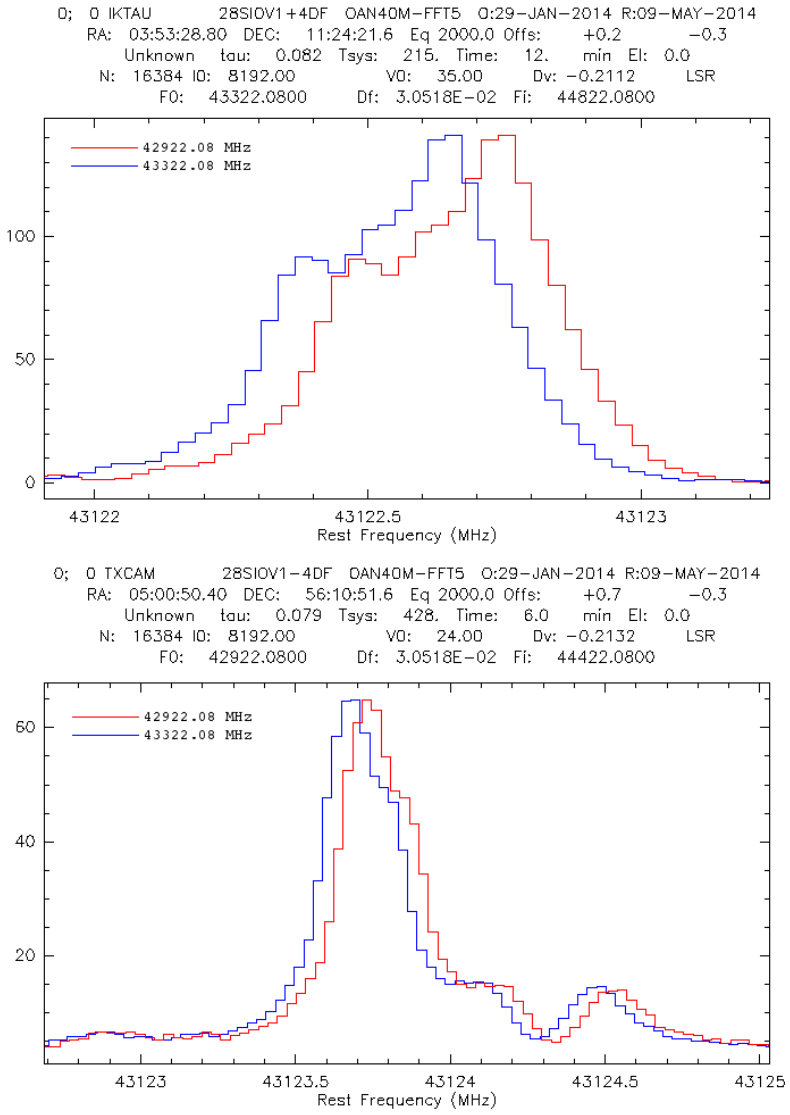

Figure 3: *Spectra obtained after shifting the central frequency $\simeq$ 400 MHz without taking into account second order Doppler corrections.*

According to Fig. 3, there is a mismatch of more than 3 channels towards IK Tau, and more than 2 towards TX Cam. The discrepancy in channels, as explained in the preceding section, depends on the topocentric velocity (v_t). An average of both spectra would yield a smeared spectrum.

4 CLASS and Doppler effect

Recent CLASS versions can apply a correction to each channel, provided $-v_t/c$ is supplied (v_t is the topocentric velocity). This ratio is stored in the header of each spectrum and later used to correct the frequency in every channel when averaging spectra or plotting them. This requires that the control system of the telescope computes and provides this number at the beginning of every ON-OFF scan.

When this value is not provided, CLASS automatically assigns it a value of -1 and tries to compute it from the date and time of the scan and from a list of known radio observatories in which Yebes is not included at the time of this report. The name of the observatory is obtained from TELESCOPE variable. This strategy loads the CPU and slows the analysis process because it requires a computation every time the spectrum is read. Therefore the optimum solution is to store the topocentric velocity in light speed units every time the pipeline writes the data. In any case the position of the 40m radiotelescope will be included in versions above May 2014 and the software will look for the string "OAN-40M" in the TELESCOPE field. The list of known observatories can be found in the source code in file: `astro/lib/observatory.f90`

Since March 2014 it is also possible to use command `set observatory Name long lat alt` to override the observatory read from the TELESCOPE variable name. This allows to use any observatory independently if the source code contains or not its position.

Fig. 4 shows spectra obtained after March 2014, date at which we implemented the correction described above. According to the plots, channels do not align perfectly when trying to match two or more spectra. This lack of match is never above 1 channel width. We believe this is an effect due to the channel frequency resolution, in this case 30.5 KHz. CLASS allows the average using the following command and options:

```
average /resample /nocheck
```

We have observed that it is possible to avoid channel mismatch between channels if the observing frequency (f_0) is shifted by a given quantity which is a (non-integer) multiple of the number of channels. If we use the FFTS with a total bandwidth of 500 MHz and 16384 channels, then the new central frequency f' should be:

$$f' = f_0 + \frac{500}{16384}\, ch(1 + \frac{v_t}{c}) \tag{3}$$

where v_t is the topocentric velocity, c the speed of light and ch the number of channels we want to shift.

Figs. 5 and 6 shows a portion of the spectrum (where the maser emission is maximum) after having shifted the central frequency several times using equation 3. Channels match perfectly.

In order to understand what equation 3 means we have prepared Fig. 7 in which we show two arbitrary channels which approximately have the same frequency assigned but have been

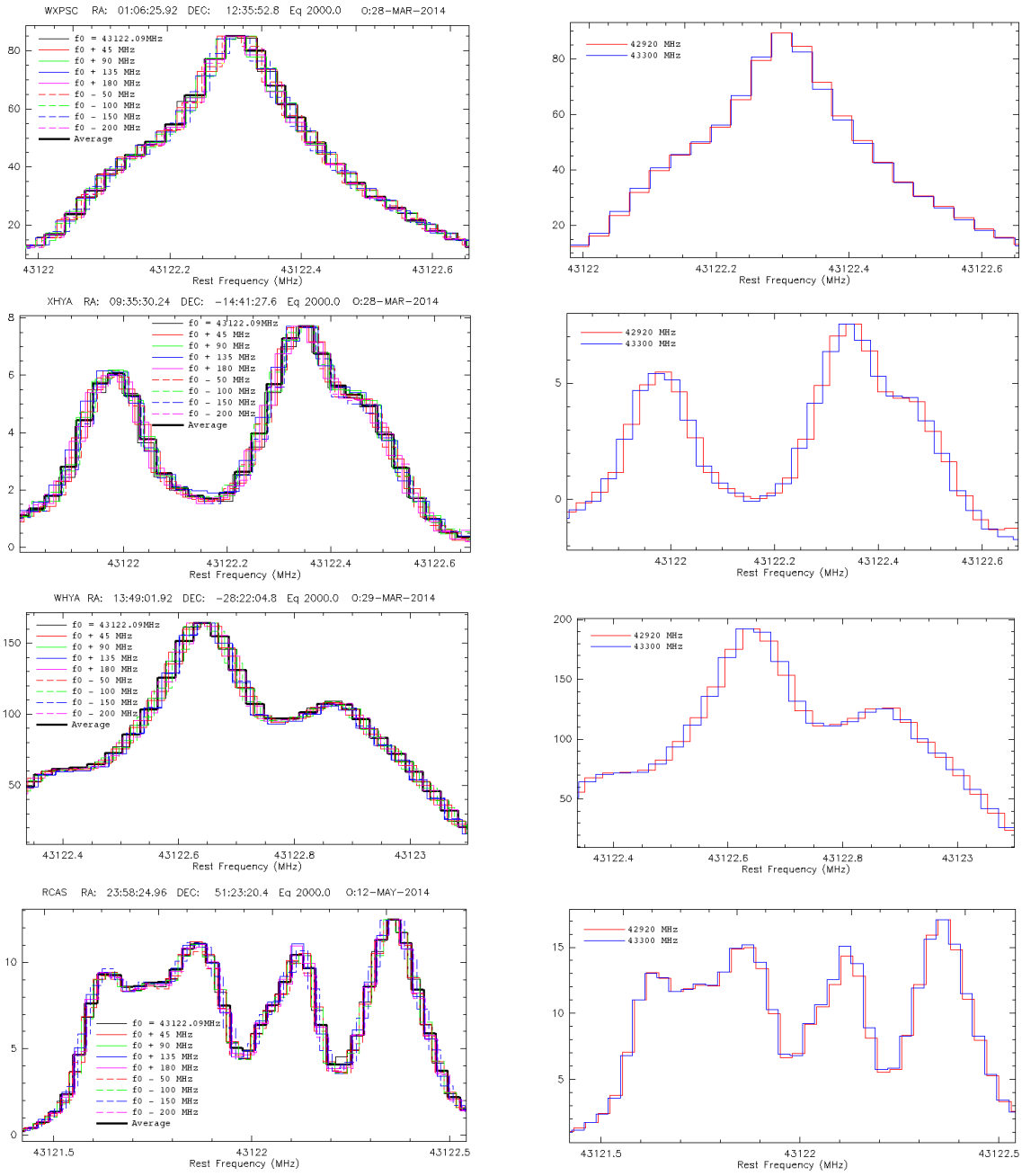

Figure 4: ^{28}*SiO v=1 (1-0) maser emission from sources: WX-Psc, X-Hya, W-Hya & R-Cas. Left panels: spectra after shifting the central frequency from -200 to +180 MHz with respect to frequency of the maser emission in steps of 45 MHz and 50 MHz. Right panels: Two spectra obtained with the maximum frequency shift: 380 MHz.*

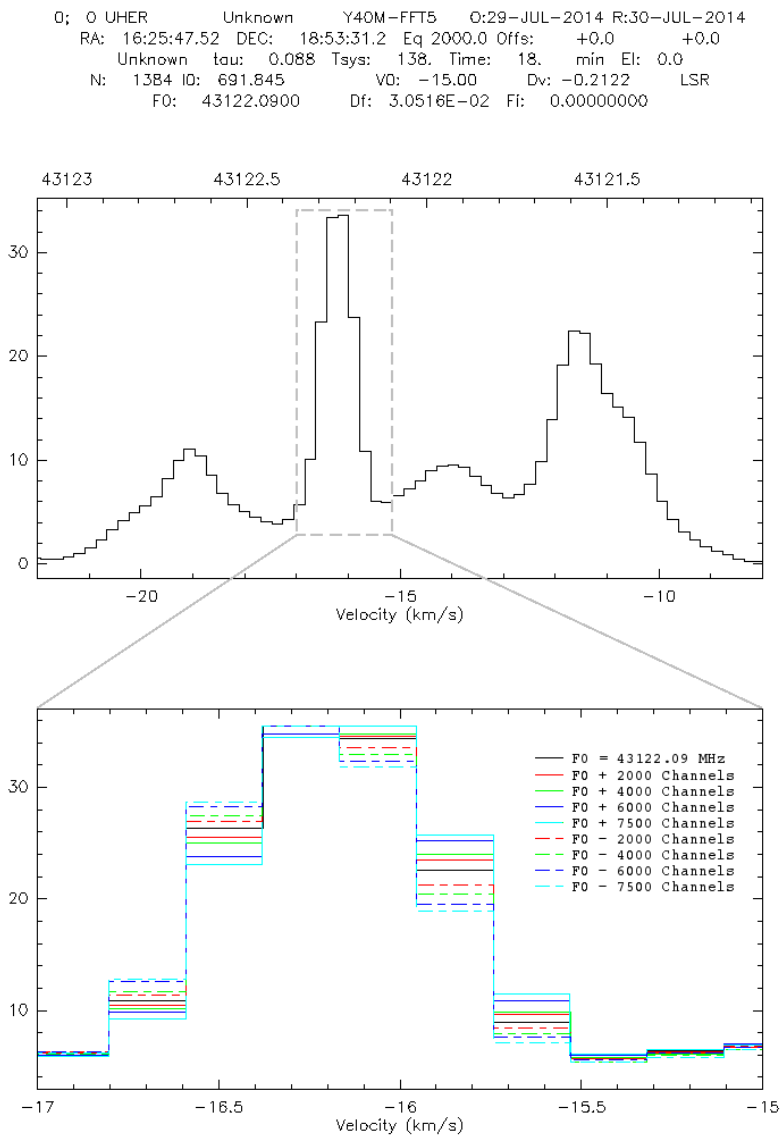

Figure 5: *^{28}SiO v=1 (1-0) maser emission spectrum towards source U-Her using different observing frequencies (each represented with a color code) and using equation 3.*

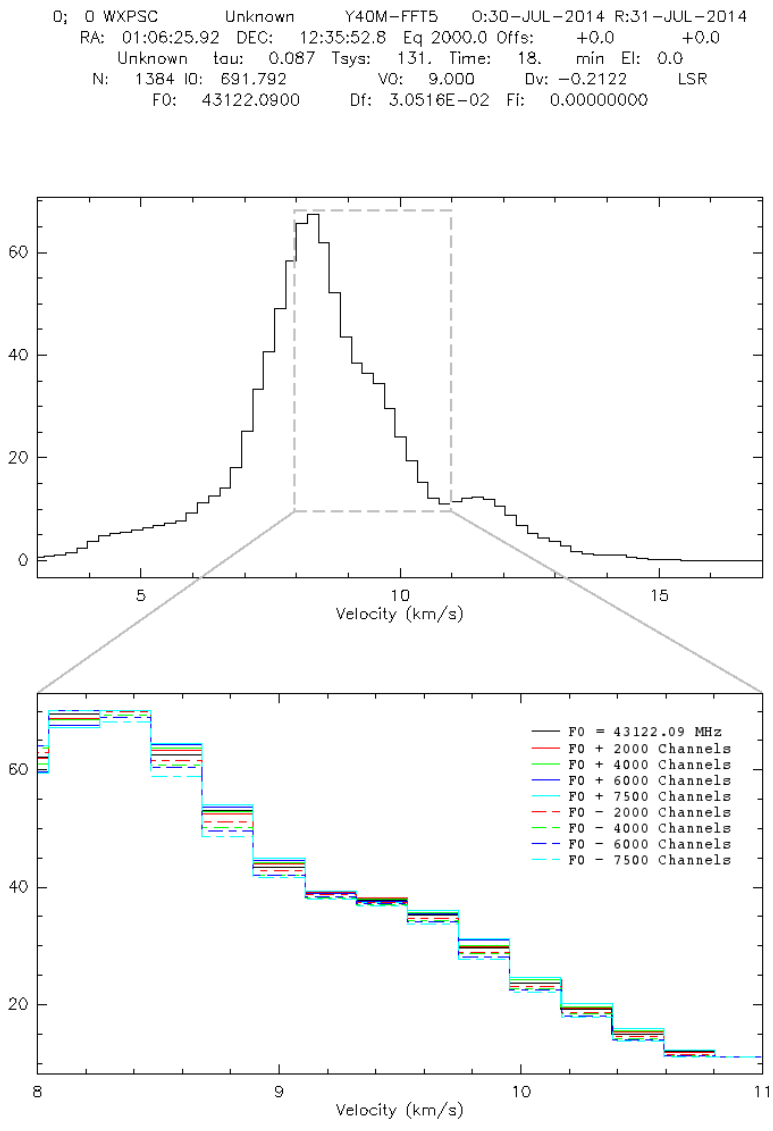

Figure 6: *^{28}SiO v=1 (1-0) maser emission spectrum towards WX-PSC using different observing frequencies (each represented with a color code) and using equation 3.*

observed with different reference frequencies. For clarity, and to stress that channels are different, we have arbitrarily assumed channel numbers to be 200 and 453. The first channel has a frequency (f_1) associated to its center. If a second observation is performed with a different reference frequency, the most probable is that the previous frequency (f_1) does not lay in the center of the matching channel, but off from the center. To average both channels (which have different numbers as shown in the figure) we can only sum and divide by 2 the intensity associated to each channel. CLASS provides several possiblities of aligning: "frequency align", "velocity align" or "channel align". When averaging channels whose central frequency differs the resulting channel will have a final frequency which is the average (weighted or not) from the original channels. If the alignment is done by frequency, the edges of the channels cannot match exactly and some resampling in the X axis is required. This is exactly what we see in Fig. 4. This nasty effect can be avoided by performing observations whose frequency is shifted verifying equation 3 as shown in Fig. 5 and 6. In such way we guarantee that the central frequency of the channels match exactly.

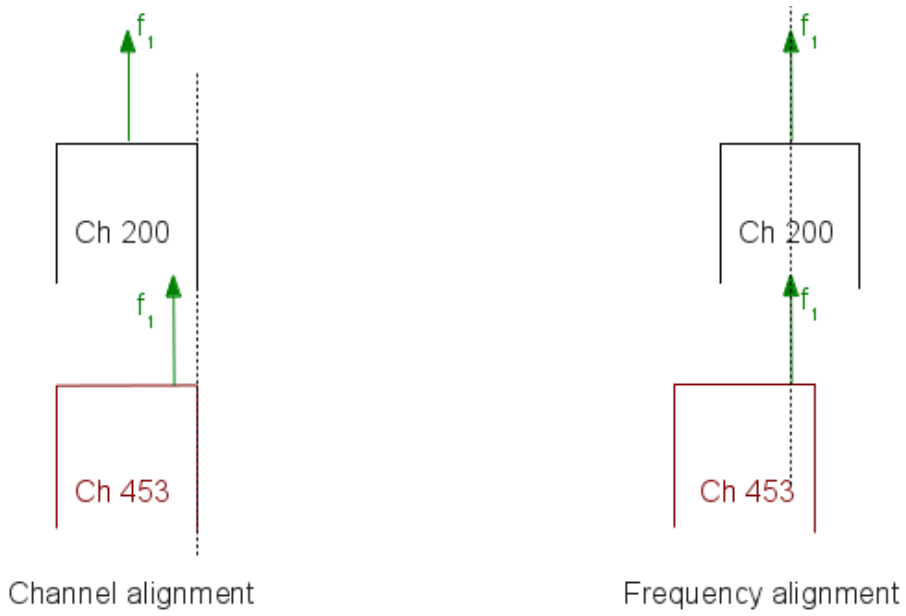

Figure 7: *Two almost matching channels from two observations performed with different reference frequencies. Channels are arbitrary and just for clarity have been assumed to be 200 and 453 respectively. Their central frequencies do not match. If the channels are aligned their central frequency do not match. If the frequencies are aligned the channel edges do not align.*

5 Pyclass filler

This section is only partially related to the main goal of this report but we believe it is interesting to summarize how the current pipeline creates CLASS compatible files and how the doppler information is taken into account.

Since May 2013 it is possible to write header values using a python package. Previous to that moment the procedure used at the 40m radiotelescope was a bit cumbersome: a Gildas compatible FITs file was created on the fly, this file was imported by CLASS which wrote the spectra on a data file. Currently the method is straightforward and we describe it below.

The pipeline software is written in Python and requires importing the pyclass filler:

```
import pyclassfiller
from pyclassfiller import code
```

The import should work correctly provided the account where the pipeline works has GILDAS python packages properly installed.

The output file has to be created using the following instruction, that we perform in the constructor of the pipeline:

```
self.fileClassOut = pyclassfiller.ClassFileOut()
try:
     self.fileClassOut.open(file=self.fileName, new=False, over=True,
          size=1000000, single=True)
except Exception, e:
     if self.__debug:
          print "Error opening the file: %s" % (str(e) )

     if not os.path.isfile(self.fileName):
          self.fileClassOut.open(file=self.fileName, new=True, over=True,
                size=1000000, single=True)
          if self.__debug:
                 print "Created a new file"
```

The output file can be created using options: `new` to create a new file (True) or append data (False), `over` to overwrite (True) or not the file version (False), `size` to determine the maximum allowed size of the file in number of observations (spectra or drifts), `single` to allow (True) or not (False) only one version of each observation in the file. In the previous code we want to append data if the file already exists. If not we create a new file.

The file will be closed when the component is also closed, that is in the destructor of the main class:

```
self.fileClassOut.close()
```

Data are filled with the following procedure. An object of the class `ClassObservation` is created:

```
try:
   obs = pyclassfiller.ClassObservation()
   #obs.head.presec[:]            = False  # Disable all sections except...
   #obs.head.presec[code.sec.gen] = True  # General
   #obs.head.presec[code.sec.pos] = True  # Position
   #obs.head.presec[code.sec.cal] = True  # Position
```

```
   #if self.obsType == "continuum":
   #       obs.head.presec[code.sec.dri] = True  # Continuum drifts
   #else:
   #       obs.head.presec[code.sec.spe] = True  # Position

   #First the size:
```

Then the data are created and filled using a “numpy” array.

```
   npoin = len(self.tantenna)
   nchannels = npoin
   obs.datay = numpy.array(range(npoin),dtype=numpy.float32)
   i = 0
   for element in self.tantenna:
    obs.datay[i] = element
        i = i + 1
```

and then the header is filled. The header is divided in several sections, which contain different information:
General header:

```
obs.head.gen.num = 0
obs.head.gen.ver = 1
obs.head.gen.teles = self.telbackend
obs.head.gen.dobs = int(self.mjdStart) - 60549
obs.head.gen.dred = int(self.mjdStart) - 60549
obs.head.gen.typec = code.coord.equ
if self.obsType == "continuum":
   obs.head.gen.kind = code.kind.cont
else:
   obs.head.gen.kind = code.kind.spec
obs.head.gen.qual = code.qual.unknown
obs.head.gen.scan = self.scanNumber
obs.head.gen.subscan = subscan
obs.head.gen.ut = self.utcrad
obs.head.gen.st = self.lstrad
obs.head.gen.az = self.azimuth * math.pi /180.
obs.head.gen.el = self.elevation * math.pi /180.
obs.head.gen.tau = self.opac0
obs.head.gen.tsys = self.tsys
obs.head.gen.time = self.integTime
obs.head.gen.xunit = code.xunit.velo  # Unused
```

The meaning of each field is self explanatory or can be learnt by typing in CLASS:

```
help set var write
```

or from the documentation or from the pyclass filler itself
(`gildas-src-may13b/integ/x86_64-debian6-gfortran/python/pyclassfiller/__init__.py`

Some interesting variables are `dobs` and `dred`, observing and reduction date in days respectively from the birth of GILDAS (modified Julian day 60549). The type of coordinates requires to know the coding, as well as the type of observations (spectral or continuum). UTC and sidereal time are in radians as well as azimuth and elevation. `tau` is the opacity towards the zenith and integration time is in seconds.
Position section:

```
obs.head.pos.sourc = self.sourceName
obs.head.pos.epoch = self.equinox
obs.head.pos.lam = float(self.radeg) * math.pi/180.
obs.head.pos.bet = float(self.dec) * math.pi/180.
obs.head.pos.lamof = self.raoff
obs.head.pos.betof = self.decoff
obs.head.pos.proj = code.proj.none
obs.head.pos.sl0p = 0.          #  ?
obs.head.pos.sb0p = 0.          # ?
obs.head.pos.sk0p = 0.          # ?
```

Variables are self explanatory. Units are radians in most variables.
Calibration section:

```
obs.head.cal.beeff = self.beamEff
obs.head.cal.foeff = self.forEff
obs.head.cal.gaini = self.gainImg
obs.head.cal.h2omm = self.h2o
obs.head.cal.pamb = self.pressure
obs.head.cal.tamb = self.tempK
obs.head.cal.tatms = self.tempK - 40.
obs.head.cal.tchop = self.thot
obs.head.cal.tcold = self.tcold
obs.head.cal.taus = self.opacs                  # FIXME
obs.head.cal.taui = self.opaci                  # FIXME
obs.head.cal.tatmi = self.tempK - 40.
obs.head.cal.trec = self.trec                   # FIXME
obs.head.cal.cmode = code.calib.auto
obs.head.cal.atfac = 0.                         # FIXME calf or tcal
obs.head.cal.alti = self.siteHeight
obs.head.cal.count = numpy.zeros((3,),dtype=numpy.float32)      #TODO
obs.head.cal.lcalof = 0.
obs.head.cal.bcalof = 0.
obs.head.cal.geolong = self.siteLongitude
obs.head.cal.geolat = self.siteLatitude
```

Continuum header:

```
#
if self.obsType == "continuum":
    self.centralChannel = self.offIndex + 1
```

```
    obs.head.dri.freq = self.restFreqMHz
    obs.head.dri.width = self.bandwidth
    obs.head.dri.npoin = npoin
    obs.head.dri.rpoin = self.offIndex + 1
    obs.head.dri.tref = 0.                  # TODO
    obs.head.dri.aref = 0
    obs.head.dri.tres = self.integTime
    if self.longitudeDrift == 1:
       obs.head.dri.apos = 1.570796
    else:
       obs.head.dri.apos = 0
    obs.head.dri.ares = self.meanStep * math.pi / 180.
    obs.head.dri.bad = 0.
    obs.head.dri.ctype = code.coord.equ
    obs.head.dri.cimag = 0.
    obs.head.dri.colla = self.corrColAz
    obs.head.dri.colle = self.corrEl
else:
```

The rest frequency and the bandwidth are supplied in MHz. `npoin` is the number of points of the continuum drift and the reference point `rpoin` is also set. If the drift is in azimuth `apos` is π, if it is in elevation it is 0. The distance between points is given in radians. The coordinate system for the drifts is suplied (in this case equatorial system) and pointing corrections in azimuth (in the sky) and elevation are also provided.
Spectral header:

```
   # For the notification channel
   offsetFrequencyAt0 = 0

   if self.deltav == 0:
       self.deltav = - self.clight * self.freqRes / (1000.*self.restFreqMHz)

   if nchannels % 2 == 0:
       self.centralChannel = nchannels/2
       offsetFrequencyAt0 = -self.freqRes / 2.
       #offsetFrequencyAt0 = - self.deltav / 2.
   else:
       self.centralChannel = nchannels/2 + 1
       offsetFrequencyAt0 = 0

   print "centralChannel: ", self.centralChannel, self.deltav, offsetFrequencyAt

   obs.head.spe.restf = self.restFreqMHz
   obs.head.spe.nchan = nchannels
   obs.head.spe.rchan = self.centralChannel
   obs.head.spe.fres = self.freqRes
```

```
obs.head.spe.foff = offsetFrequencyAt0
obs.head.spe.vres = - self.deltav
obs.head.spe.voff = self.vlsr/1000.     # FIXME Does CLASS want it in km/s?
obs.head.spe.bad = -1000.
obs.head.spe.image = 0.
obs.head.spe.vtype = code.velo.lsr      # FIXME
obs.head.spe.doppler = -(self.vobserver + self.vlsr)/self.clight

obs.head.spe.line = self.lineName
```

`rchan` is the reference channel, in our case if the number of channels is odd it is the integer part of the number of channels divided by 2. In this case the reference frequency is assigned to the center of the channel and hence `foff` is 0. If the number of channels is even the reference channel is the number of channels divided by 2 plus 1 and the frequency offset is minus the frequency resolution (channel width) divided by 2. The velocity offset `voff` is the LSR velocity in m/s and the topocentric velocity in units of speed light is stored in `doppler`. To summarize it, we understand that the frequency of each channel (f_n for channel n) at its center should be computed as follows:

$$f_n = f_0 + f_{off} + (c_n - c_r)\, f_{res}(1 - v_t/c)$$

where f_0 is the rest frequency, f_{off} is the frequency offset, c_n is the channel number, c_r is the reference channel, f_{res} the frequency resolution of each channel and v_t/c the topocentric velocity in units of speed of light.

The observation is finally written as follows:

```
obs.write()
```