Reducing data from SALSA in MATLAB – SalsaSpectrum v1.6

Daniel Dahlin

March 11, 2013

Abstract

This document describes the use of the SalsaSpectrum MATLAB class, designed to reduce data from the Salsa Onsala telescopes.

Contents

1 Introduction

The SALSA Onsala telescopes are two 2.3 meter antennas located at Onsala space observatory outside Gothenburg. The telescopes are designed to detect the faint radiation from cold hydrogen gas in our galaxy, the Milky way. The hydrogen gas emits radiation at a wavelength of 21 cm (or a frequency of 1420.4 MHz). This signal can be detected by the microwave receiver in the SALSA telescope which is especially designed for this purpose. The observed spectrum can be exported into a fits-file, which is the currently most used type of file for astronomical images and spectra. The fits file has two parts: (1) the data itself and the (2) header which includes information about the data such as the velocity resolution, central frequency and many others.

Having completed the observations, the observer may want to process the data further in order to more accurately deduce the kinematics, as well as the amount of hydrogen gas in the Milky Way. There are currently several options when reducing data from SALSA, and they are summarized here:

- **Salsa-J**: a software package that can reduce spectral data from the SALSA telscopes, but can also be used as a simple image editor and processor. A spectral module with functionality to reduce SALSA data is available. The main advantage of SalsaJ is its easy-to-use point-and-click interface. The main disadvantage is that reducing many spectra can be tedious. SalsaJ can be downloaded at http://www.euhou.net/index.php/salsaj-software-mainmenu-9 and was developed for the European Hands-On Universe project, an education project aimed to provide interesting exercises in the field of astronomy for high school students.

- **XS** was developed at Onsala Space observatory by Per Bergman. It is aimed toward professional radio astronomers and can handle large amounts of data. It cannot be scripted or called from a scripting language. XS is free and can be downloaded from http://www.chalmers.se/rss/oso-en/observations/data-reduction-software.

- **Class** is another professional software for radio astronomers, and is part of the Gildas package (http://www.iram.fr/IRAMFR/GILDAS/).

- **SalsaSpectrum** is a data reduction environment written in the popular mathematical software MATLAB - aimed for reduction of data from SALSA Onsala, developed by Daniel Dahlin.

There has previously not existed any published MATLAB code to handle data from SALSA (although several students have written their own code over the years). This effort to write a matlab class that reads, reduces and analyzes data from SALSA was made to provide such a code for use with MATLAB. It was also a good exercise for the author in order to learn more about object oriented programming in MATLAB. The class, named `SalsaSpectrum` can be freely used, changed and improved by anyone interested in the project. This document describes the use of the `SalsaSpectrum` class at its current state.

2 The SalsaSpectrum class

The code is implemented as a MATLAB class. A fits-file from the SALSA telescopes is read into memory as an object of the `SalsaSpectrum` class. The user can then perform different operations on the spectrum object, such as fitting a baseline, plot the spectrum, calculate the noise level and fit a number of gaussians to the spectrum. All results can be printed to screen and the commands

used to reduce the data can be placed in a MATLAB m-file for easier use. More complex scripts reducing many spectra at once are also possible.

The different capabilities of the `SalsaSpectrum` class will now be presented.

2.1 Installation

The code used for the reduction environment is contained within one MATLAB m-file which includes the class definitions and functions. No installation of the code is therefore needed. Move the m-file to the directory where you do your work. MATLAB will then automatically find and use the `SalsaSpectrum` class.

If you want to use the class from many locations, you can store the m-file in a central directory, and use the matlab `addpath` command to add that location to the matlab path.

```
addpath /path/to/directory/with/SalsaSpectrum/class/files
```

2.2 Read a spectrum

The class constructor method `SalsaSpectrum` is used to read a spectrum from a fits-file and place it into the `SalsaSpectrum` object `spec`

```
>> spec = SalsaSpectrum('../l120_sw_40s.fits')
```

If everything goes well the following information will be printed

```
>> spec

spec =

  SalsaSpectrum handle

  Properties:
                     c: 299792458
               HIfreq: 1.4204e+09
              fittype: 0
             fileName: '../l120_sw_40s.fits'
                 freq: [1x256 double]
                  vel: [1x256 double]
                index: [1x256 double]
                 info: [1x1 struct]
                 data: [1x256 double]
             baseLine: []
           baseWindow: []
     baseWindowParVel: []
     baseWindowParInd: []
    baseWindowParFreq: []
                  rms: []
             gaussFit: []
          gaussConfInt: []
             gaussPar: []
           gaussParVel: []
          gaussParFreq: []
             gaussErr: []
           gaussErrVel: []
          gaussErrFreq: []
```

```
gaussIntegrated: []
      residuals: []
         labVel: []
         labSig: []
```

Several properties have already been filled. Arrays containing the data, info (the header), and frequency, velocity and index have been created, as well as constants (the speeed of light and the frequency of the HI line). Several properties are not yet filled. The functions that calculate and fill the empty variables are discussed in the following sections.

2.3 Plot a spectrum

It is easy to plot a spectrum. The MATLAB class system allows two types of calls to the plot function. A `SalsaSpectrum` object `spec` may be plotted either by using `spec.plot()` or `plot(spec)`. Using these commands on a SALSA spectrum results in the first graph shown in figure 1.

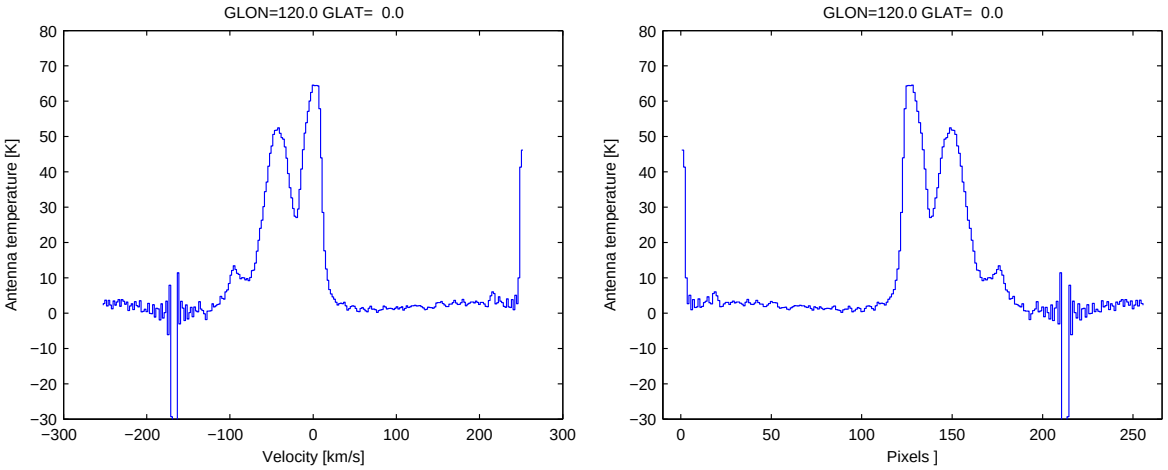

Figure 1: Salsa spectra as a function of velocity and channel index ("pixel").

The `plot` is an overloaded function, i.e. it has the same name as the normal plot command but some extra features that makes the plot more useful. A few such thing are apparent when looking at the spectrum. The `SalsaSpectrum` class has made the decision to display the spectrum as a function of velocity, which is the most useful representation when working with a radioastronomical spectrum. Labels are present for both axes. It is also possible to display the spectrum as a function of frequency or channel index, by sending either `'freq'` or `'pix'` to `plot`. An example of a spectrum plotted in pixel scale is also shown in figure 1.

2.4 Fit and subtract a baseline

In radioastronomical jargon, the "baseline" is the part of the spectrum where no line emission is present. Spectra from SALSA are produced by so-called *frequency switching* which principally removes any continuum signal that is present. However, the switching technique is not perfect and sometimes a spectrum either is offset from the zero-level, or has a small slope from one end to the other. On such occasions it is costumary to fit a baseline to the spectrum. These principles are most easily understood when seeing an example.

Fitting a baseline with `SalsaSpectrum` consists of finding the spectral coordinates of the parts of the spectrum with no signal. The end coordinates of that range are called *spectral windows*. There are currently two ways of defining the spectral windows

1. *Manual method*: define spectral windows on the command line. Useful in scripts.

2. *Interactive method*: define spectral windows using the mouse.

The two methods are described below in more detail.

2.4.1 Manual method

In the spectrum showed in Fig. 1 there is no signal between velocities $+40\ \mathrm{km\,s^{-1}}$ and $+200\ \mathrm{km\,s^{-1}}$. Although there is a negative spike due to RFI (Radio Frequency Interference) at negative velocities, the edge of the spectrum can be used to define another baseline window, $-240\ \mathrm{km\,s^{-1}}$ and $-210\ \mathrm{km\,s^{-1}}$. The `fitBaseline` command fits a polynomial to the data in the baseline windows. In the following example, a polynomial of order $n = 1$ (a straight line) is fitted to the data in the spectral windows between $[-240\ \mathrm{km\,s^{-1}} \leq v \leq -210\ \mathrm{km\,s^{-1}}]$ and $[50\ \mathrm{km\,s^{-1}} \leq v \leq 200\ \mathrm{km\,s^{-1}}]$.

```
>> spec.fitBaseline([-240 -210 50 200],'vel',1)
```

let us now take a look at the spectrum object again.

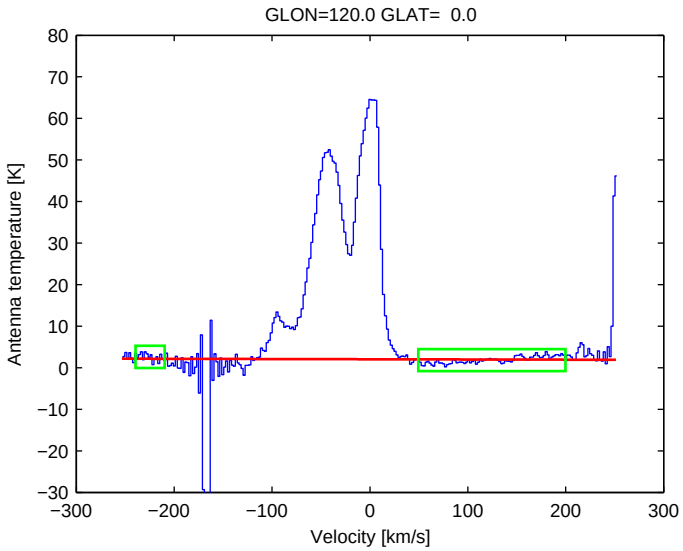

Figure 2: Fitting a baseline. The `showBaseline` command produces this graph showing the fitted baseline as well as the baseline windows used for the fit.

```
>> spec

spec =

  SalsaSpectrum handle

  Properties:
                     c: 299792458
                HIfreq: 1.4204e+09
```

```
          fittype: 0
         fileName: '../l120_sw_40s.fits'
             freq: [1x256 double]
              vel: [1x256 double]
            index: [1x256 double]
             info: [1x1 struct]
             data: [1x256 double]
         baseLine: [1x256 double]
       baseWindow: [1x93 double]
 baseWindowParVel: [-239.4155 -209.7359 49.4660 199.8427]
 baseWindowParInd: [249 234 103 27]
baseWindowParFreq: [1.4216e+09 1.4214e+09 1.4202e+09 1.4195e+09]
              rms: 0.8904
         gaussFit: []
     gaussConfInt: []
         gaussPar: []
      gaussParVel: []
     gaussParFreq: []
         gaussErr: []
      gaussErrVel: []
     gaussErrFreq: []
  gaussIntegrated: []
        residuals: []
           labVel: []
           labSig: []
```

The object has now been filled with more information related to the baseline fit. The coefficients of the fitted polynomial, as well as the actual polynomial as well the indices for the baseline windows in all units have been added. Finally, the task has calculated the rms value of the data in the baseline windows.

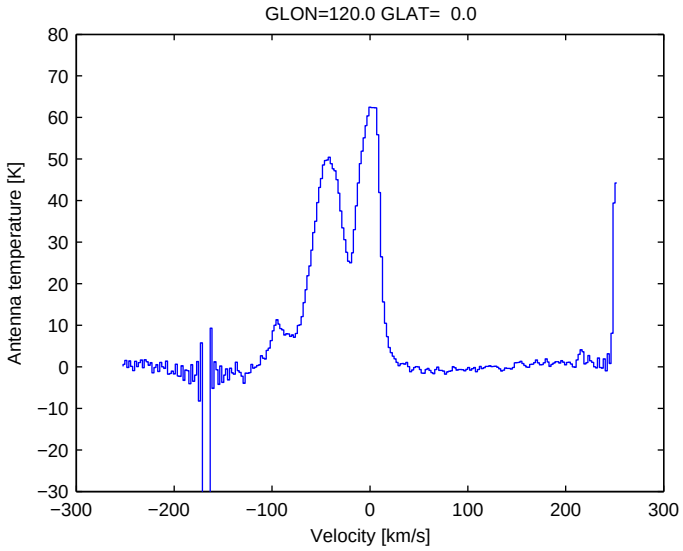

Figure 3: A baseline-subtracted spectrum plotted using `spec.plot()`.

To show the fitted baseline, use the command `showBaseline`.

```
>> spec.showBaseline()
```

which produces the following plot (Figure 2).

At this point, if you are satisfied with the baseline fit, you can subtract the baseline from the spectrum with the function `subtractBaseline`. The syntax is the same as before:

```
>> spec.subtractBaseline()
```

In case you are not satisfied with the fit and want to change some parameters, just redo the fit with new baseline windows or change the order of the polynomial (as a rule-of-thumb, use `order` ≤ 3 for good results). The new fit overwrites the previous one. Once you are satisfied with the fit, use `subtractBaseline` to subtract the baseline. A baseline-subtracted spectrum is shown in Fig. 3.

It is not possible to subtract a baseline twice.

2.4.2 Interactive fit

If you prefer to use the mouse to mark the baseline windows, you can try the interactive method. Invoke the function `fitBaseline` without arguments

```
>> spec.fitBaseline()
```

The following text is shown in the command window:

```
Mark each baseline window by clicking twice, on each side of the window.
You can mark several windows but it is important that you only
click an even number of times.
Press return when you are finished.
```

The baseline windows are then defined by marking the left and rightmost channel of each window with the mouse. A crosshair appears when the function is invoked. Left-click with the mouse to start defining the baseline windows. Between pairs of points the color changes, making it easier to see which pair a point belongs to. When finished, press enter. After that, focus is given back to the command window where you are asked to give the polynomial order.

```
Input polynomial order: 3
```

After that the same fitting routine is used as in the manual method.

2.5 Fit gaussians to the spectrum

The `fitGaussians` function fits up to five Gaussian functions to the spectrum. There are from version 1.7 of the code three ways of fitting gaussians to a spectrum

1. *Manual method*: supply starting guesses for the peak value, central velocity and velocity width of each gaussian function to be fitted.
2. *Blind fit*: allow the software to find peaks and fit Gaussians to them. This works well when the spectral lines are separated, but is not as good when fitting to blended lines.
3. *Interactive fit*: mark peaks in the spectrum with the mouse and use those as starting guesses for the Gaussian fit.

2.5.1 Manual method

Starting guesses are entered in velocity units. The spectrum plotted in Figures 1, 2 and 3 appears to consist of three peaks (at velocities about 0, -40 and -90 $\mathrm{km\,s^{-1}}$). The starting guesses for fitting three gaussian functions using the `fitGaussians` function would then be

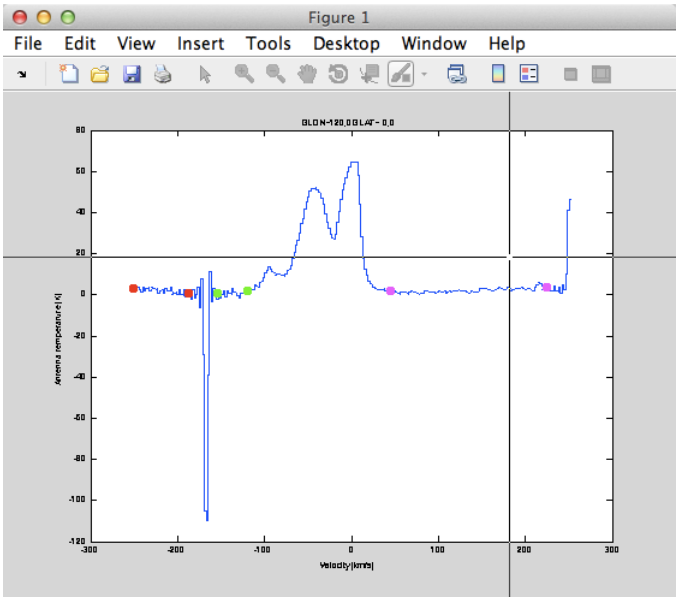

Figure 4: Screenshot from interactive baseline fitting

```
>> spec.fitGaussians([60 0 10 40 -40 12 10 -90 9])
Fitting 3 Gaussians.
Use plot() to see the fitted Gaussians.
```

When the fit is completed, the spectrum object `spec` has been extended with further parameters.

```
>> spec

spec =

  SalsaSpectrum handle

  Properties:
                     c: 299792458
               HIfreq: 1.4204e+09
              fittype: 0
             fileName: '../l120_sw_40s.fits'
                 freq: [1x256 double]
                  vel: [1x256 double]
                index: [1x256 double]
                 info: [1x1 struct]
                 data: [1x256 double]
             baseLine: [1x256 double]
           baseWindow: [1x93 double]
     baseWindowParVel: [-239.4155 -209.7359 49.4660 199.8427]
     baseWindowParInd: [249 234 103 27]
    baseWindowParFreq: [1.4216e+09 1.4214e+09 1.4202e+09 1.4195e+09]
                  rms: 0.8876
             gaussFit: [1x256 double]
         gaussConfInt: [256x2 double]
             gaussPar: [1x9 double]
          gaussParVel: [1x9 double]
```

```
    gaussParFreq: [1x9 double]
        gaussErr: [4.7456 0.5043 0.4973 3.8144 0.8286 1.0231 ...]
     gaussErrVel: [4.7456 0.9978 0.9839 3.8144 1.6395 2.0244 ...]
    gaussErrFreq: [4.7456 0.0047 0.0047 3.8144 0.0078 0.0096 ...]
gaussIntegrated: [1.5583e+03 2.0376e+03 219.7342]
       residuals: [1x256 double]
          labVel: []
          labSig: []
```

The parameters `gaussFit`, `gaussPar`, `gaussParVel` and `gaussParFreq` are now non-empty. `gaussFit` is an array with the best-fit sum of gaussians. The three other parameters hold the best-fit values for the peak, central value and width of the gaussians.

The guesses should be quite close in order for the fit to be good. When the fit is done the `spec.plot` command is used to show the fitted gaussian functions.

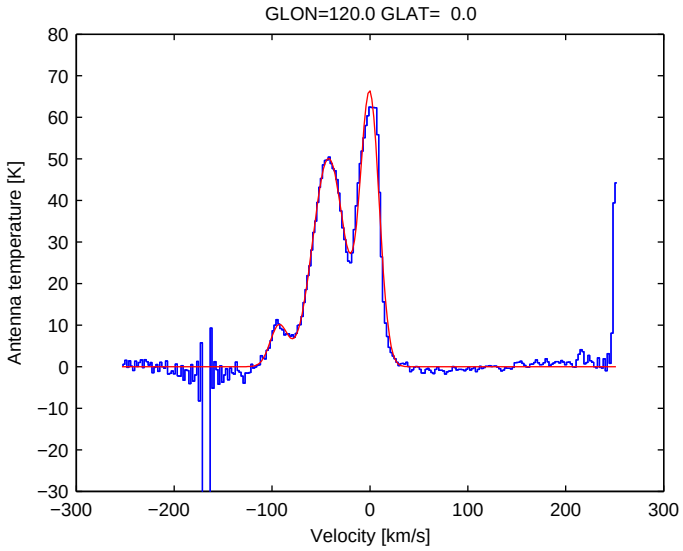

Figure 5: Plot of a spectrum with three fitted gaussians overlaid.

2.5.2 Automatic method

The automatic method will search for peaks in the spectrum. It will then fit Gaussians with the peaks as starting guesses for amplitude, velocity center and width. To use the automatic method, call the `fitGaussians` function without arguments.

```
>> spec.fitGaussians()
```

An example of this fitting method is shown in Figure 6. First the automatic fit is applied, finding three gaussians. The fit at the negative velocities could be improved, so two extra gaussians are manually added using the following syntax.

```
>> spec.fitGaussians([guesses], 'dummy')
```

"Dummy" could be number or string. This refits the previous gaussians together with the new which can improve the total fit to the spectrum. The parameter `guesses` has the exact same format as in the manual fitting method, i.e. [*Gauss peak 1*, *Gauss center 1*, *Gauss width 1*, *Gauss peak 2* ...].

It is also possible to show the individual gaussians in a fit overlaid on the spectrum. Call the `plot` function with the following syntax.

```
>> spec.plot('vel','individual')
```

where `'freq'` or `'pix'` could also be used.

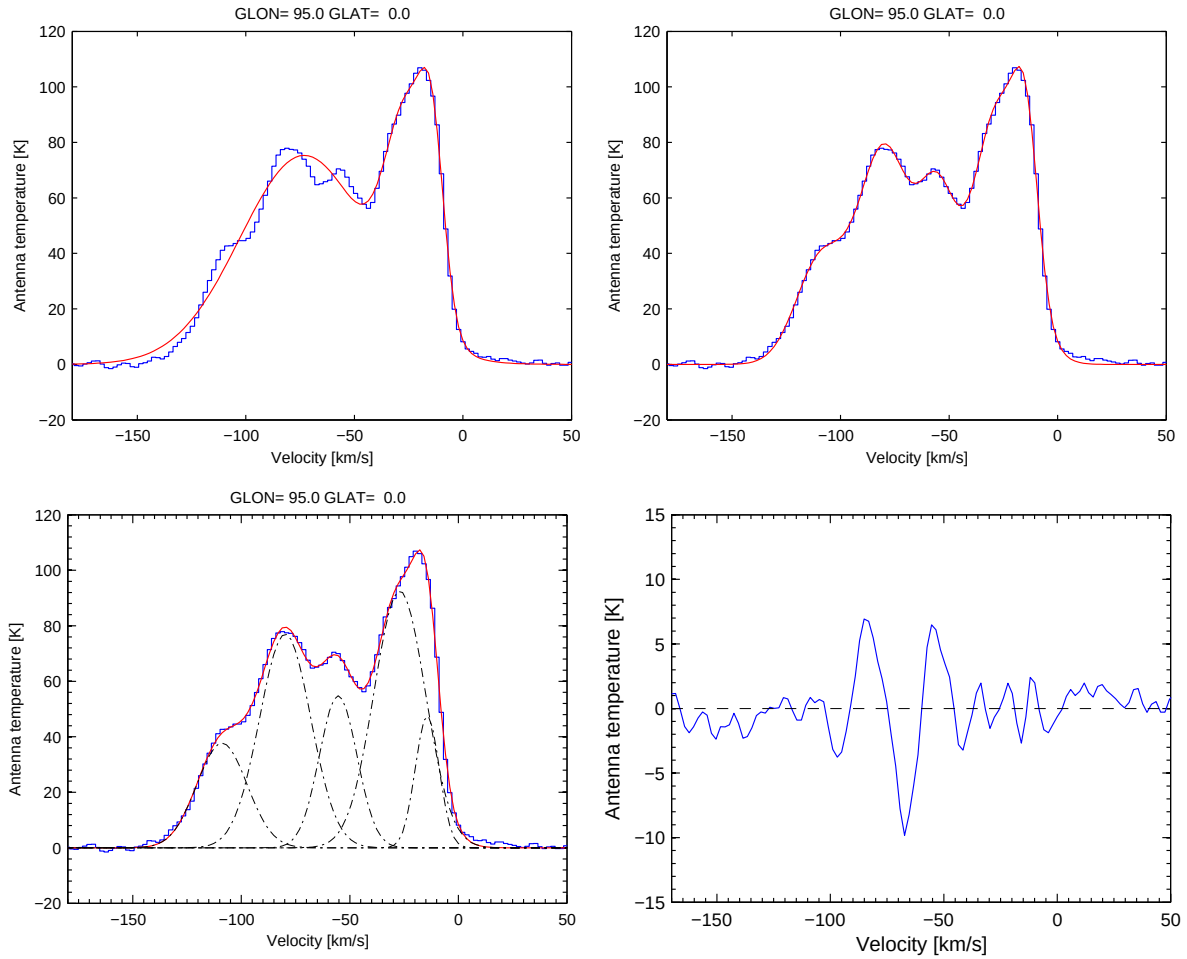

Figure 6: Zoom-in on fit to blended spectrum. *Top left*: blind fit. The algorithm cannot distuingish between the blended peaks between -120 and -50 km s^{-1}. *Top right*: refining the blind fit by adding two additional gaussians. *Bottom left*: same as top right, but also showing the individual gaussian functions. *Bottom right*: residuals *data* $-$ *Gaussian fit* shown with the `showResiduals` function.

2.5.3 Interactive method

From version 1.7 of `SalsaSpectrum`, an interactive Gaussian fit is included. It is designed as a wrapper around the `fitGaussians` function, and it is called without arguments.

```
>> spec.fitGaussiansInteractive()
```

When this function is invoked, the user can mark peaks in the displayed spectrum with the mouse. Press the enter key when you are finished marking the peaks. The data will then be sent to `fitGaussians` which performs the actual fitting. An example is shown in figure 7.

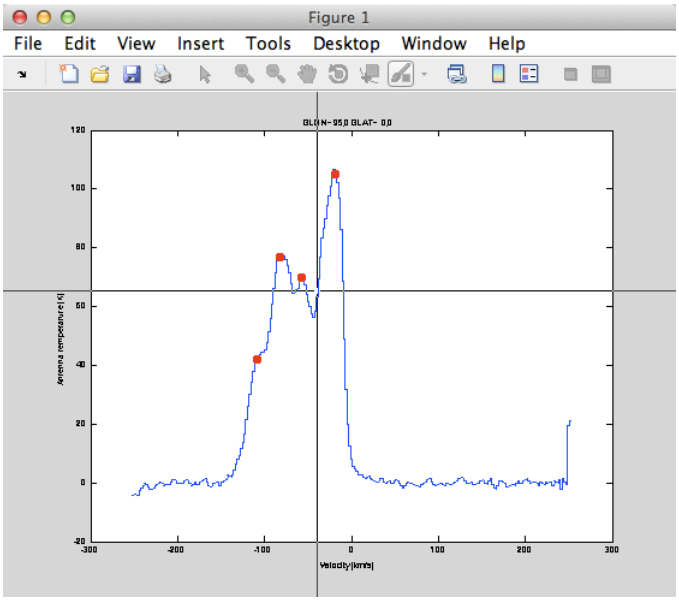

Figure 7: Screenshot from interactive Gaussian fitting

2.5.4 Extracting fit parameters

After performing the gaussian fit, all fit parameters are available within the `SalsaSpectrum` object. Of most interest to users is probably the peak intensity of each Gaussian, their central velocity and width. The integrated intensity of each gaussian is also calculated.

Using the fit to the spectrum presented in figure 6 above, the central velocities of the five gaussians are extracted with the commands:

```
>> spec.gaussParVel(1:3:end)  % peak of each gaussian
   47.6   92.5   77.0   54.8   37.7
>> spec.gaussParVel(2:3:end)  % central velocity of each gaussian
  -14.4  -27.0  -79.8  -55.4 -108.9
>> spec.gaussParVel(3:3:end)  % width (1sigma) of each gaussian
    5.5   11.9   11.8    8.7   11.5
```

and the integrated intensity:

```
spec.gaussIntegrated
```

Errors on these parameter values are also calculated. They are kept in the `gaussErr`, `gaussErrVel` and `gaussErrFreq` properties of the `SalsaSpectrum` object, as soon as Gaussians have been fitted. Finally, a 68% confidence interval on the total sum of Gaussians is calculated, and saved in the property `gaussConfInt`. To visualize the confidence interval, use the `showConfInt` function.

```
>> spec.showConfInt()
```

which for the spectrum and Gaussian fit displayed in Figure 5, produces the plot shown in Figure 8.

2.6 Saving your work

A reduced spectrum can be saved so that the fitted baseline and gaussians can be brought back later, without having to redo the reduction. The built-in MATLAB functions `save` and `load` can

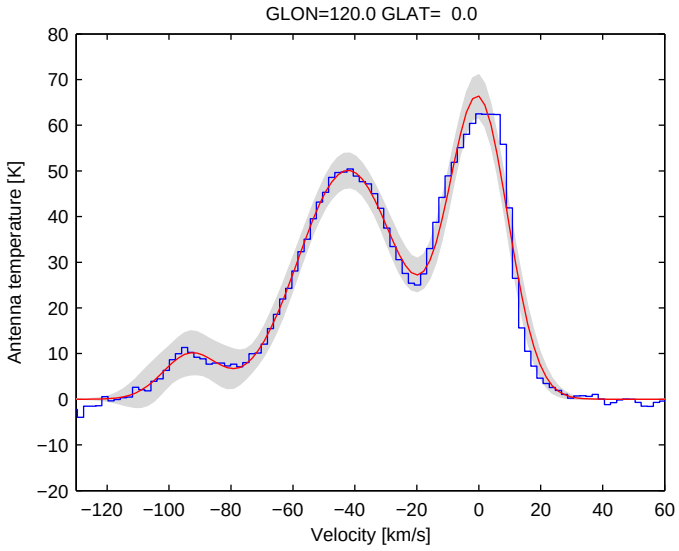

Figure 8: Spectrum with Gaussian fit and confidence intervals overlaid.

be used to save and load a spectrum. To save the spectrum stored in the object `spec1` to the MATLAB `.mat` file `spec1.mat`, simply type:

```
>> save spec1.mat spec1
```

This command places the .mat file spec1.mat in the current directory. To read the file and bring back the SalsaSpectrum object, type

```
>> load spec1.mat
```

Note that to load a spectrum from a mat-file, the `SalsaSpectrum` class must either be stored in the working directory, or added to the MATLAB path.

2.7 Getting help

Most of the functionality of `SalsaSpectrum` is described in this manual. Online help is available in MATLAB for several of the functions, using the `help` syntax. Type `help functionName`. An example for the `fitBaseline` function follows.

```
>> help fitBaseline
 --- help for SalsaSpectrum/fitBaseline ---

  fit a polynomial to the spectrum baseline. Define the
  baseline using a set of windows in the following way:

  [ x11 x12 x21 x22 x31 x32 ... ]

  as the first parameter passed to the function.

  check that the baseline vector consists of an even
  number of values.

   supply the baseline windows in units of
```

```
1. indices      ['pix']
2. velocity     ['vel']
3. frequency    ['freq']

by sending the string as a second argument. If you do not
give any units, it is assumed that you have given indices.

By default, this routine fits a first order polynomial
to the chosen spectral windows. If you want another
polynomial order, supply it as the third parameter.

spec.fitBaseline([-200 -150 40 200], 'vel',2)

will fit a second order polynomial between velocities -200
to -150 and 40 to 200
```

3 Cookbook

How was the spectrum presented in figure 6 reduced? Here are the steps summarized. This data reduction is sufficient for most spectra.

```
% read the spectrum
   spec=SalsaSpectrum('../SalsaFits/obs20060314/00031c.fits');
% fit a baseline
   spec.fitBaseline([-220 -150 30 220],'vel',3)
% subtract the baseline
   spec.subtractBaseline()
% perform blind gaussian fit
   spec.fitGaussians()
% refine the blind gaussian fit
   spec.fitGaussians([30 -55 6 30 -110 5], 'dummy')
% read the LAB spectrum
   spec.readLab
% plot everything
   spec.plot; spec.showLab
```

4 Additional functions

There is further functionality in the `SalsaSpectrum` class which are intended for more advanced users. They are listed here.

4.1 Download data from the LAB survey

The LAB survey is a compilation of three major HI surveys performed with telescopes in the Netherlands, Argentina and Australia. It covers the entire sky and has a resolution of 30 arcminutes. It can be useful to compare the observed spectra taken with SALSA with those from the LAB survey.

LAB spectra can be viewed and dowloaded at http://www.astro.uni-bonn.de/hisurvey/profile/. The high-resolution spectra can also be convolved to the resolution of the SALSA telescopes ($\sim 6.4^\circ$ FWHM). The LAB data are available for download as ascii-files.

`SalsaSpectrum` includes functionality so that the user can automatically download and display a spectrum from LAB. This function (`readLab`) downloads a spectrum at the coordinates of the current spectrum. This function can be called for any `SalsaSpectrum` object.

```
>> spec.readLab
```

To reduce the use of the LAB server, the function also checks if the appropriate data has already been downloaded. In figure 9 a spectrum from SALSA toward the calibration region S7 ($l = 132^\circ$, $b = -1^\circ$) is compared to the LAB spectrum.

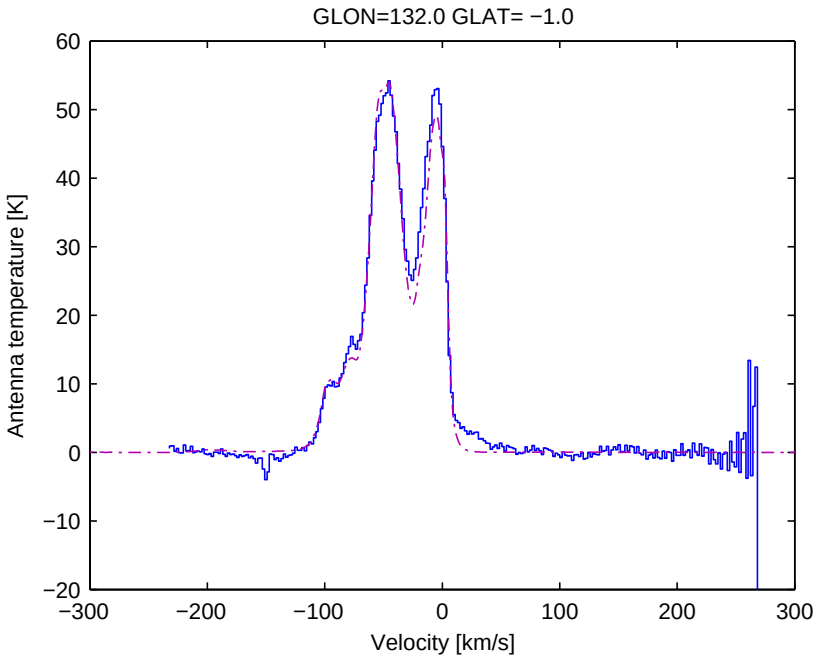

Figure 9: SALSA spectrum taken toward S7 compared with the LAB spectrum, convolved to 6.4° resolution.

The spectra from LAB are presented in the brightness temperature scale. The absolute calibration of data from SALSA is still uncertain. There may therefore be differences between the LAB and SALSA spectra due to calibration.

Reference to the LAB survey: Kalberla, P.M.W., Burton, W.B., Hartmann, Dap, Arnal, E.M., Bajaja, E., Morras, R., & Pöppel, W.G.L. (2005), A&A, 440, 775

4.2 Reduce several spectra

The `SalsaSpectrum` class is easily extensible to allow reduction of large sets of spectra from SALSA. If several fits files from SALSA are stored in a directory, the following code will read them and store them in the MATLAB cell array `spec`. It also fits baselines with different baseline windows depending on which

```
direc = '/home/daniel/SalsaSpectra/SalsaFits/mwscan_l201210/'
files = dir([direc,'*.fits']);

for i = 1:length(files)

    fname = [direc,files(i).name];
    spec{i} = SalsaSpectrum(fname);

    longspec(i) = spec{i}.getKeyword('CRVAL2');
    l = longspec(i);

    if (l>0 & l<40)
        spec{i}.fitBaseline([-240 -205 -140 -100 120 220],'vel',3);
    elseif (l>=40 & l<90)
        spec{i}.fitBaseline([-230 -200 -150 -110 100 220],'vel',3);
    elseif (l>=90 & l<180)
        spec{i}.fitBaseline([-230 -190 -130 -110 50 220],'vel',3);
    elseif l>=180
        spec{i}.fitBaseline([-230 -180 -120 -40 60 220],'vel',3);
    end

    spec{i}.showBaseline()
    spec{i}.subtractBaseline()
    spec{i}.fitGaussians()
    spec{i}.plot()

end % end loop over spectra
```

5 List of functions and properties

5.1 List of properties

```
properties (Constant)
    c = 2.99792458e8;           % speed of light
    HIfreq = 1.42040575177e9;   % HI frequency in Hz
end

properties (GetAccess = 'private', SetAccess = 'private')
    baseSubtracted
    coordType
    gaussiansFitted
end

properties(GetAccess = 'public', SetAccess = 'private')
    % public read access, but private write access.
    fileName
end

properties
    freq                % frequency array
    vel                 % velocity array
    index               % spectrum index array
    info                % fitsinfo
    data                % data spectrum
    baseLine            % fitted polynomial baseline
    baseWindow          % all indices of the baseline windows
    baseWindowParVel    % velocity values of start-end baseline window
    baseWindowParInd    % index values -"-
    baseWindowParFreq   % frequency values -"-
    rms                 % rms value calculated in the baseline windows
    gaussFit            % fitted gaussian functions
    gaussConfInt        % 68% confidence interval on the gaussian fit
    gaussPar            % parameters of gaussians
    gaussParVel         % -"- in velocity units
    gaussParFreq        % -"- in frequency units
    gaussErr            % 1 sigma errors on gaussians
    gaussErrVel         % -"- in velocity units
    gaussErrFreq        % -"- in frequency units
    gaussIntegrated     % integrated intensity in K km/s
    residuals           % residuals data - gauss fit
    labVel              % velocity axis of LAB spectrum
    labSig              % data of LAB spectrum
end % properties
```

5.2 List of functions

A complete list of the functions within the `SalsaSpectrum` class follows, along with the information in each functions's help file.

SalsaSpectrum

```
Class to initialize and do operations on spectra from the Salsa
Onsala telescopes and the Qradio software. Create a spectrum object
using the syntax

spec = SalsaSpectrum('filename.fits');

which then can be displayed and analyzed.


This software comes with no warranty. It has been thoroughly
tested, but there may still be bugs. It can be downloaded on the
SALSA onsala Web site at brage.oso.chalmers.se , under "Salsa
data in \textsc{Matlab}".

A manual describing how the class can be used is posted at together
with this matlab class at
http://brage.oso.chalmers.se/salsa/?q=node/142
```

plot

```
HANDLE = PLOT(obj,varargin)

Plot a spectrum. The default syntax

spec.plot()

By default the velocity scale is used. Other options are
'pix' for indices and 'freq' for frequency.

spec.plot('freq')

If you have fitted Gaussians, they will also be
displayed.

If a second 'dummy' argument is supplied, and gaussians have
been fitted, the individual Gaussians will be shown.

spec.plot('vel', 'dummy')
```

showResiduals

```
SHOWRESIDUALS(obj)

Plot the residuals; the difference between the data and
the Gaussian model.
```

showIndividual

```
HANDLE = SHOWINDIVIDUAL(obj,varargin)
```

```
plots the individual gaussian functions fitted to the spectrum. Can
only be called after fitGaussians has ben used.
```

shiftVel

```
SHIFTVEL(obj,deltav)

Shift the velocity scale by a value. To shift the scale 20 km/s
toward positive velocities, use

spec.shiftVel(-20)
```

scale

```
SCALE(obj,scalefac)

Scale the spectrum and fitted Gaussians by a factor, by
multiplication. To scale a spectrum down by 20 percent, use the
command

spec.scale(8/10)
```

fitBaseline

```
FITBASELINE(obj,varargin)

fit a polynomial to the spectrum baseline. Define the
baseline using a set of windows in the following way:

[ x11 x12 x21 x22 x31 x32 ... ]

as the first parameter passed to the function.

The baseline vector should consist of an even number of
values. If not, the function gives an error.

The baseline windows are supplied in units of

1. indices     ['pix']
2. velocity    ['vel']
3. frequency   ['freq']

Send the string (e.g. 'vel') as a second argument. The index
unit is default.

By default, this routine fits a first order polynomial to
the chosen spectral windows. Supply another polynomial order
as an argument

spec.fitBaseline([-200 -150 40 200], 'vel', 2)
```

```
This command will fit a second order polynomial between
velocities -200 to -150 and 40 to 200
```

getIndices

```
help function for the fitBaseline function
```

showBaseline

```
SHOWBASELINE(obj)
Show the fitted baseline overlaid on the spectrum, as well as boxes
indicating the baseline windows.

spec.showBaseline()

This function does not subtract the baseline. Use the
subtractBaseline function to do that.
```

subtractBaseline

```
SUBTRACTBASELINE(obj)

Subtract the fitted baseline from the data.
```

fitGaussians

```
FITGAUSSIANS(obj,varargin)
Fit a number of gaussians to the spectrum. Supply guess
values of height, central value and width of each
gaussian, in units of velocity.

spec.fitGaussians([60 0 8 30 -55 10])

If you don't supply any guess, the function will search
for peaks in the spectrum and use those as starting
guesses for the fit. If the fit is not good, you can
redo it with a new guess.

If you want to add gaussians to a fit, call this
function again with guesses for the new gaussian, and
send a second dummy argument, like

spec.fitGaussians([60 -10 10],'dummy')

which will add an extra gaussian of peak 60 K, central
velocity -10 km/s and velocity width 10 km/s to the fit.
```

getRms

```
GETRMS(obj)

calculate and display the rms of the data contained in the baseline
windows.
```

showBaselineWindows

```
SHOWBASELINEWINDOWS(obj)

Plot a graphical representation of the chosen
baseline windows.
```

getKeyword

```
GETKEYWORD(OBJ, KEY) returns the value of KEY from the fits
header OBJ.INFO.

This function was kindly provided by Magnus Sand n and Eskil
Varenius.
```

showLab

readLab

```
READLAB(obj)

download data from the LAB survey at
http://www.astro.uni-bonn.de/hisurvey/profile/index.php

convolved to Salsa's angular resolution
```

clipSpectrum

```
BACK = CLIPSPECTRUM(obj,window)

NOTE: EXPERIMENTAL, USE AT YOUR OWN RISK.

given the indices in the window parameter, this
function "clips" (removes) the indicated spectral
channels from the data. Several spectral windows can
be specified.  Only the 'pixel' spectral unit can be
used.

  spec.clipSpectrum([1 10 230 250])
```

```
would clip all channels between channels 1 and 10 (1 2 3
4 5 6 etc) and between channels 230 and 250.
```

despike

```
DESPIKE(obj,varargin)

NOTE: EXPERIMENTAL, USE AT YOUR OWN RISK.

experimental function to remove 'spikes' in the data. Works well for
the spike at the edge of the spectrum, works
sometimes also for strong RFI in the spectrum.
```

smoothSpectrum

```
SMOOTHSPECTRUM(obj,varargin)

Smooth a spectrum to a lower spectral resolution. To lower
the resolution by a factor of 2, supply the command

spec.smoothSpectrum(2)
```

6 Changelog

version 1.8

- Rewrote the `readLab` function so that it now also works on Windows machines.

version 1.7

- Added information about interactive fitting of baselines and Gaussians

Version 1.6

- added documentation for all functions
- tidied up the document

Version 1.3

- Major reorganisation of the text
- added figures and examples

Version 1

- Initial documentation

7 Conclusions and outlook

This document is a brief manual in the use of the `SalsaSpectrum` MATLAB class. The manual is intended to reflect the methods available in the MATLAB class. However, it is possible that this documentation will lag the code itself. My hope is that the code will be useful for some people interested in reducing data from SALSA.

Göteborg March 11, 2013

Daniel Dahlin