# Astroinformatics School 2009

**ASAP Component on Friday 17 April 2009**

## Tutorial 5 – Flagging the Lag Spectrum and Baseline Ripples

**Prepared by:** Andrew Walsh

### File Information:

| | |
|---|---|
| orion.dat | Broadband 12mm spectrum of Orion-KL (partially reduced) |
| baseline.py | Data reduction script |
| flagging.py,f2.py | Auxillary flagging scripts called by "baseline.py" |
| r.py | partial reduction script |

### Introduction:

The data were taken over about 40 hours of observations with Mopra. They consist of a single broadband spectrum from 19.5 to 27.5GHz. The purpose of the observations was to look for weak spectral features due to Glycine – the simplest amino acid. The spectrum does show a rich variety of spectral lines but also shows the baseline ripple quite prominently. The task here is to try and reduce the baseline ripple so that the weakest (real) spectral line features can be seen. This is done with the command "lag_flag".

Lag_flag fourier transforms the data and then flags out data in the fourier domain according to specified frequencies. This is useful for the Mopra baseline ripple as it appears in the spectrum as a sinusoidal signal with frequency about 30MHz.

Questions: What wavelength does this correspond to?

What is the significance of this wavelength?

**Instructions:**

1. Load in the spectrum of Orion and display it. This can be done with:

   ```
   localhost> asap
   s = scantable('orion.dat')
   s.set_unit('GHz')
   plotter.set_mode('i','s')
   plotter.plot(s)
   ```

2. Observe what happens when applying a lag_flag command on the data. Eg:

   ```
   iav = s.copy()
   iav.lag_flag(30,6,unit="MHz")
   plotter.plot(iav)
   ```

3. Use the flagging scripts (flagging.py and f2.py) to remove the strongest lines from the spectrum. Then observe what happens with lag flagging. Eg:

   ```
   del iav
   iav = s.copy()
   execfile('flagging.py')
   execfile('f2.py')
   plotter.plot(iav)
   iav.lag_flag(30,6,unit="MHz")
   plotter.plot(iav)
   ```

4. Baseline flagged spectrum to reduce edge effects and then look at the result. Eg:

   ```
   del iav
   iav = s.copy()
   execfile('flagging.py')
   execfile('f2.py')
   b = iav.copy()
   b.poly_baseline(order=3,plot='True')
   b.lag_flag(30,6,unit="MHz")
   plotter.plot(b)
   ```

5. Remove ALL the low frequency ripples in the spectrum after flagging and baselining the data with:

   ```
   del b
   b = iav.copy()
   b.poly_baseline(order=3,plot='True')
   c = b.copy()
   c.lag_flag(12,11.99999,unit="MHz")
   d = b-c
   plotter.plot(d)
   ```

   You'll notice that the above commands do three unexpected things. What are they?

6. Now the basic process is outlined, your task is to play around with the various options for baselining, flagging and lag_flagging to produce the best spectrum. Note that best spectrum is one that most clearly shows the weakest lines WITHOUT significantly reducing the intensity of those lines through the lag_flagging process.

*Estimated time to complete ~ 40 mins*