Notebook - March 17 - Part 2

March 21, 2016

1 Reading FITS spectra

```
In [197]: %%bash
          curl -O https://raw.githubusercontent.com/astropy/specutils/master/specutils/io/tests/files/gl
          ls -lh gbt_1d.fits
```

```
-rw-r--r--+ 1 adam staff 40K Mar 17 12:12 gbt_1d.fits

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 40320  100 40320    0     0  56047      0 --:--:-- --:--:-- --:--:-- 56000
```

```
In [198]: ls -lh gbt_1d.fits
```

```
-rw-r--r--+ 1 adam 40K Mar 17 12:12 gbt_1d.fits
```

```
In [13]: pwd
```

```
Out[13]: '/Users/adam/Dropbox/eso_python_2016'
```

```
In [14]: !pwd
```

```
/Users/adam/work/teaching/eso_python_2016
```

```
In [17]: !ls -lh gbt_1d.fits
```

```
-rw-r--r--+ 1 adam staff 40K Mar 17 11:05 gbt_1d.fits
```

```
In [19]: from astropy.io import fits
```

```
In [54]: fh = fits.open('gbt_1d.fits')
         header = fits.getheader('gbt_1d.fits')
         data = fits.getdata('gbt_1d.fits')
```

```
In [55]: len(fh)
```

```
Out[55]: 1
```

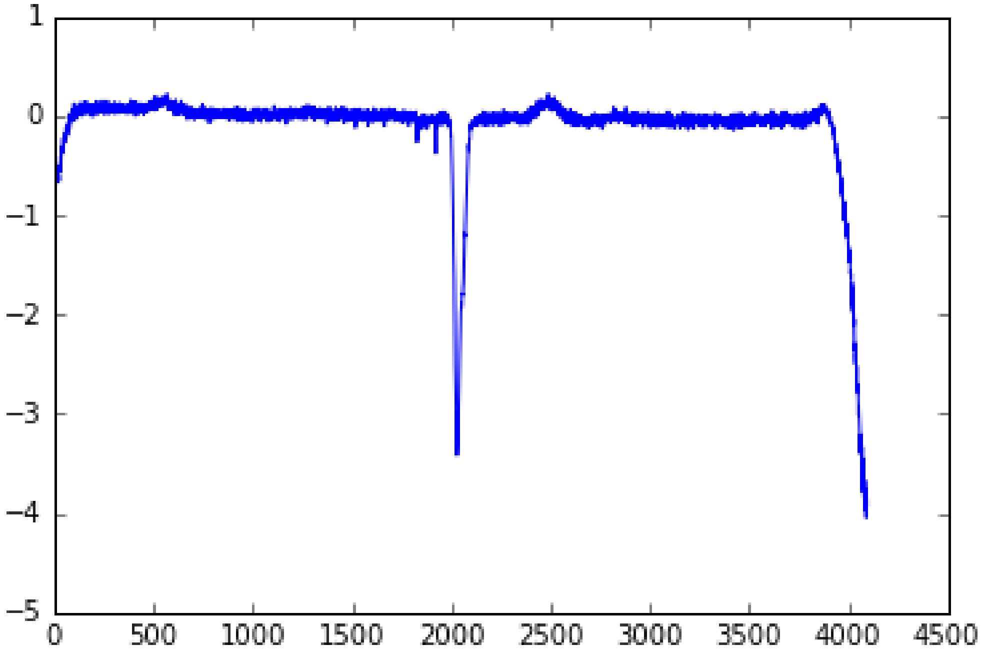
```
In [23]: fh[0]
```

```
Out[23]: <astropy.io.fits.hdu.image.PrimaryHDU at 0x10d8ab978>
```

```
In [26]: fh[0].data.shape
```

```
Out[26]: (4096,)
```

```
In [28]: %matplotlib inline
         import pylab as pl
         pl.plot(fh[0].data)

Out[28]: [<matplotlib.lines.Line2D at 0x10e9218d0>]
```

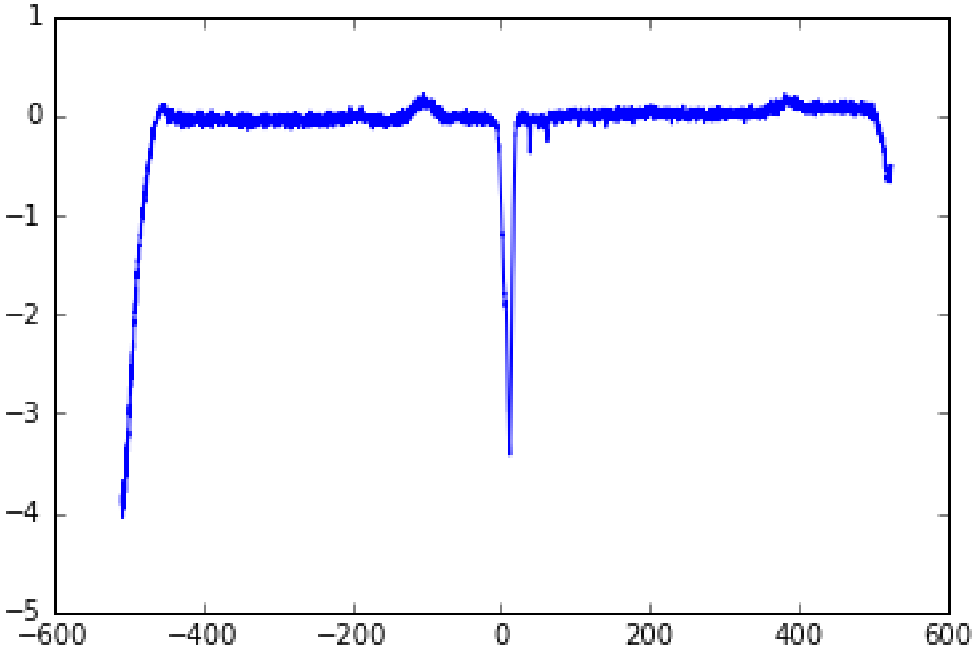


```
In [44]: hdr = fh[0].header

In [59]: d = {'a' : (1,2,3), 'b': 2}

In [60]: d['a'][0]

Out[60]: 1

In [61]: d[0] # not valid!
```

```
        ---------------------------------------------------------------------------

        KeyError                                  Traceback (most recent call last)

        <ipython-input-61-67c04099f1d8> in <module>()
    ----> 1 d[0] # not valid!


        KeyError: 0
```

```
In [62]: hdr['NAXIS1']

Out[62]: 4096
```

```
In [63]: data.size

Out[63]: 4096

In [31]: xarr_pixels = np.arange(hdr['NAXIS1'])
         xarr_pixels

Out[31]: array([   0,    1,    2, ..., 4093, 4094, 4095])

In [65]: cdelt = hdr['CDELT1'] # spacing between pixels
         crval = hdr['CRVAL1'] # reference coordinate
         crpix = hdr['CRPIX1'] # pixel of the reference coordinate
         cunit = hdr['CUNIT1']

In [66]: cdelt, crval, crpix, cunit

Out[66]: (-0.25258831, 7.5845751, 2049.0, 'km/s')

In [33]: # +1 because FITS is 1-indexed, python is 0-indexed
         xarr = (xarr_pixels - crpix + 1)*cdelt + crval

In [39]: cunit

Out[39]: 'km/s'

In [40]: xarr

Out[40]: array([ 524.88543398,  524.63284567,  524.38025736, ..., -508.95851885,
                -509.21110716, -509.46369547])

In [41]: pl.plot(xarr, fh[0].data)

Out[41]: [<matplotlib.lines.Line2D at 0x10dbdfcc0>]
```



```
In [67]: from astropy import units as u

In [68]: u.km/u.s
```

Out[68]:

$$\mathrm{\frac{km}{s}}$$

```
In [71]: xarr_u = xarr * u.km/u.s
         xarr_u
```

Out[71]:

$$[524.88543,\ 524.63285,\ 524.38026,\ \ldots,-508.95852,\ \ -509.21111,\ \ -509.4637]\ \mathrm{\frac{km}{s}}$$

```
In [72]: xarr_u.to(u.m/u.s)
```

Out[72]:

$$[524885.43,\ 524632.85,\ 524380.26,\ \ldots,-508958.52,\ \ -509211.11,\ \ -509463.7]\ \mathrm{\frac{m}{s}}$$

```
In [75]: xarr_u = xarr * u.Unit(hdr['CUNIT1'])
         xarr_u
```

Out[75]:

$$[524.88543,\ 524.63285,\ 524.38026,\ \ldots,-508.95852,\ \ -509.21111,\ \ -509.4637]\ \mathrm{\frac{km}{s}}$$

```
In [77]: type(u.m), type(xarr_u)

Out[77]: (astropy.units.core.IrreducibleUnit, astropy.units.quantity.Quantity)

In [78]: xarr_u.cgs
```

Out[78]:

$$[52488543,\ 52463285,\ 52438026,\ \ldots,-50895852,\ \ -50921111,\ \ -50946370]\ \mathrm{\frac{cm}{s}}$$

```
In [80]: xarr_u.si
```

Out[80]:

$$[524885.43,\ 524632.85,\ 524380.26,\ \ldots,-508958.52,\ \ -509211.11,\ \ -509463.7]\ \mathrm{\frac{m}{s}}$$

```
In [82]: (500*u.M_jup/u.yr).to(u.g/u.s)
```

Out[82]:

$$3.0083086 \times 10^{25}\ \mathrm{\frac{g}{s}}$$

```
In [83]: from astropy import constants

In [85]: # how many protons in earth?
         constants.M_earth / constants.m_p
```

Out[85]:

$$3.5717579 \times 10^{51}$$

```
In [89]: xarr_u.unit.to_string(format='latex')

Out[89]: '$\\mathrm{\\frac{km}{s}}$'

In [91]: xarr_u.unit.to_string()

Out[91]: 'km / s'

In [92]: np.cos(5*u.rad)

Out[92]:
```

$$0.28366219$$

```
In [93]: np.cos(45*u.deg)

Out[93]:
```

$$0.70710678$$

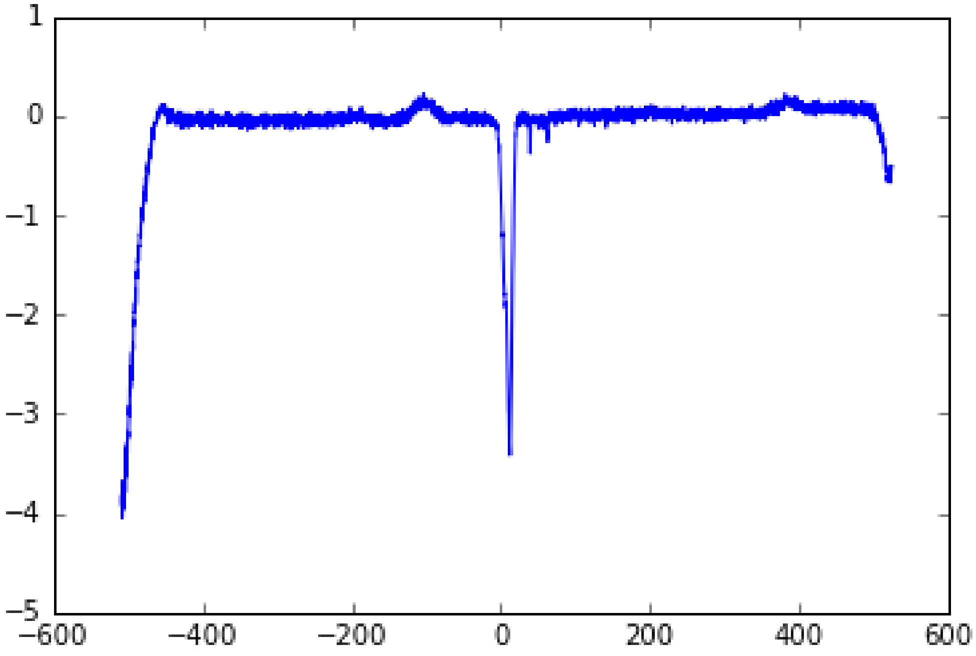
```
In [95]: np.cos(45*u.arcsec)

Out[95]:
```

$$0.99999998$$

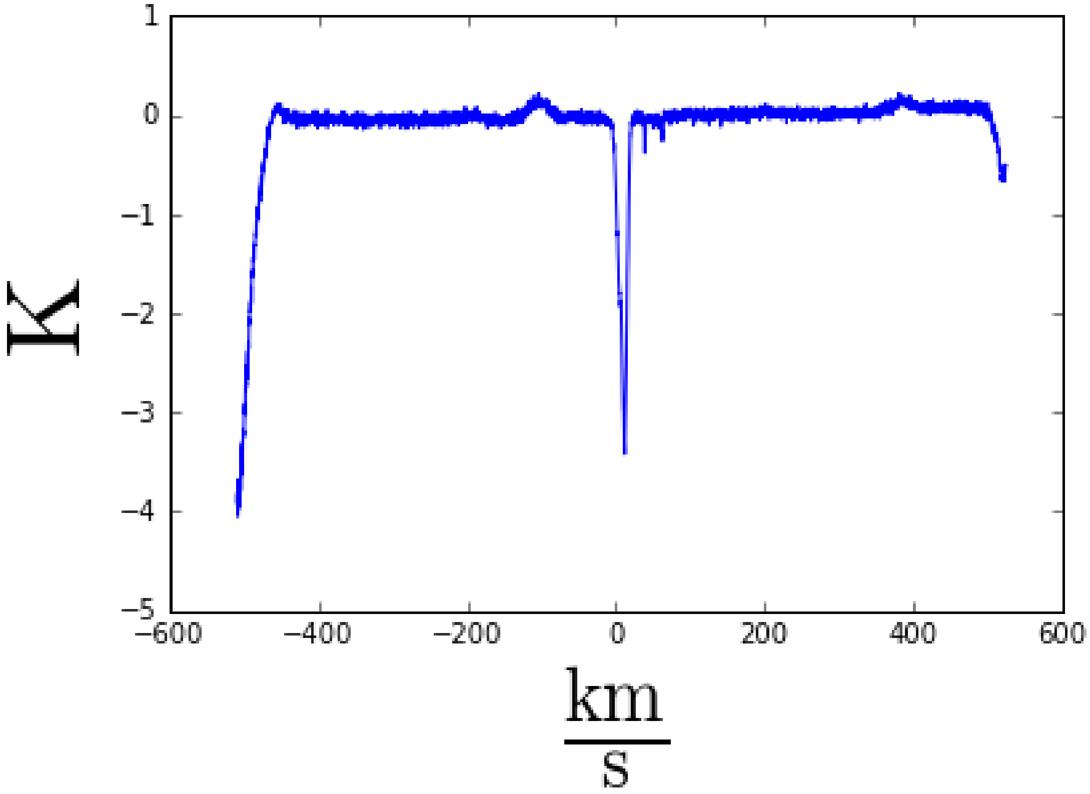
```
In [97]: xarr_u

Out[97]:
```

$$[524.88543,\ 524.63285,\ 524.38026,\ \ldots, -508.95852,\ \ -509.21111,\ \ -509.4637]\ \mathrm{\frac{km}{s}}$$

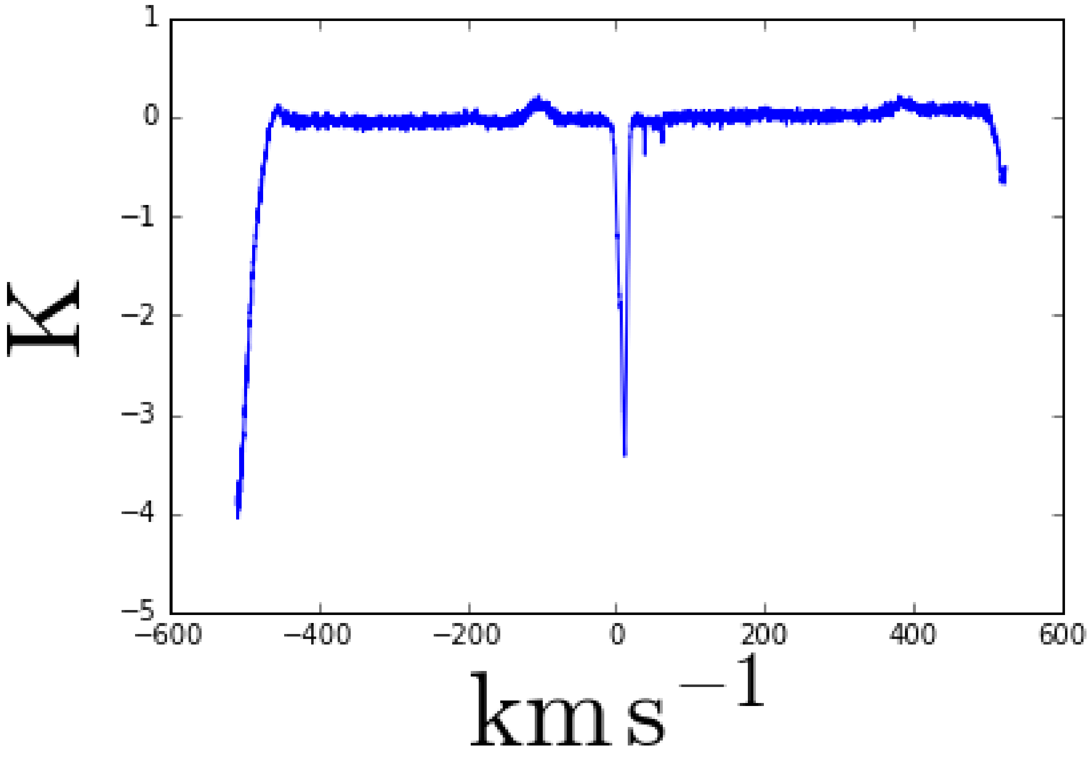
```
In [98]: data

Out[98]: array([-0.50829212, -0.49870891, -0.52269076, ..., -3.81450779,
                -3.85548328, -3.90748133])

In [99]: pl.plot(xarr_u, data)

Out[99]: [<matplotlib.lines.Line2D at 0x10e08e128>]
```



```
In [101]: pl.plot(xarr_u, data)
          pl.xlabel(xarr_u.unit.to_string(format='latex'), fontsize=40)
          pl.ylabel(u.Unit(hdr['BUNIT']).to_string(format='latex'), fontsize=40)

Out[101]: <matplotlib.text.Text at 0x10e889ac8>
```



```
In [103]: pl.plot(xarr_u, data)
          pl.xlabel(xarr_u.unit.to_string(format='latex_inline'), fontsize=40)
          pl.ylabel(u.Unit(hdr['BUNIT']).to_string(format='latex'), fontsize=40)

Out[103]: <matplotlib.text.Text at 0x10e755828>
```



```
In [105]: xarr_u.to?

In [106]: from astropy.wcs import WCS

In [107]: mywcs = WCS(hdr)

In [108]: mywcs

Out[108]: WCS Keywords

          Number of WCS axes: 1
          CTYPE : 'VRAD'
          CRVAL : 7584.5751
          CRPIX : 2049.0
          PC1_1  : 1.0
          CDELT : -252.58830999999998
          NAXIS    : 4096 0

In [139]: xarr_pixels = np.arange(hdr['NAXIS1'])
          xarr_pixels_1_indexed = np.arange(1, hdr['NAXIS1']+1)
          # 0 tells you where to start counting:
          # 1 for FITS (1,2,3...)
          # 0 for python, c, etc. (0,1,2,....)
          # 0 or 1 should correspond to the first element of xarr_pixels
          xarr_wcs, = mywcs.wcs_pix2world(xarr_pixels, 0)

In [127]: # CUNIT1, CRVAL1
          mywcs.wcs.cunit[0], mywcs.wcs.crval[0]
```

```
Out[127]: (Unit("m / s"), 7584.5751)

In [128]: xarr_wcs_u = xarr_wcs * mywcs.wcs.cunit[0]

In [129]: xarr_wcs_u

Out[129]:
```

$$[524885.43,\ 524632.85,\ 524380.26,\ \ldots, -508958.52,\ \ -509211.11,\ \ -509463.7]\ \mathrm{\frac{m}{s}}$$

```
In [124]: x, = (1,)

In [125]: x

Out[125]: 1

In [130]: x,y,z = (1,2,3)

In [131]: xarr_wcs_u == xarr_u

Out[131]: array([ True, False,  True, ...,  True, False, False], dtype=bool)

In [132]: xarr_u

Out[132]:
```

$$[524.88543,\ 524.63285,\ 524.38026,\ \ldots, -508.95852,\ \ -509.21111,\ \ -509.4637]\ \mathrm{\frac{km}{s}}$$

```
In [133]: xarr_wcs_u

Out[133]:
```

$$[524885.43,\ 524632.85,\ 524380.26,\ \ldots, -508958.52,\ \ -509211.11,\ \ -509463.7]\ \mathrm{\frac{m}{s}}$$

```
In [136]: xarr_u[1], xarr_wcs_u[1]

Out[136]: (<Quantity 524.63284567 km / s>, <Quantity 524632.84567 m / s>)

In [137]: xarr_u[1] - xarr_wcs_u[1]

Out[137]:
```

$$1.1368684 \times 10^{-13}\ \mathrm{\frac{km}{s}}$$

```
In [138]: xarr_wcs_u[1] - xarr_u[1]

Out[138]:
```

$$-1.1641532 \times 10^{-10}\ \mathrm{\frac{m}{s}}$$

```
In [140]: xarr_wcs_u[0] - xarr_u[0]

Out[140]:
```

$$0\ \mathrm{\frac{m}{s}}$$

```
In [143]: np.isclose?

In [144]: np.abs(xarr_wcs_u-xarr_u) <= (1e-8*u.m/u.s) + 1e-5*np.abs(xarr_u)
```

```
Out[144]: array([ True,  True,  True, ...,  True,  True,  True], dtype=bool)

In [145]: np.isclose(xarr_wcs_u, xarr_u, atol=1e-8*u.m/u.s)

Out[145]: array([ True,  True,  True, ...,  True,  True,  True], dtype=bool)

In [150]: all(np.isclose(xarr_wcs_u, xarr_u, atol=1e-8*u.m/u.s))

Out[150]: True

In [155]: np.min((xarr_wcs_u - xarr_u))

Out[155]:
```

$$-1.1641532 \times 10^{-10}\ \frac{\mathrm{m}}{\mathrm{s}}$$

```
In [156]: all(xarr_wcs_u == xarr_u)

Out[156]: False

In [157]: np.all(xarr_wcs_u == xarr_u)

Out[157]: False

In [158]: any(xarr_wcs_u == xarr_u)

Out[158]: True

In [161]: not_equal = xarr_wcs_u != xarr_u

In [163]: not_equal

Out[163]: array([ True, False,  True, ...,  True, False, False], dtype=bool)

In [167]: np.where(not_equal)

Out[167]: (array([   0,    2,    5, ..., 4089, 4090, 4093]),)

In [168]: np.arange(4096)[not_equal]

Out[168]: array([   0,    2,    5, ..., 4089, 4090, 4093])

In [166]: len(not_equal), np.count_nonzero(not_equal)

Out[166]: (4096, 2444)

In [162]: xarr_u[not_equal]

Out[162]:
```

$$[524.88543,\ 524.38026,\ 523.62249,\ \ldots, -507.94817,\ -508.20075,\ -508.95852]\ \frac{\mathrm{km}}{\mathrm{s}}$$

```
In [169]: xarr_u[np.where(not_equal)]

Out[169]:
```

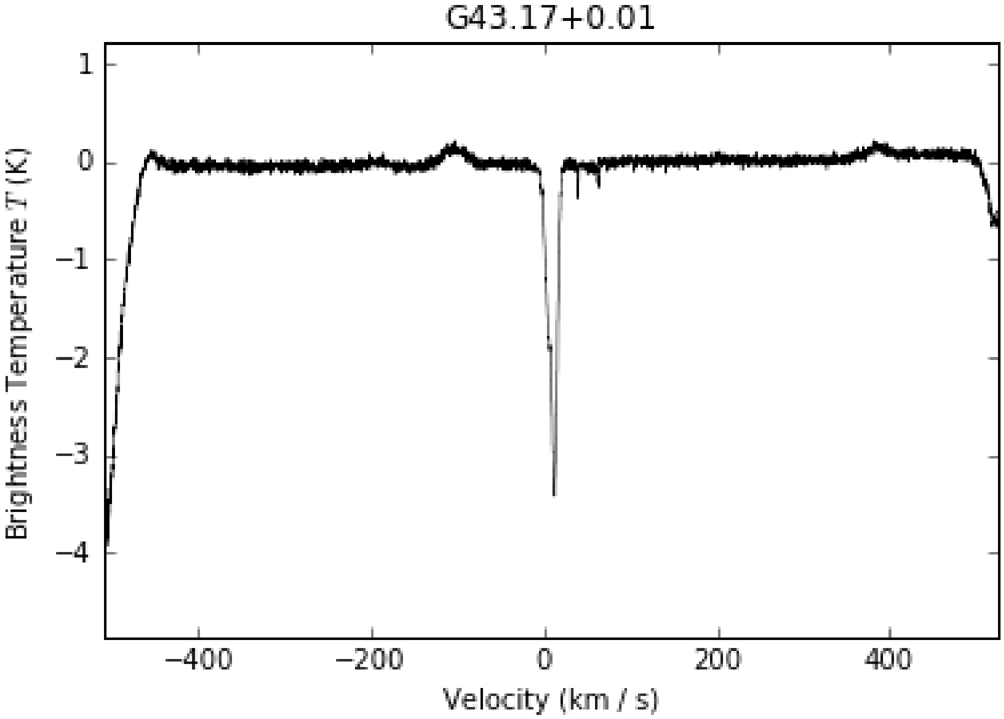
$$[524.88543,\ 524.38026,\ 523.62249,\ \ldots, -507.94817,\ -508.20075,\ -508.95852]\ \frac{\mathrm{km}}{\mathrm{s}}$$

```
In [170]: xarr_u[[1,5,7]]
```

```
Out[170]:
```

$$[524.63285,\ 523.62249,\ 523.11732]\ \mathrm{\frac{km}{s}}$$

```
In [173]: not_equal.shape, xarr_u[not_equal].shape, np.where(not_equal)[0].shape, xarr_u[np.where(not_e

Out[173]: ((4096,), (2444,), (2444,), (2444,))

In [177]: from specutils.io import fits

In [179]: !which pip

/Users/adam/anaconda/envs/esopython2016/bin/pip

In [ ]: # pip install https://github.com/astropy/astroquery/archive/master.zip

In [176]: %%bash
          pip install specutils
          pip install pyspeckit

Requirement already satisfied (use --upgrade to upgrade): specutils in /Users/adam/anaconda/envs/esopyt
Requirement already satisfied (use --upgrade to upgrade): astropy in /Users/adam/anaconda/envs/esopytho
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.6.0 in /Users/adam/anaconda/envs/eso
Collecting pyspeckit
  Downloading pyspeckit-0.1.18.1.tar.gz (30.9MB)
Requirement already satisfied (use --upgrade to upgrade): astropy in /Users/adam/anaconda/envs/esopytho
Requirement already satisfied (use --upgrade to upgrade): numpy in /Users/adam/anaconda/envs/esopython2
Requirement already satisfied (use --upgrade to upgrade): matplotlib>=1.4 in /Users/adam/anaconda/envs/
Requirement already satisfied (use --upgrade to upgrade): python-dateutil in /Users/adam/anaconda/envs/
Requirement already satisfied (use --upgrade to upgrade): pytz in /Users/adam/anaconda/envs/esopython20
Requirement already satisfied (use --upgrade to upgrade): cycler in /Users/adam/anaconda/envs/esopython
Requirement already satisfied (use --upgrade to upgrade): pyparsing!=2.0.4,>=1.5.6 in /Users/adam/anaco
Requirement already satisfied (use --upgrade to upgrade): six>=1.5 in /Users/adam/anaconda/envs/esopyth
Building wheels for collected packages: pyspeckit
  Running setup.py bdist_wheel for pyspeckit: started
  Running setup.py bdist_wheel for pyspeckit: finished with status 'done'
  Stored in directory: /Users/adam/Library/Caches/pip/wheels/71/d8/ec/e6702463cbb41827afeb16f9fcdf0df29
Successfully built pyspeckit
Installing collected packages: pyspeckit
Successfully installed pyspeckit-0.1.18.1

In [180]: from specutils.io import fits

In [181]: spec = fits.read_fits_spectrum1d('gbt_1d.fits')

In [186]: spec

Out[186]: Spectrum1D([-0.50829212, -0.49870891, -0.52269076, ..., -3.81450779,
                     -3.85548328, -3.90748133])

In [194]: import pyspeckit

In [195]: sp = pyspeckit.Spectrum('gbt_1d.fits')

In [196]: sp.plotter()
```



2 Interpolation

Functions for interpolation:

- `np.interp1d`
- `scipy.interpolate`
- `pyspeckit.interpolation`

3 Convolution

- `np.convolve`
- `scipy.ndimage.convolve`
- `astropy.convolution`
- `pyspeckit.smooth`

4 Exercises

1. Load the `gbt_1d.fits` spectrum and plot it
2. Interpolate the spectrum onto a new finer grid from -50 to 50 km/s with 1000 channels
3. Smooth the spectrum by 8 km/s, then interpolate it onto a coarser grid from -400 to 400 km/s with 200 channels

This notebook: https://goo.gl/1AM21P

```
In [199]: np.linspace(-50, 50, 101)
```

```
Out[199]: array([-50., -49., -48., -47., -46., -45., -44., -43., -42., -41., -40.,
                 -39., -38., -37., -36., -35., -34., -33., -32., -31., -30., -29.,
                 -28., -27., -26., -25., -24., -23., -22., -21., -20., -19., -18.,
                 -17., -16., -15., -14., -13., -12., -11., -10.,  -9.,  -8.,  -7.,
                  -6.,  -5.,  -4.,  -3.,  -2.,  -1.,   0.,   1.,   2.,   3.,   4.,
                   5.,   6.,   7.,   8.,   9.,  10.,  11.,  12.,  13.,  14.,  15.,
                  16.,  17.,  18.,  19.,  20.,  21.,  22.,  23.,  24.,  25.,  26.,
                  27.,  28.,  29.,  30.,  31.,  32.,  33.,  34.,  35.,  36.,  37.,
                  38.,  39.,  40.,  41.,  42.,  43.,  44.,  45.,  46.,  47.,  48.,
                  49.,  50.])
```