SpectraModels_Day1_Part2

March 21, 2016

1 Fitting data to models

1. Build a model
2. Create a "fitness function", i.e. something that returns a scalar "distance" between the model and the data
3. Apply an "optimizer" to get the best-fit parameters

```
In [1]: def gaussian_model(xaxis, amplitude, offset, width):
            amplitude = u.Quantity(amplitude, u.K)
            offset = u.Quantity(offset, u.km/u.s)
            width = u.Quantity(width, u.km/u.s)

            return amplitude*np.exp(-(xaxis-offset)**2/(2.*width**2))
```

```
In [2]: from specutils.io import fits
        spec = fits.read_fits_spectrum1d('gbt_1d.fits')
```

```
In [3]: from astropy import units as u
```

```
In [4]: %%bash
        which conda
```

```
/Users/adam/anaconda/envs/esopython2016/bin/conda
```

```
In [5]: import specutils
        import numpy
        import astropy
        specutils.__version__, astropy.__version__, numpy.__version__, astropy.__path__
```

```
Out[5]: ('0.2.dev0',
         '1.1.1',
         '1.10.4',
         ['/Users/adam/anaconda/envs/esopython2016/lib/python3.5/site-packages/astropy'])
```

```
In [6]: spec.velocity
```

```
Out[6]:
```

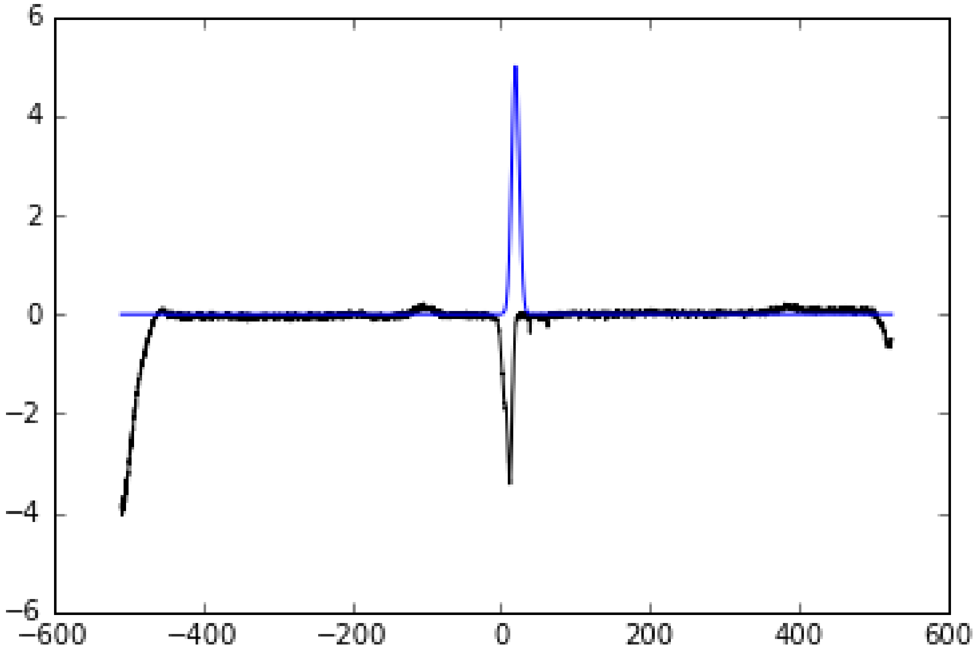
$$[524.88543,\ 524.63285,\ 524.38026,\ \ldots, -508.95852,\ -509.21111,\ -509.4637]\ \frac{\mathrm{km}}{\mathrm{s}}$$

```
In [7]: model = gaussian_model(spec.velocity, amplitude=5*u.K, offset=20*u.km/u.s, width=5*u.km/u.s)
```

```
In [8]: %matplotlib inline
        import pylab as pl
```

```
In [9]: pl.plot(spec.velocity, spec.flux, 'k-')
        pl.plot(spec.velocity, model)
```

```
Out[9]: [<matplotlib.lines.Line2D at 0x10e852940>]
```



```
In [10]: def cost_function(params):
             return ((spec.flux*u.K-gaussian_model(spec.velocity, *params))**2).sum().value
```

```
In [11]: from scipy.optimize import curve_fit, minimize
```

```
In [12]: result = minimize(cost_function, (-5, 20, 20))
```

```
In [13]: result
```

```
Out[13]:       fun: 874.080855382803
          hess_inv: array([[ 0.02026627, -0.00628146,  0.02600413],
                [-0.00628146,  0.09292427, -0.02506448],
                [ 0.02600413, -0.02506448,  0.09152247]])
               jac: array([ 0.,  0.,  0.])
           message: 'Optimization terminated successfully.'
              nfev: 115
               nit: 17
              njev: 23
            status: 0
           success: True
                 x: array([ -2.98664862,  10.56106783,   5.22460489])
```

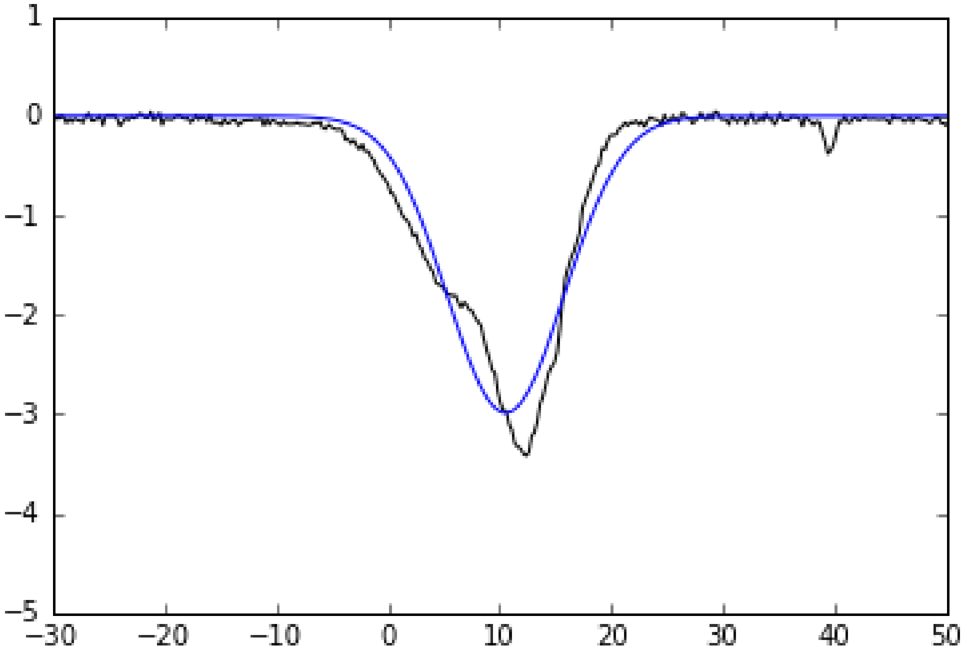
```
In [14]: best_fit_parameters = result.x
```

```
In [15]: best_fit_model = gaussian_model(spec.velocity, *best_fit_parameters)
         best_fit_model
```

```
Out[15]:
```

$$[-0,\ -0,\ -0,\ \ldots, -0,\ -0,\ -0]\ \mathrm{K}$$

```
In [16]: pl.plot(spec.velocity, spec.flux, 'k-')
         pl.plot(spec.velocity, best_fit_model)
         pl.xlim(-30,50)

Out[16]: (-30, 50)
```



1.1 Fitting Tools

1. `scipy.optimize.curve_fit`: simpler fitter, lets you skip the cost-fitting section
2. `astropy.models`
3. `pyspeckit.specfit`

1.1.1 scipy.optimize.curve_fit

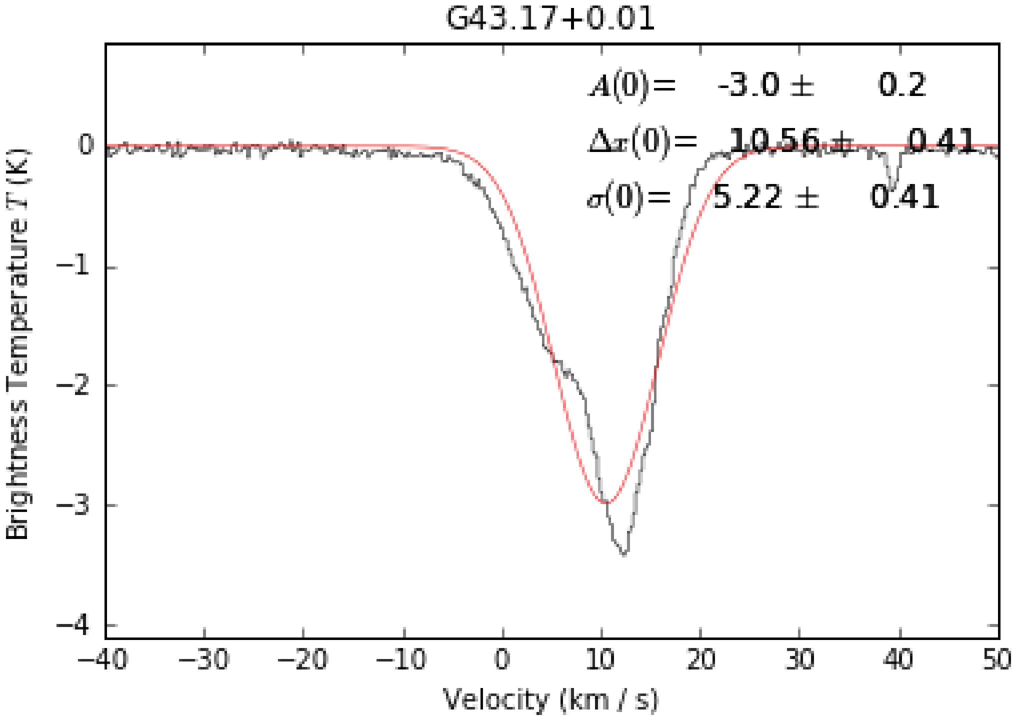
```
In [17]: from scipy.optimize import curve_fit

In [18]: # curve_fit does not play well with units
         #result_curve_fit = curve_fit(gaussian_model, spec.velocity, spec.flux*u.K, p0=(-5, 20, 20))
```

1.1.2 astropy.modeling

1.1.3 pyspeckit.specfit

```
In [19]: import pyspeckit
         sp = pyspeckit.Spectrum('gbt_1d.fits')
```

```
In [20]: sp.plotter(xmin=-40, xmax=50)
         sp.specfit(guesses=(-5, 20, 20))
```



2 Exercise

1. Get a better fit to the data (create a better model & fit it)

- try using different optimizers in scipy.optimize

```
In [ ]:
```