Computational Physics Project / Modeling a point spread function

This project involves fitting a model to observational data, to charactering the resolution of an image from a telescope.

Telescope images have a resolution that changes across their field of view, because of their optics and because of the atmosphere. Precise measurements of these images require first determining these *point spread functions* as a function of image position. This point spread function can be thought of as a Green's Function — i.e. the response of the system to a delta-function input.

Here you will fit a simple model to the point spread function of images.

You can retrieve images at the SDSS imaging data site. You can use whatever fields you want, but here are some examples with "run," "camcol," and "field" values:

```
94 3 200
752 4 400
1336 6 61
```

I'll ask you to look at all the bandpasses for whatever fields you choose.

You will need the FITS format files. These files can be read in using the `fitsio` Python package, which is `pip`-installable. You will be able to read the images into a NumPy array with:

```
image = fitsio.read(filename, ext=0)
```

In order to look at the image within Python, you will need to use the `imshow` utility in `matplotlib`. Note that these images already are calibrated and background-subtracted; they don't come off the telescope looking this clean!

Some of the sources in the image are stars, and some are galaxies. Stars are "unresolved," in the sense that how big the image appears is set purely by the point spread function. Their intrinsic angular width is extremely small. Galaxies are generally somewhat bigger on the sky than stars — they are often "resolved." The result is that in order to determine the point spread function, you should limit yourself to stars.

For any given field, you may consult the SQL query server on SkyServer to get a list of identified stars in it, using the following query:

```
SELECT colc_u,colc_g,colc_r,colc_i,colc_z,
       rowc_u,rowc_g,rowc_r,rowc_i,rowc_z
FROM photoObj
WHERE run = 94 AND camcol = 2 AND field = 200 AND type = 6
```

This will return the row and column positions (defined with 0 at the first pixel edge, not the center). Use the "CSV" or "FITS" output formats for ease of reading the results into Python.

- Download the image and the star positions for at least one field and bandpass. The star positions over a section of the image, to verify you are interpreting the row and column values correctly.
- Write code to: identify isolated stars (stars without any other nearby star within 20 pixels); cut out 31×31 pixels around each; shift each array so that the stellar center is the center of a pixel using a high order 2-dimensional interpolation; normalize each image using the flux with 15 pixels radius of the center.
- Apply principal components analysis (PCA; Landau Chapter 13.7) on these images. Do not forget to subtract off the mean before applying PCA. Perform an analysis of the residual error as a function of number of components retained.
- Probably 3–4 components are okay. Write a routine to fit a low-order (i.e. up to 2 or so periods across the image) sum of sine and cosine functions to each PCA coefficient in the row and column directions, using least squares on the values of the coefficient for each star. Use SVD for this solution for stability. E.g. your model for coefficient i from the PCA will look like:
$$a_i = \sum_j A_j \sin(2\pi j x/n_x) + B_j \cos(2\pi j y/n_y) \quad (1)$$
in terms of the linear model parameters A_j and B_j.
- The PSF at any location can be recovered from your fit to the a_i. Test the goodness-of-fit by subtracting the resulting model from the stars.
- Run your code on all the bandpasses for several images.