Astronomical Observing Techniques 2016: Exercises on Fourier Transforms (Due on 7 March 2016 at 11:15)

February 29, 2016

1 Diffraction-limited Point-Spread Function

Write a python program that can take a circular aperture function a and calculate the diffraction-limited Point-Spread Function (PSF) as the absolute value squared of the Fourier transform of the aperture, which is defined as being 1 where light is transmitted and 0 everywhere else. Submit the program as well as the result from one example calculation.
Hints:

- If you use an array of e.g. 1024 by 1024 pixels, make sure that the aperture does not extend beyond half the size of the array, i.e. stay within the pixel range 256 to 768.
- Use `numpy.fft.fft2` for 2-D forward Fourier transforms and `numpy.fft.ifft2` for the corresponding inverse Fourier transform.
- The Fourier transform in `numpy` defines the zero frequency to be at `[0,0]` instead of the center of the array. Use `numpy.fft.ifftshift` before a transform (forward or backward) and use `numpy.fft.fftshift` after each transform.

2 Speckles

Expand your program such that your aperture function can have a complex value, i.e. $a \cdot e^{i\phi}$. Download the file `phase.fits` containing ϕ (simulating the effect of the Earth's atmosphere). The phase screen is 512 by 512 pixels.

1. Simulate the images of a binary star by making a perfect image of two point sources and convolve it with the perfect PSF from exercise 1. Choose the distance between the two stars such that they are very close but still clearly visible as two separate stars.
2. Calculate the PSF that would correspond to the atmospheric aberrations ϕ.
3. Calculate the observed image of the binary by convolving the two point sources with the PSF. The convolution is best implemented via Fourier transforms.

Submit the program and the results.
Hints:

- Use `astropy.io.fits.getdata` to read the `phase.fits` file
- If you created a real-valued aperture `real_aperture`, then you can create a complex-valued version with `complex_aperture = real_aperture.astype('complex').copy()`.
- Use `numpy.exp` for the complex exponential. Use `1j` for i.