Computational Physics Project

Evan Arch, Daniel Bateman, & Nana Ama Darpaah

November 20, 2024

1 Prep Work

1.1 Confirmation of Numpy's J_1 via Simpson's Rule

The 1st order Bessel function of the first kind, $J_1(x)$ can be calculated as

$$J_1 = \frac{1}{\pi} \int_0^{\pi} d\theta \cos(\theta)(\theta - x \sin(\theta)) \tag{1}$$

which has no closed form analytic solution. Thus, $J_1(x)$ is generally computed numerically. Scipy, however, uses an approximate analytic form J_1 rather than numeric computation. We verified Scipy's J_1 using our own implementation which computes the integral numerically for each x using Simpson's rule. A comparison of the results of the two method is shown in figure 1.

1.2 Analytic Point Spread Function

The point spread function (PSF) is given by

$$\text{PSF}(x_f, y_f) = |\text{FT}[P(x,y)] \left(k_x = \frac{x_f}{\lambda f}, k_y = \frac{y_f}{\lambda f} \right) |^2 \tag{2}$$

where FT denotes the Fourier transform, λ is the wavelength, f is the focal length, and $P(x,y)$ is the complex pupil function.

Neglecting phase, we can write the pupil function $P_r(r)$ for a Cassegrain telescope in polar coordinates as

$$P_r(r) = \begin{cases} 1 & R_0 \leq r \leq R_1 \\ 0 & r < R_0 \cup r > R_1 \end{cases} \tag{3}$$

where R_0 and R_1 are the inner and outer diameter of the aperture respectively. In polar coordinates, the Fourier transform of a radially symmetric function is given by

$$F(\rho) = \int_0^{\infty} P_r(r) r J_0(\rho r) \tag{4}$$

where ρ is the radial coordinate in the transformed space and J_0 is the 0th order Bessel function of the first kind. Evaluating this integral with Wolfram Mathematica gives

$$F(\rho) = \frac{1}{\rho} \left(R_1 J_1(R_1 \rho) - R_0 J_1(R_0 \rho) \right) \tag{5}$$

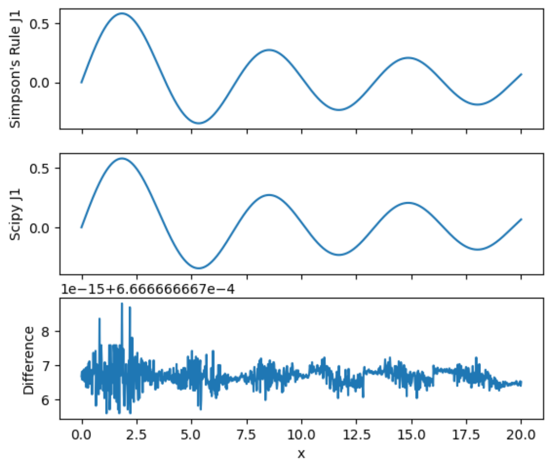

Figure 1: Comparison of Scipy's $J_1(x)$ with the numeric solution to eq 1 found using Simpson's rule.

where J_1 is the first order Bessel function of the first kind. Converting back to Cartesian coordinates with $\rho = \sqrt{k_x^2 + k_y^2}$, and using eq 2, we find the PSF of a Cassegrain telescope to be

$$\mathrm{PSF}(x_f, y_f) = \frac{1}{k_x^2 + k_y^2} * \left(R_1 J_1(R_1\sqrt{k_x^2 + k_y^2}) - R_0 J_1(R_0\sqrt{k_x^2 + k_y^2})\right) \tag{6}$$

where $k_x = \frac{x_f}{\lambda f}$ and $k_y = \frac{y_f}{\lambda f}$ a plot of which is shown in figure 2

1.3 Gaussian Random Fields

To generate a Gaussian random field, we must first generate an even random field $f(x, y)$. Here, even implies that $f(x, y) = f(-x, -y)$. This means that if we wish generate a $N \times N$ random field, half of the values in said field are determined by the other half. If we have a 2D python array indexed by x and y, we treat $x = -1$ as being equivalent to $x = N - 1$. Luckily, python implements this by default. Thus, all we must do is carefully arrange our $\frac{N^2}{2}$ random numbers and their duplicates in $N \times N$ grid such that the result is even. The correct arrangement is shown in figure 3. An array of random numbers generated using this method is shown in figure 4. Taking the Fourier transform of the random array gives a real result, implying that the random array is indeed even.

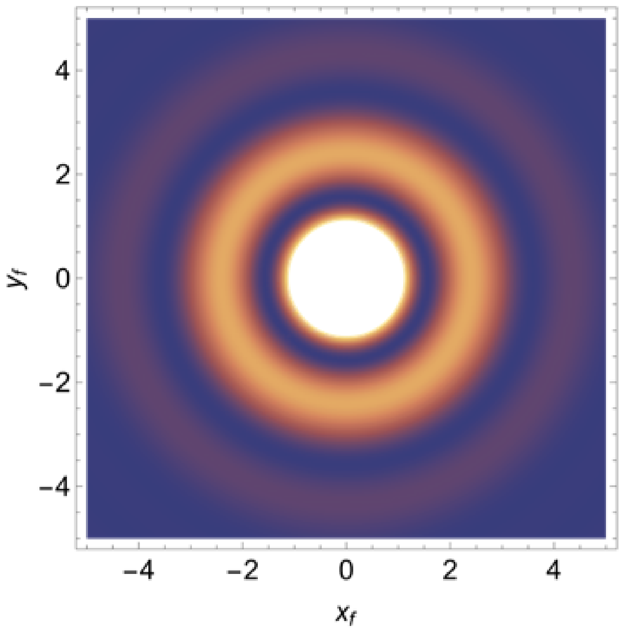

Figure 2: Analytic point spread function(PSF) of a Cassegrain aperture with $R_0 = 1$, $R_1 = 2$, and $\lambda = f = 1$ in arbitrary units.

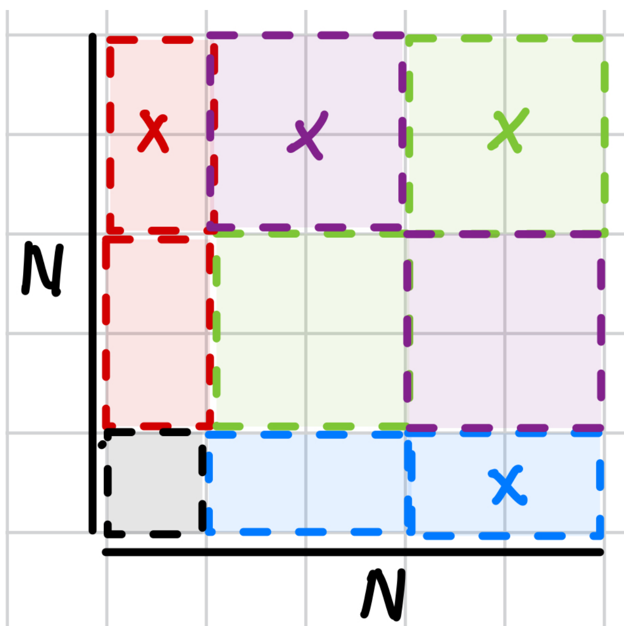

Figure 3: Arrangement of random values in $N \times N$ array to create an even function. Each square on the grid represents a single element in the array. Every square in a colored region without an x is uniquely determined. Those with an x are duplicates of the other region of the same color that have been rotated 180 degrees. The origin is located in the lower left and is highlighted in black.

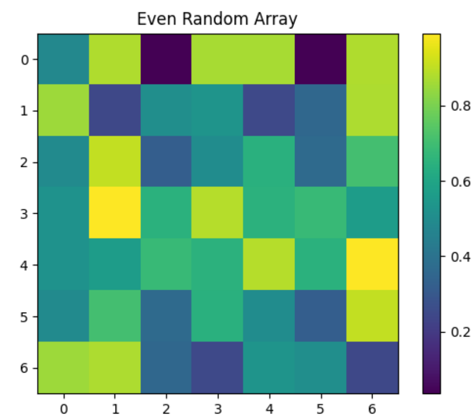

Figure 4: Graphical representation of an even random array generated using the method shown in figure 3. The origin is located at the top right corner and indices labeled 6, 5, 4 also correspond to indices -1, -2, -3.