ESC407 Lab 1

Maggie Wang

October 2, 2023

1. Diffraction

   (a) The procedure to numerically compute the Bessel function $J_m(x)$ using Simpson's rule is outlined below. Fig. 1 shows the results for Bessel functions J_0, J_1, and J_2 as a function of x from $x = 0$ to $x = 2$.

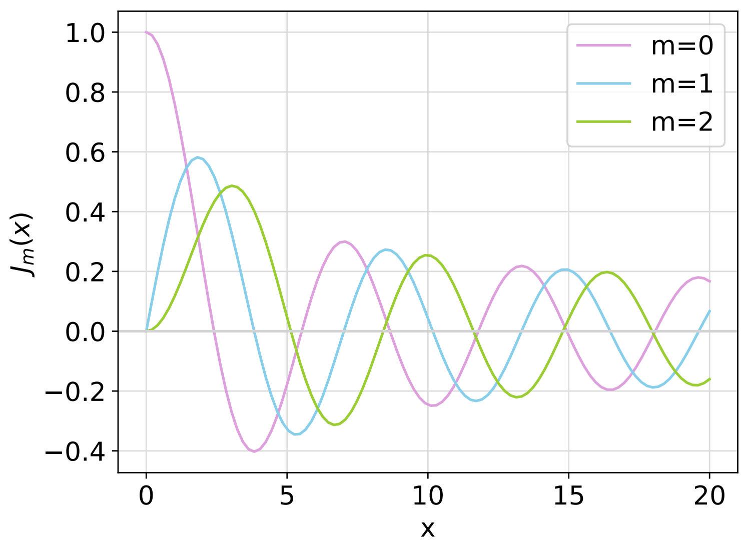

Figure 1: Bessel functions $J_m(x)$ computed using Simpson's rule

Pseudocode

Simpson's rule

- Define function `simpson(N, a, b, f)`, which computes Simpson's rule on some function `f` over `[a, b]` with `N` integration points
  - Compute step size `h = (b-a)/N`
  - Initialize `sum_f` with value `f(a) + f(b)`
  - For each integer `k` from 1 to `N`:
    - If `k` is even, add `2f(a+kh)` to `sum_f`
    - If `k` is odd, add `4f(a+kh)` to `sum_f`
  - Return `h*sum_f/3`

Bessel function

- Define function `J_integrand(theta, m, x)`
  - Return `cos(m theta - x sin(theta))`

- Define function `J(m, x)`
  - Call `simpson` with arguments `N`=1000, `a`=0, `b`=π, and `f=J_integrand(theta, m=m, x=x)`

Plotting

- Initialize and set the number of plotted points `nvals=100`
- For each value of `m`:
  - Initialize `x_arr` with `nvals` equally spaced points between 0 and 20
  - Compute `y_arr` as `J(m, x)` for each `x` in `x_arr`
  - Plot `y_arr` vs `x_arr`

(b) The calculation using Simpson's rule closely matches results using `scipy.special.jv`, as seen in Fig. 2. The pseudocode to generate Fig. 2 is outlined below.

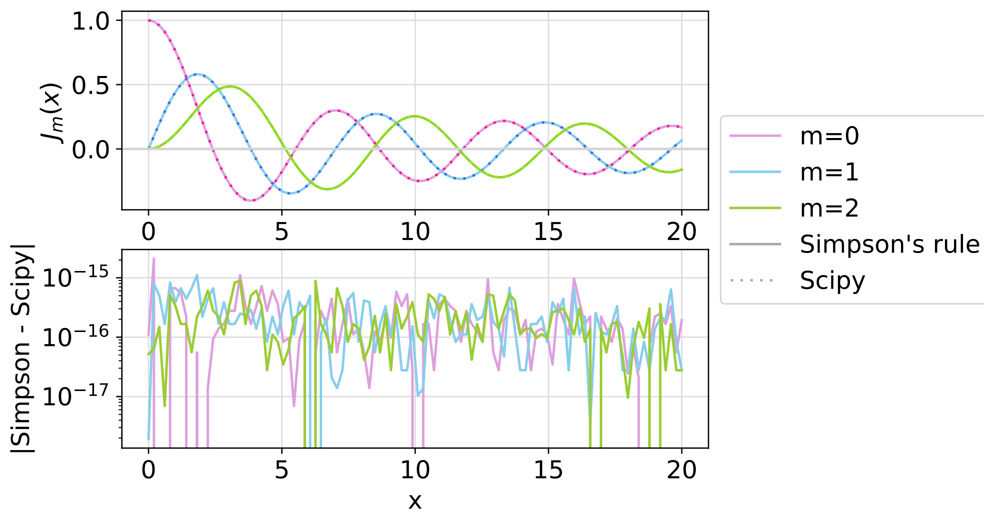

Figure 2: Bessel functions $J_m(x)$ computed using Simpson's rule and `scipy.special.jv`

Pseudocode

- Initialize and set `nvals=100`
- For each value of `m` of interest:
  - Initialize `x_arr` with `nvals` equally spaced points between 0 and 20
  - Computearrays `J_simpson` as `J(m, x)` and Compute `J_scipy` using Scipy functions for each `x` in `x_arr`
  - Plot `J_simpson` and `J_scipy` vs `x_arr`

(c) Figure 3 shows a plot of the intensity of the diffraction pattern from a point light source with λ = 500 nm in the focal plane of a telescope, with Bessel functions computed using Simpson's rule. The pseudocode for the program which generates the plot is outlined below.

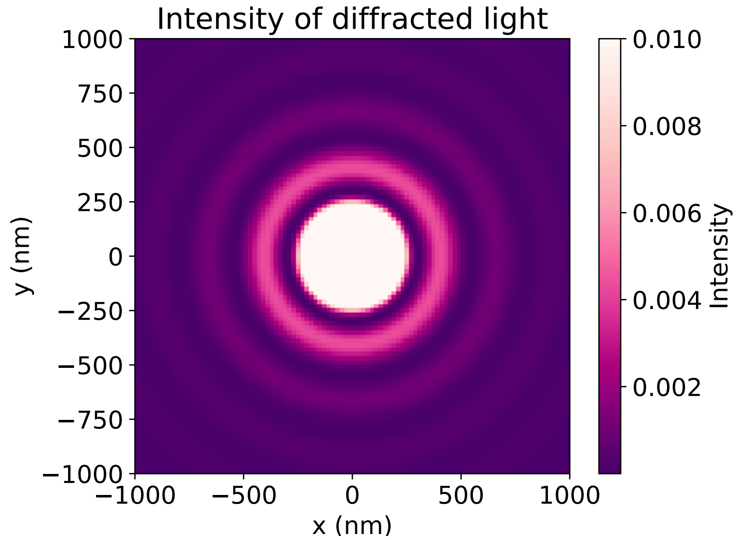

Figure 3: Intensity of diffracted light from 500 nm point source over a radius of 1 um

Pseudocode

- Define function `I(r, wavelength)`
  - Return $\frac{J(2\pi r/\text{wavelength})}{(2\pi r/\text{wavelength})}^2$
- Initialize and set `num_grid_points=100`, `wavelength=500` nm, radius `R=1000` nm
- Initialize arrays `x_arr` and `y_arr` with `nvals` evenly spaced values from `-R` to `R`
- Compute a 2D array `r_mtx` with entries $x^2 + y^2$, for `x` in `x_arr` and `y` in `y_arr`, representing the radial distance from the beam centre
- Compute a 2D array `intensity_mtx` with entries `I(r, wavelength)` for each `r` in `r_matrix`

2. Trapezoidal and Simpson's rules for integration

   (a) Table 1 summarizes the value of the Dawson function at $x = 4$ computed using Simpson's and trapezoidal rules with 8 integration slices, and `scipy.special.dawsn`.

Simpson's rule	Trapezoidal rule	Scipy
0.1826909645971217	0.26224782053479523	0.1293480012360051

Table 1: Value of Dawson function at $x = 4$ computed using Simpson's and trapezoidal rules with 8 slices, and with `scipy.special.dawsn`

   (b) Table 2 summarizes the number of slices needed to approximate the Dawson function at x=4 with an error $\mathcal{O}(10^{-9})$, and the corresponding run time. *scipy.special.dawsn* was used as the reference value, and the runtime of each method was averaged over 50 calls.

	N	runtime (ms)	value	error
Simpson's rule	1024	0.845	0.12934800196026494	$7.242598465406758 \times 10^{-10}$
Trapezoidal rule	65536	48	12934800371953178	$2.483526689855964 \times 10^{-9}$

Table 2: Number of integration slices (N), corresponding runtime, output, and error required to approximate $D(4)$ to $\mathcal{O}(10^{-9})$, with *scipy.special.dawsn* used as a reference value.

(c) Using $N_2 = 64$ and $N_1 = 32$, the error estimate of D(4) using Simpson's rule is

$$\begin{aligned}\epsilon_2 &= \frac{1}{15}(I_2 - I_1) \\ &= 0.00020578842293380212,\end{aligned}$$

and for trapezoidal rule,

$$\begin{aligned}\epsilon_2 &= \frac{1}{3}(I_2 - I_1) \\ &= 0.0005093137305911358\end{aligned}$$

3. Exploring roundoff error

(a) Fig. 4 shows $p(u) = (1-x)^8$ and $q(u)$, the Taylor expansion of $p(u)$ up to degree 8. The plot of $q(u)$ appears noisier because there are more terms involved which are each affected by roundoff error.

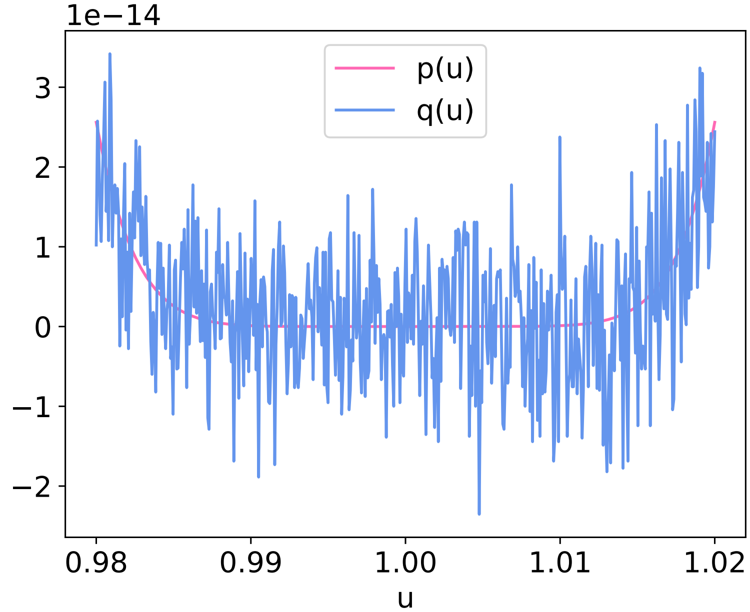

Figure 4: $p(u) = (1-x)^8$ and its Taylor expansion to degree 8, $q(u)$, around u = 1

(b) Fig. 5 a) plots $|p(u) - q(u)|$ around u = 1. Fig. 5 b) shows the histogram associated with Fig. 5 a), which has a standard deviation of 8×10^{-15}. This standard deviation should be the same order of magnitude as an estimate obtained using equation 1, since they are both measures of roundoff error when summing over multiple terms.

The estimate of the error obtained using equation 1 for $p(1) - q(1)$ is the same order of magnitude as the standard deviation above.

$$\begin{aligned}\sigma &= C\sqrt{N}\sqrt{\bar{x^2}} \\ &= 10^{-16}\sqrt{10}\sqrt{1287} \\ &= 1.1 \times 10^{-14}\end{aligned}$$

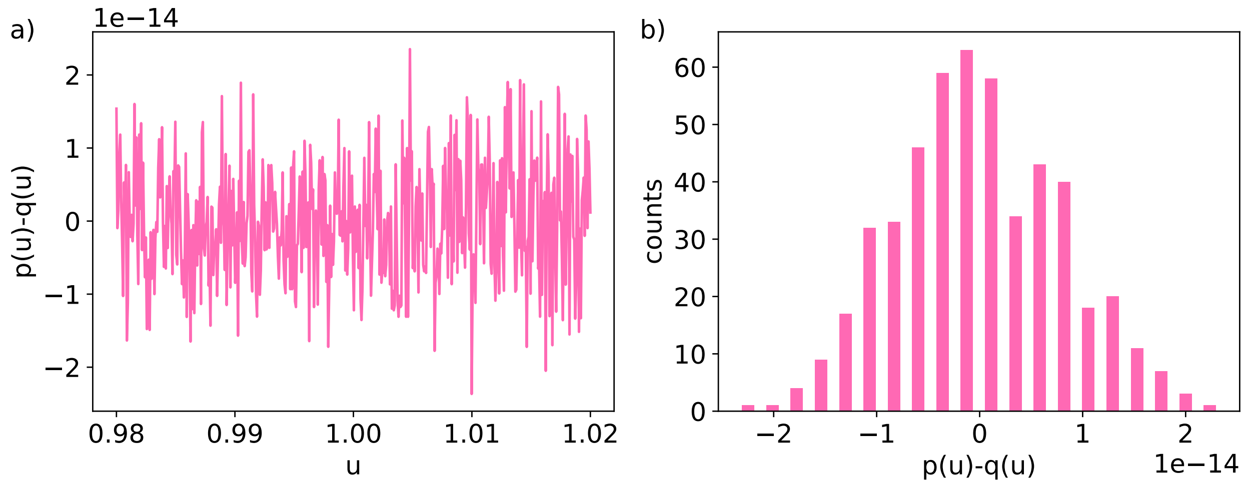

Figure 5: a) $|p(u) - q(u)|$ around u = 1 and b) the corresponding histogram

The operation $p(u) - q(u)$ is a sum over $p(u)$ and each term in $q(u)$, for $u = 1$. Since there are 10 terms in total, $N = 10$. $\bar{x^2}$ is calculated as

$$\begin{aligned}\bar{x^2} &= \frac{(1-1)^{8\cdot 2} + 1^2 + 8^2 + 28^2 + 56^2 + 70^2 + 56^2 + 28^2 + 8^2 + 1^2}{10} \\ &= 1287\end{aligned}$$

(c) Equation 2 estimates the fractional error as

$$\frac{\sigma}{\sum_i x_i} = \frac{C}{\sqrt{N}} \frac{\sqrt{\bar{x^2}}}{\bar{x}}$$

In the calculation of $p(u) - q(u)$, the x_i's alternates sign and as a result, $\bar{x}$ is much smaller than $\sqrt{\bar{x^2}}$ and the fractional error becomes quite large. Using $N = 10$ and the same method to calculate $\bar{x^2}$ as above, for $u = 0.985$,

$$\begin{aligned}\sqrt{barx^2} &= 33.778 \\ \bar{x} &= 2.232 \times 10^{-15} \\ \frac{\sigma}{\sum_i x_i} &= 0.478\end{aligned}$$

For $u = 0.99$,

$$\begin{aligned}\sqrt{barx^2} &\approx 34.465 \\ |\bar{x} &= 1.898 \times 10^{-16} \\ |\frac{\sigma}{\sum_i x_i}| &= 5.741\end{aligned}$$

Between $u = 0.985$ and $u = 0.99$, the estimated fractional error approaches 1.00

Fig. 6 shows that the fractional error $|p(u) - q(u)|/p(u)$ approaches 1.00 when u is between 0.978 and 0.985

(d) Plotting $u^8/(u^4u^4) - 1$ shows the error is on the order of 10^{-16}, which is the same as what equation (4.5), $\sigma = \sqrt{2}Cx = 1.41 \times 10^{-16}$, predicts for $C = 10^{-16}$ and $x = 1$

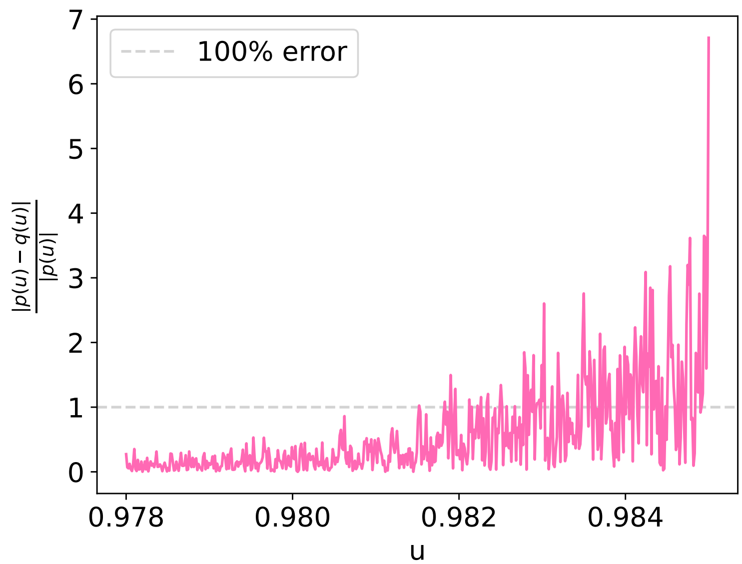

Figure 6: Fractional error $|p(u) - q(u)|/p(u)$ for u between 0.978 and 0.985

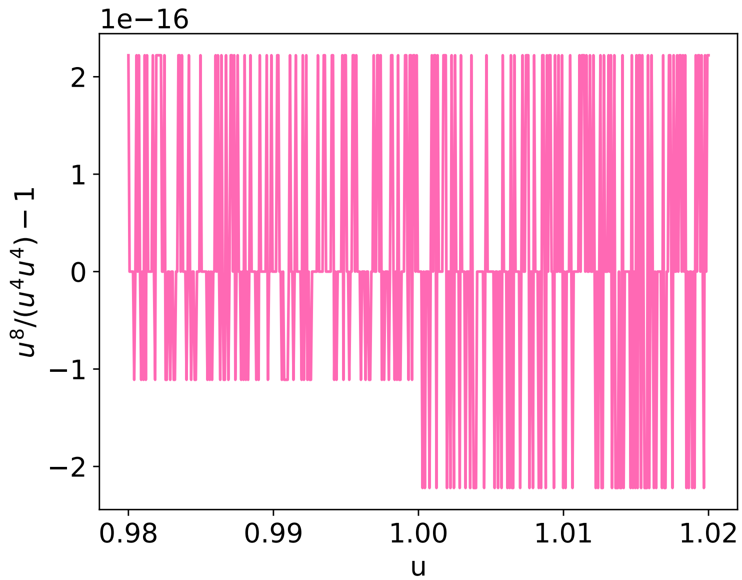

Figure 7: $u^8/(u^4u^4) - 1$ around u=1