Lab assignment #3: Gaussian quadrature, numerical differentiation

In this lab, the Gaussian quadrature method of integration and the central difference method of differentiation were explored. These tools were used to model harmonic oscillations (in Q2) and plot the topography of Hawai'i (in Q3). The codes used in this lab implemented the following Python modules: numpy, scipy, matplotlib, and math. In addition to the main codes containing answers to the three questions posed in the lab manual, please download the following file containing all the original functions created in this lab: Lab03_combined_functions.py

Question 1.

a) For reference, Scipy's erf() method output is 0.9999779095030014. Below, the outputs corresponding outputs for n = 20, 200, 700, and 1000 are displayed for the Trapezoidal, Simpson, and Gaussian quadrature integration methods.
    - N = 20:
        i. Trapezoidal integration method output: 0.9999763603313787
        ii. Simpson integration method output: 0.9999778429104651
        iii. Gaussian integration method output: 0.9999779095029918
    - N = 200
        i. Trapezoidal integration method output: 0.9999778938387949
        ii. Simpson integration method output: 0.999977909495955
        iii. Gaussian integration method output: 0.999977909503001
    - N = 700
        i. Trapezoidal integration method output: 0.9999779082241579
        ii. Simpson integration method output: 0.9999779095029518
        iii. Gaussian integration method output: 0.9999779095030004
    - N = 1000
        i. Trapezoidal integration method output: 0.9999779088763651
        ii. Simpson integration method output: 0.9999779095029889
        iii. Gaussian integration method output: 0.9999779095030014

As expected, the gaussian integration method was closer than trapezoidal and Simpson integration, using a much lower value of N. By N=1000, the Gaussian integration method outputted the exact same result as the scipy. erf() function and the Simpson integration method followed closely behind in terms of accuracy.

B) Plots of the dependence of relative error on N (number of slices) are displayed below on log-log axes for each of the 3 integration methods. Since log(x) is defined only for x>0, a function was created to parse through inputted data points (x, y) and remove any points where $y \leq 0$. Using equation 2, the gaussian integral was evaluated at N= 800 and N= 400, which yielded a difference of 7.7715611723760958e-16. This result makes sense as N gets very large, the relative error is on the magnitude of 1e-16 (Fig 3).

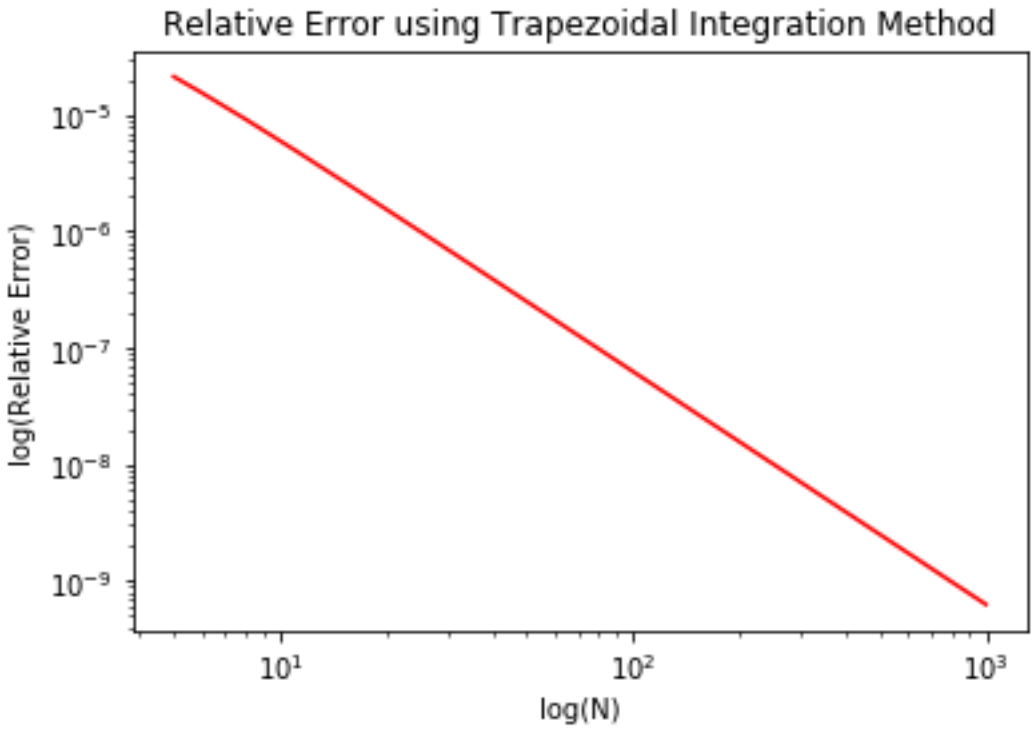

Fig 1. Plot of the linear relationship between log(relative error) and log(N) for the Trapezoidal integration method. It is clear that relative error decreases as N increases.

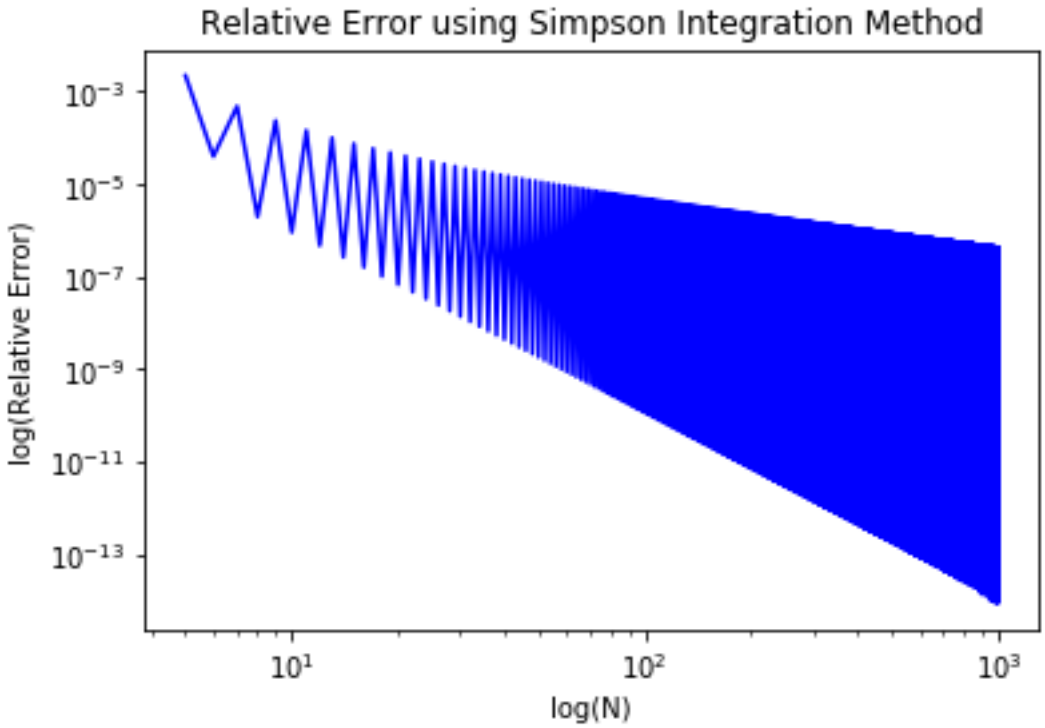

Fig 2. Plot of the complicated oscillatory relationship between log(relative error) and log(N) for the Simpson integration method. The overall trend is that relative error decreases with increasing N.

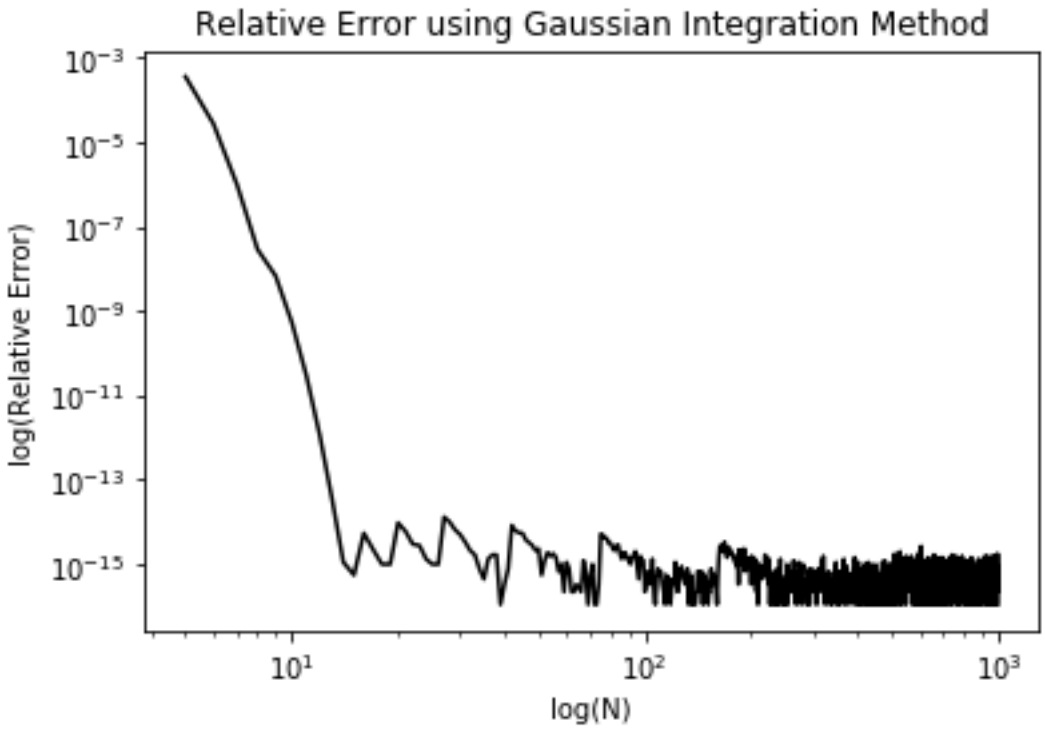

Fig 3. Plot of the complex relationship between log(relative error) and log(N) for the Gaussian integration method. The overall trend is that relative error decreases with increasing N.

While Fig 1 displays a steady decrease in relative error as N increases, Fig 2 and Fig 3 display more complex relationships. With the trapezoidal method, the order of magnitude of relative error decreased from e-3 to e-9 over a range of N = 1000 steps. In the same domain, the Simpson method reached a maximum order of e-3 and a minimum of e-14, which is much lower than what the trapezoidal method was able to achieve. That being said, the oscillatory nature of Fig 2. suggests that users will reap the benefits of the Simpson integration method only for certain values of N; for example, as N approaches 1000, the graph has several peaks with an order of magnitude of about e-6, which is worse than the values attained by the trapezoidal method in this same region. Considering Fig 3., a steep decrease in magnitude from e-3 to e-15 is displayed after only N = 10 steps. Although the shape of the curve is very chaotic thereafter, it remains bounded between an order of magnitude of e-12 and e-16 for all remaining values of N. It is evident that the Gaussian quadrature integration method offers the least relative error for any value of N when compared to Fig 1 & 2.

C) The probability for blowing snow was plotted as a function of temperature (T_a, in ◦C) and is displayed below. In the lab manual, values of u_10 = (6, 8, 10) and t_h = (24, 48, 72) were given; the following plots show all 9 possible combinations of those 2 arrays. The independent variable, T_a, ranges from -20◦C to 30◦C.

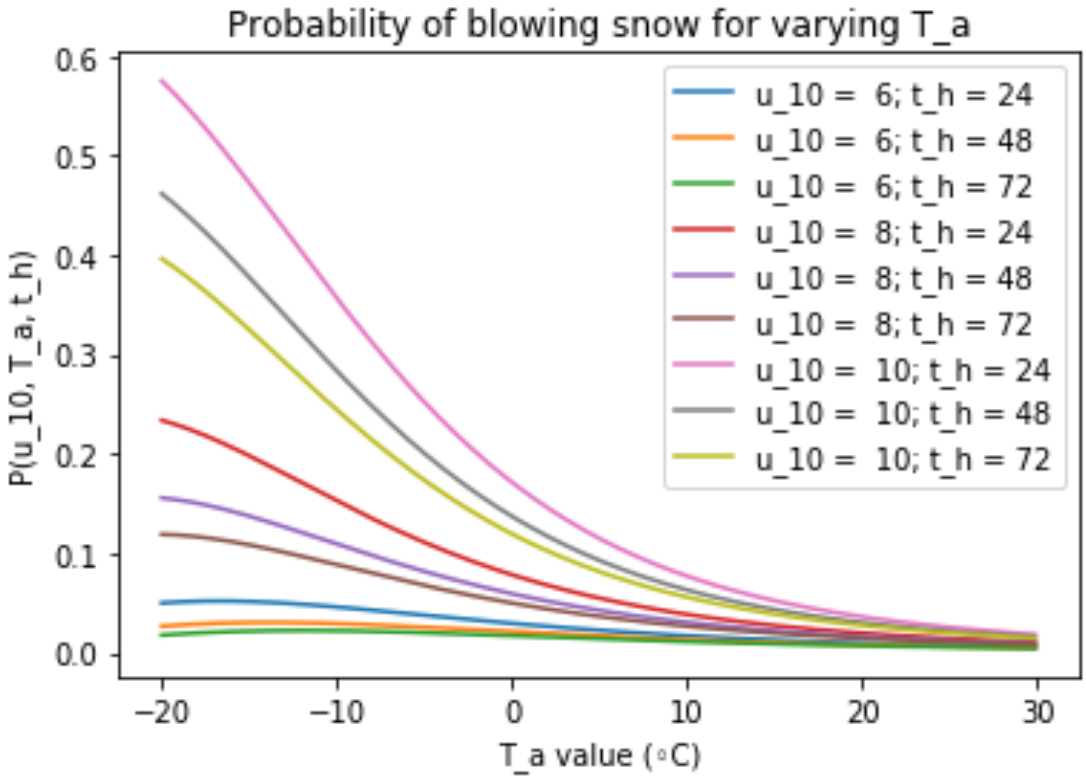

Fig 4. Plot of the probability of blowing snow for different values of u_10 and t_h; it is plotted as a function of T_a

The error estimate for gaussian was outputted to be : -5.55111512313e-16

Each of the curves resembles the right-half of a bell-curve (gaussian curve), reaching a stable probability value of roughly 0.0 as T_a approaches 30◦C. The shape of the curve is modulated by u_10 (average hourly windspeed) and t_h (snow surface age): u_10 will determine the general shape of the bell-curve, while t_h determines the height of its peak (which translates to how probable blowing snow is). This can be observed in Fig 4. - curves with the same u_10 tend to stay close to each other (average windspeeds tend to give more similar probabilities of blowing snow), and the main difference within each u_10 group is the height of their peaks. In particular, a higher u_10 value and a lower t_h value both lead to a higher bell-curve peak (aka greater probability of blowing snow with higher average hourly windspeed and lower snow surface age). The temperature at which these peaks occur appear to shift to the left (ie. get colder) as u_10 (average hourly windspeed) increases.

Question 2.

a) A plot of wavefunctions for n = 0, 1, 2, 3 is displayed below. The wavefunction is an exponentially modulated form of the Hermite polynomials; in other words, it is the product of a Hermite polynomial and an exponential function that depends on position. The wavefunction is also modulated by a coefficient that depends on the order, n. The value of each Hermite polynomial was outputted using a recursive for-loop implemented in Python.

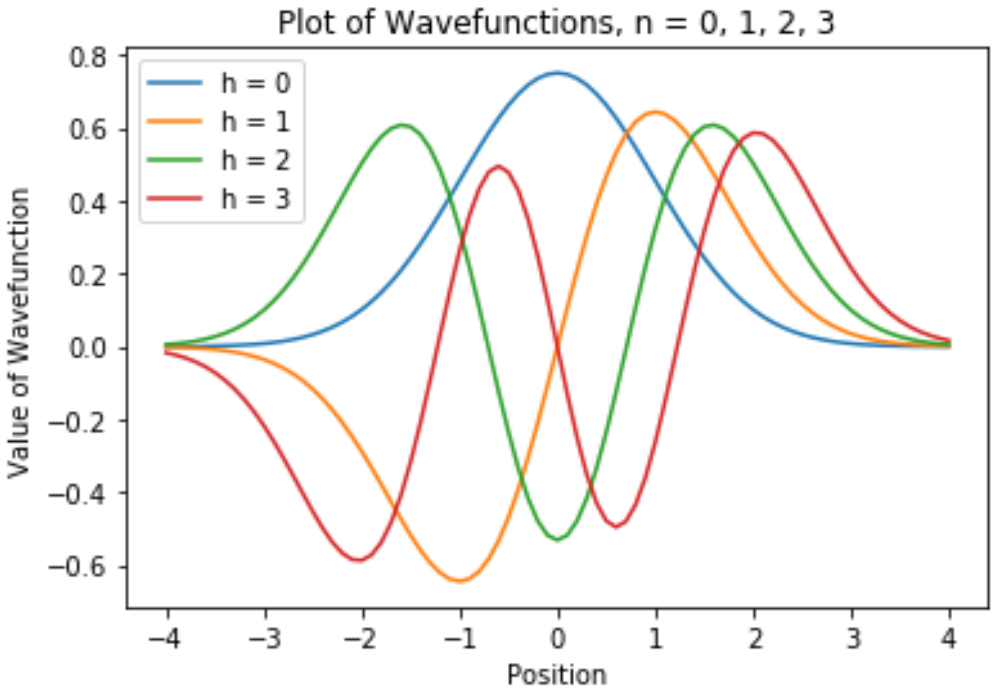

Fig 5. Plot of the Hermite Polynomials for n = 0, 1, 2, 3. It was observed that for a polynomial of order N, it contains N+1 peaks.

b) A plot of the wavefunction evaluated at n = 30 for x = [-10, 10] is displayed below.

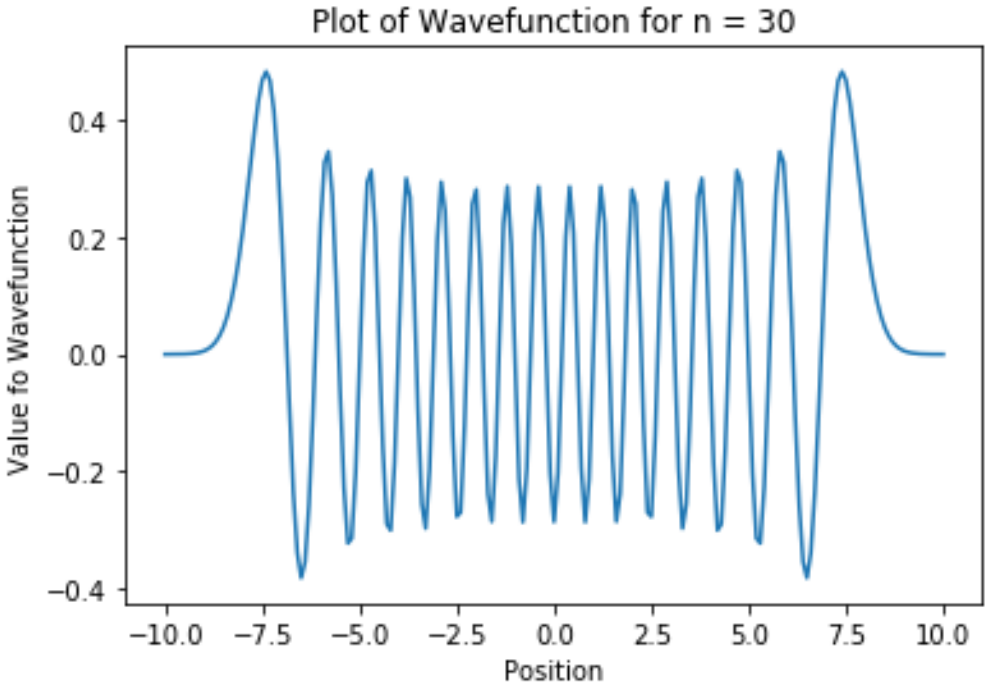

Fig 6. Plot of a wavefunction (order n = 30).

c) The expectation values of x^2 and p^2 were calculated and used to calculate the energy of the system. The functions created to calculate these expectations values implemented the Gaussian quadrature method of integration. Since the original

equations for expectation values contained improper integrals, we made the change of variable x = tan(z), as suggested by Newman (ch. 5.8).

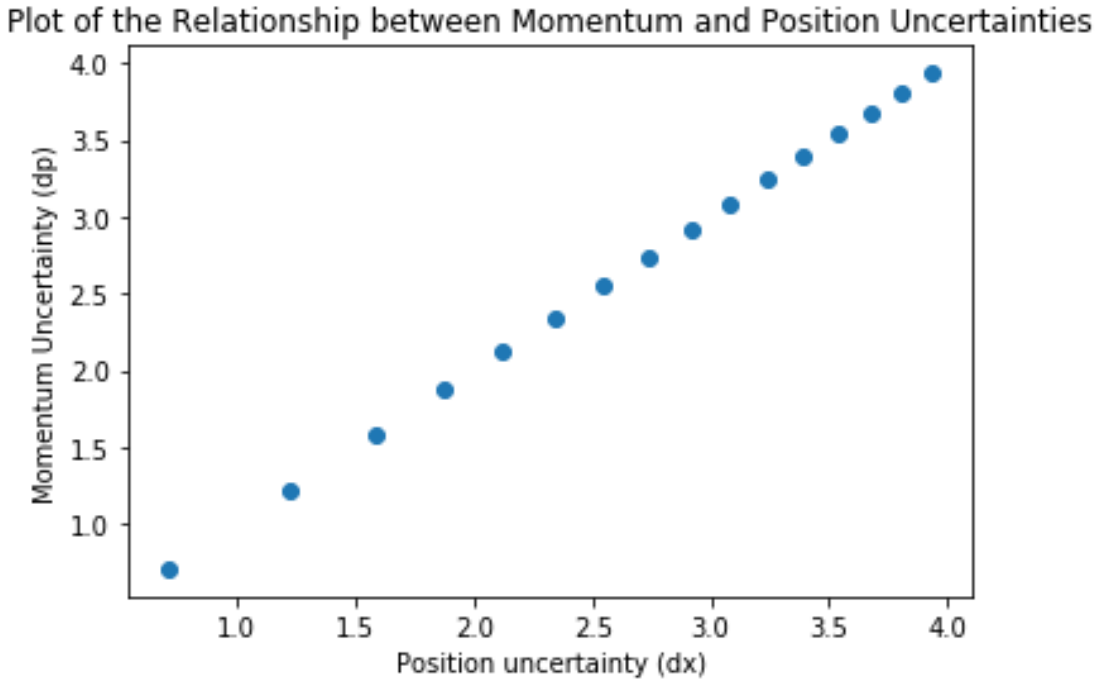

Fig 7. a) Plotting the relationship between position uncertainty and momentum uncertainty; it is evident that there is a linear relationship between these two uncertainties.

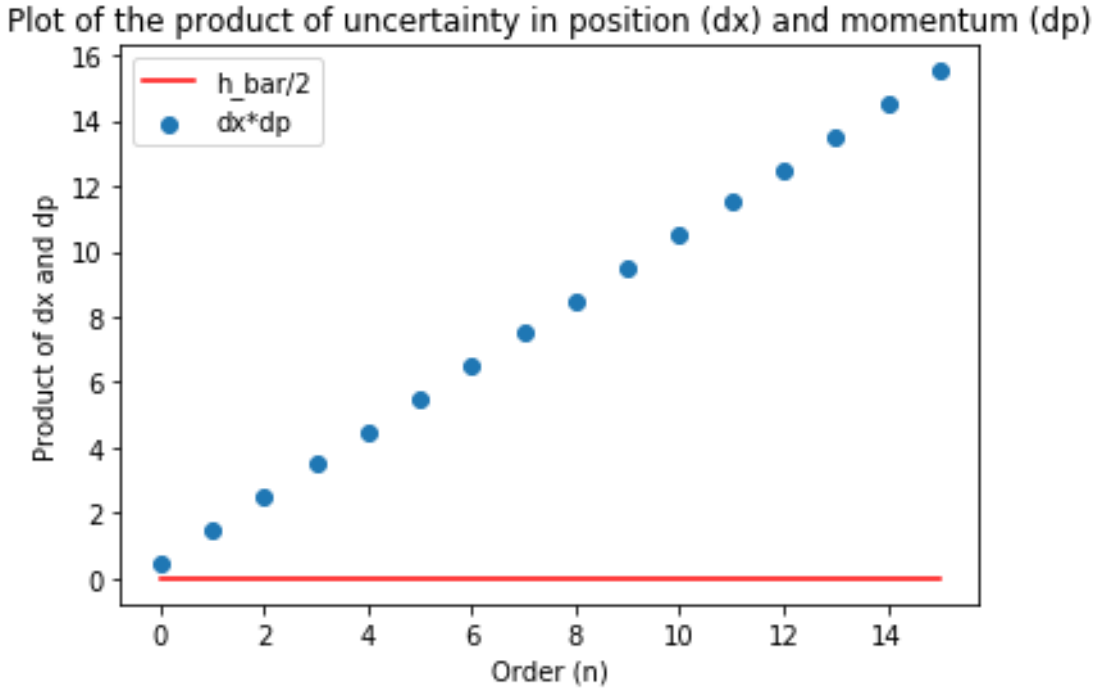

Fig 7. b) Plotting the product of dx and dp over a range of orders (N) to confirm Heisenberg's uncertainty principle, dx*dp ≥ hbar/2.

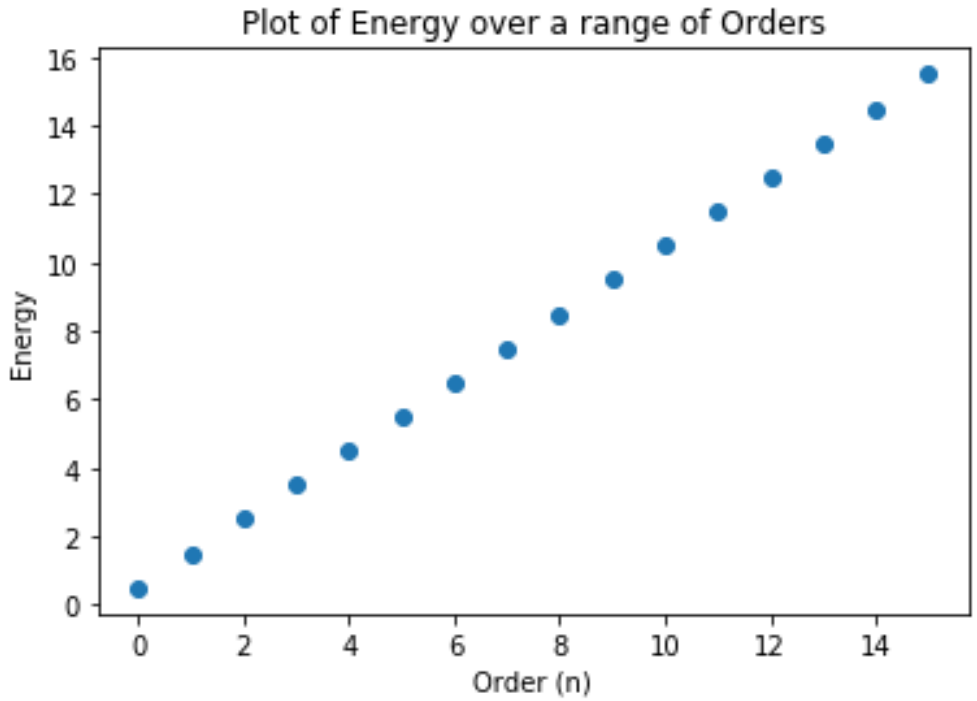

Fig 7. c) A plot of the energy's system for a range of orders (N). The energy is linearly dependent on N, which is the order of the associated Hermite polynomial – this is confirmation that the Hermite polynomials are useful in describing harmonic oscillations in QM.

```
In [29]: x_rms(5)
Out[29]: 2.3452078797796547

In [30]:
```

Figure 7. d) Displaying the output of position uncertainty when evaluated at 5; as expected, the output is approximately 2.35.

Question 3.

a) Pseudocode for Lab03_Q3.py:
   1) Data from the file N19W156.hgt was imported; a storage matrix (1201 x 1201) was created.
   2) The data was read, starting from the top left corner, moving towards the right, then shifting down to the next row once the end of the previous row was reached. The data was read 2 bytes at a time and stored to the appropriate index in the storage matrix

3) Two functions were created to calculate the derivative of the matrix with respect to x and y, respectively; this was done using the central difference method of numerical differentiation. For the function calculating d/dx, the iteration for x began at i = 1 and ended at i = N – 1 (N = 1201) to account for border issues; a similar troubleshoot method was implemented for d/dy.
4) A function was created to calculate the intensity of illumination for each point in the matrix (phase $= \pi$) based on eq. 16 provided in the lab manual.
5) Plots of the matrix (w) and intensity (I) were created using plt.imshow(); the minimum and maximum values were adjusted until the images were well-rendered.

b) The plots for w and I were created and displayed below. If we assume that the phase ϕ = π, then the intensity formula becomes $I = \frac{\frac{\partial w}{\partial x}}{\sqrt{\left(\frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial y}\right)^2 + 1}}$

The main difference between the w and I plots is the significant increase in resolution for the intensity (I) plot – the w plot displays a smoother gradient as the altitude value changes, whereas the I plot displays abrupt changes in intensity on a pixel-by-pixel basis. Thus, the I plot is a more accurate representation of the topography of Hawai'i.

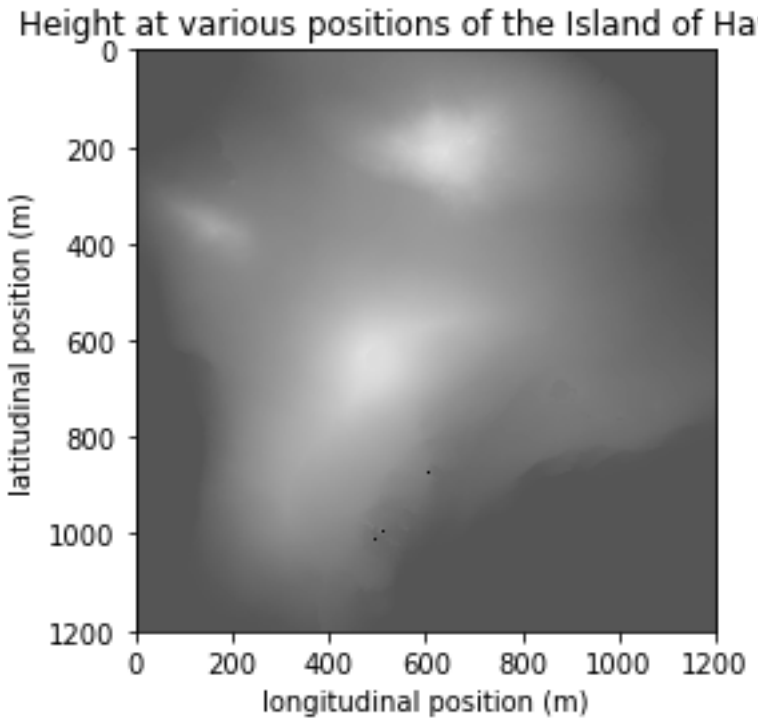

Fig 8. Plot of the altitude of Hawai'i's topography. The following volcanoes are visible in this plot as intensity peaks: Mauna Loa (bottom – most peak); Mauna Kea (top right peak), Hualalai (top left peak).

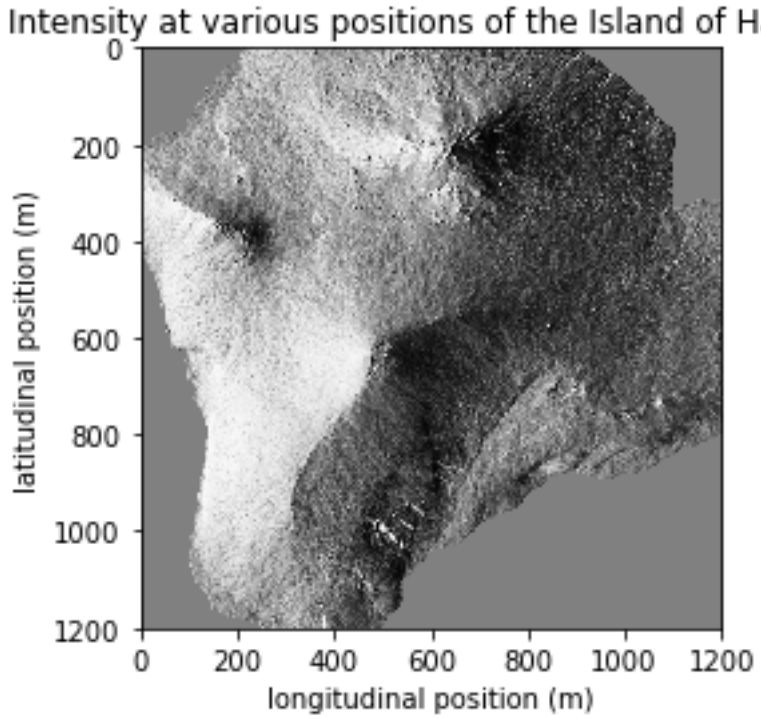

Fig 9. Plot of the illumination intensity of Hawai'i's topography, for phase ϕ = π. The following volcanoes are visible in this plot: Mauna Loa (bottom – most peak); Mauna Kea (top right peak), Hualalai (top left peak).

Below are close-up images of three different volcanoes visible in the intensity plot.

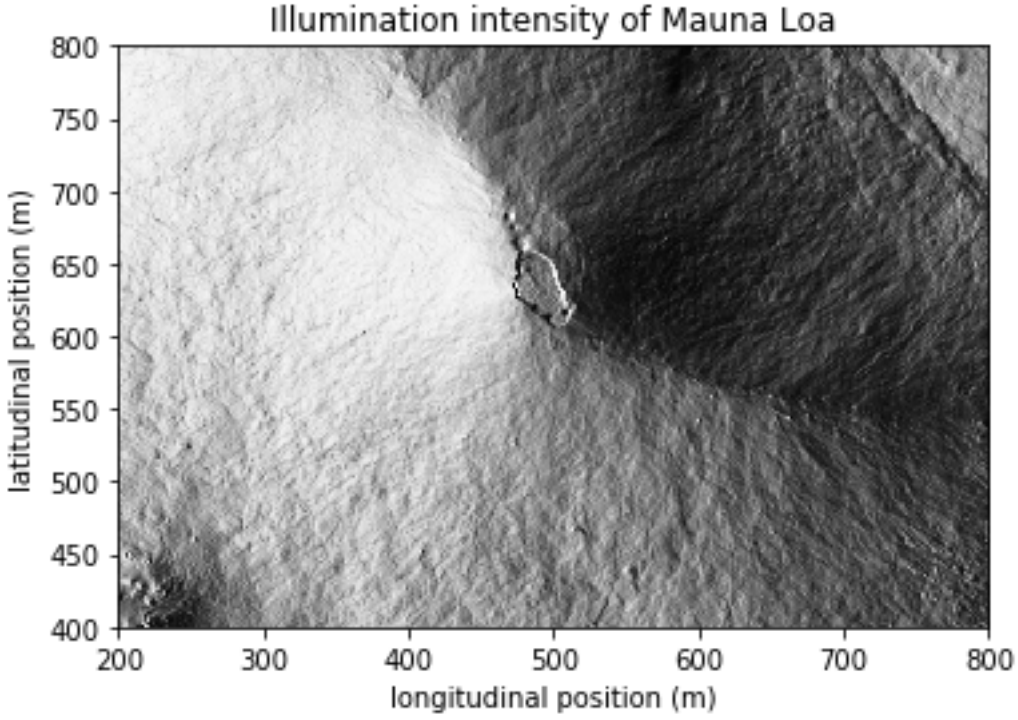

Fig 10 a). Close-up of Mauna Loa.

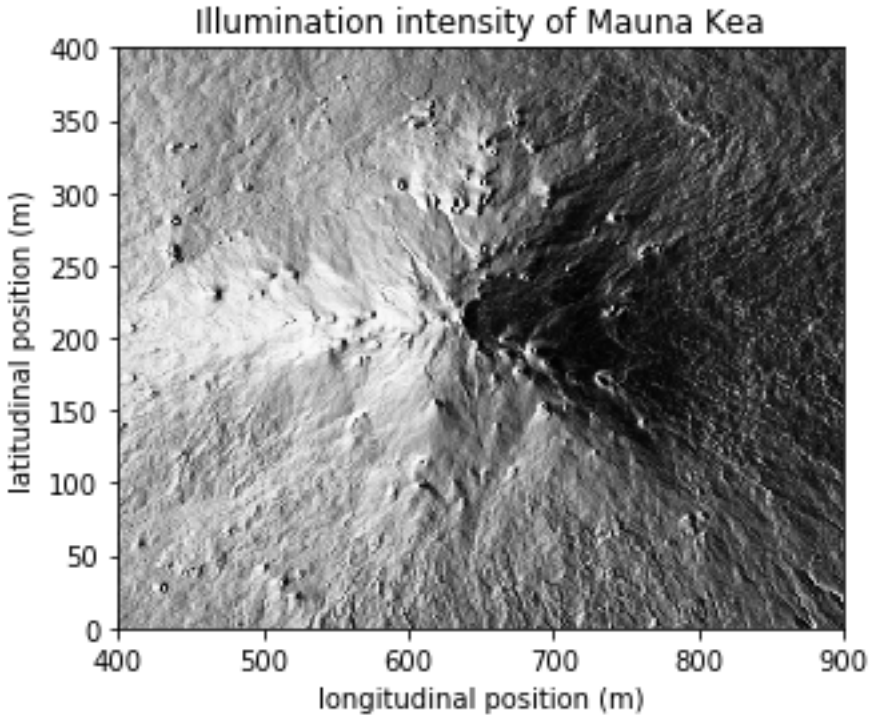

Fig 10 b). Close-up of Mauna Kea.

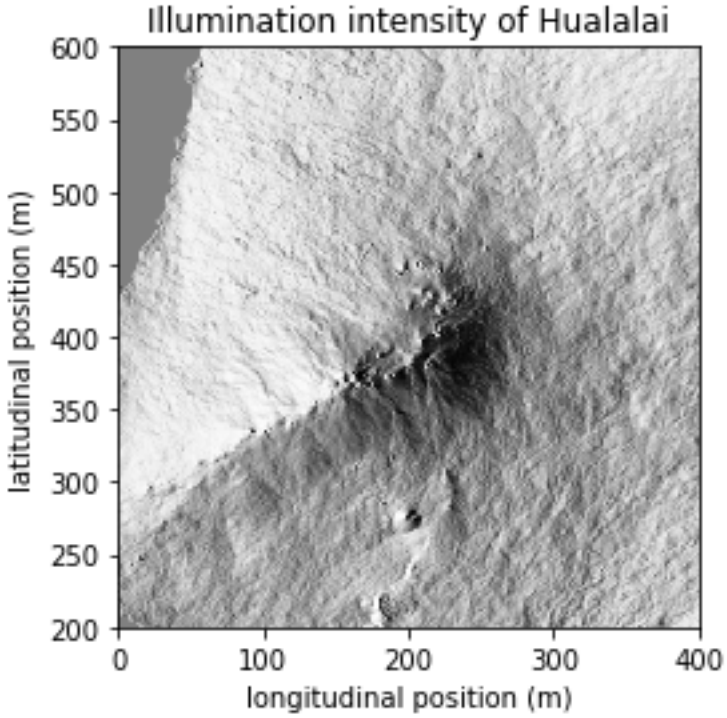

Fig 10 c). Close-up of Hualalai.