PHY407F: Explanatory Notes for Numerical Errors and Integration

Souren Salehi and Idil Yaktubay

23 September 2022

1 Question 1 by Idil Yaktubay

For question 1(b), we estimated the standard deviation of Michelsen's speed of light data in Python using equations 1 and 2, where $\bar{x}$ is the sample mean, n is the number of data points, and σ is the estimated standard deviation. To make the estimation using equation 1, we defined a function `std_1()` that first passes through the data to sum the data points one by one and to divide this sum by n to find $\bar{x}$, then passes through the data for a second time to sum the $(x_i - \bar{x})^2$ terms to calculate the standard deviation. To make the estimation using equation 2, we defined another function `std_2()` that only passes through the data once to simultaneously sum the data points x_i and the square of these data points x_i^2 and to calculate both $\bar{x}$ and $\Sigma_{i=1}^{n} x_i^2 - n\bar{x}^2$ to find the standard deviation. To find the errors in these calculations, we took the 'correct' standard deviation to be `correct_std = numpy.std(c_data, ddof=1)` where `c_data` is a one dimensional array containing Michelsen's data. Figure 1 shows our results.

$$\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \sigma \equiv \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{1}$$

$$\sigma \equiv \sqrt{\frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)} \tag{2}$$

```
QUESTION 1(B)

The 'correct' standard deviation of Michelsen's data is:  0.07901054781905067 x 10^3 km/s.
Standard deviation using equation 1 is:  0.07901054781905066  x 10^3 km/s
Standard deviation using equation 2 is:  0.07901054811458154  x 10^3 km/s
The relative error for equation (1) is:  1.7564474859226716e-14  %.
The relative error for equation (2) is:  3.740397602946237e-07  %.
```

Figure 1: Python code output for standard deviation estimations of Michelsen's speed of light data using equations 1 and 2. As shown, the error from equation 1 is less than that of equation 2.

For question 1(c), we estimated the standard deviations of two sets of data using equations 1 and 2. The data sets were generated using the function `numpy.random.normal(mean, sigma, n)` that gives a sequence of length n of data points taken randomly from normal distribution with mean `mean` and standard deviation `sigma` (from the lab manual). The first data set corresponds to `mean, sigma, n = (0., 1., 2000)` and the second data set corresponds to `mean, sigma, n = (1.e7, 1., 2000)`. As before, to calculate the errors in the estimates, the 'correct' standard deviations were generated using `numpy.std(data, ddof=1)`. Figure 2 shows our results.

```
QUESTION 1(C)

The 'correct' standard deviation of sequence #1 is:  0.9832147583684825
The 'correct' standard deviation of sequence #2 is:  0.9895308458531079
Standard deviation of data set #1 using equation 1 is:  0.983214758368483
Standard deviation of data set #1 using equation 2 is:  0.9832147583684824
Standard deviation of data set #2 using equation 1 is:  0.9895308458531076
Standard deviation of data set #2 using equation 2 is:  1.2778170480174782

The relative errors for the first distribution are:
5.645882627247316e-16 % for equation 1
1.129176525449463e-16 % for equation 2
The relative errors for the second distribution are:
3.365907275991975e-16 % for equation 1
0.2913362462347791 % for equation 2
```

Figure 2: Python code output for standard deviation estimations of two data sets randomly taken from two normal distributions. As shown, the error from equation 1 is less than that of equation 2 for both data sets. However, for the second data set, the error for equation 2 is *significantly* larger than that of equation 1.

(d) The possibility of a negative argument when using equation 2, as well as the problems that will arise when the standard deviation is small relative to the mean suggest that the one-pass method is not numerically sound. The only thing we could do to avoid the first problem was to implement a warning for when the computation meets a negative square root argument. To avoid the second problem, however, it is best to use a more numerically stable equation such as equation 1.

2 Question 2 by Souren Salehi and Idil Yaktubay

For question 2, we used the Trapezoidal and Simpson's rules to evaluate the integral given by equation 3.**(a)** The exact value of this integral is $\pi = 3.1415926535....$ **(b)** To numerically obtain the integral using the Trapezoidal rule, we used equation 4 with $N = 4$, $a = 0$, $b = 1$, and $h = \frac{b-a}{N}$. This Trapezoidal estimate of the integral was 3.1311764705882354, which is accurate up to $O(10^{-2})$. To do the same thing using Simpson's rule, we used equation 5 with the same values of N, a, b and h. Simpson's estimate of the integral was 3.14156862745098, which is accurate up to $O(10^{-5})$. Therefore, we have shown that Simpson's rule gives a much better numerical estimation of the integral from equation 3 than the Trapezoidal estimation.

$$\int_0^1 \frac{4}{1+x^2} dx \tag{3}$$

$$I(a,b) \approx h\left(\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{k=1}^{N-1} f(a+kh)\right) \tag{4}$$

$$I(a,b) \approx \frac{1}{3}h\left(f(a) + f(b) + 4\sum_{k\,odd:\,1}^{N-1} f(a+kh) + 2\sum_{k\,even:\,2}^{N-2} f(a+kh)\right) \tag{5}$$

(c) To increase the accuracy of our estimations, we increased the number of slices $N = 2^n$ until we could estimate the integral with an accuracy of $O(10^{-8})$ and therefore an error of $O(10^{-9})$. We found that this condition is met at $N = 2^{13}$ for the Trapezoidal rule, and at $N = 2^4$ for Simpson's rule.

(d) To further investigate the errors associated with the Trapezoidal rule, we employed the "practical estimation of errors" from the textbook. Specifically, to estimate the error for a Trapezoidal estimation of the integral from equation 3 with 32 slices, we calculated a Trapezoidal integral with 16 slices and used equation 6 to estimate the error. In this case, I_2 is the estimated integral for 32 slices, I_1 is the estimated integral for 16 slices, and ϵ_2 is the estimated error for I_2. We found this error to be 0.00016276037786200348, or, to the appropriate digits, ± 0.0002. Our findings are depicted on Figure 3.

$$\epsilon_2 = \frac{1}{3}(I_2 - I_1) \tag{6}$$

```
QUESTION 2(B)

Actual value of integral =  3.141592653589793 ...
Integral for trapezoidal rule with 4 slices =  3.1311764705882354
Integral with Simpson's rule with 4 slices =  3.14156862745098

QUESTION 2(D)
Practical estimation of error for Trapezoidal rule for 32 slices =
0.00016276037786200348
```

Figure 3: Python code output for Trapezoidal and Simpson's estimations for the integral from equation 3. We used 4 slices for both estimations. Additionally, we calculated the practical error for a Trapezoidal estimation with 32 slices.

(e) The practical estimation method will not work with the Simpson's rule in our particular case working with this integral because the third derivatives of the integrand evaluate to zero at 1 and 0. The practical estimation formula from the textbook was derived based on the assumption that these derivatives give us definite, nonzero values. So, for the estimation to work, we need shrink the area under the integrand from 0 to 1 by slightly changing the lower and upper bounds to values where the third derivatives exist and are nonzero. In other words, we could estimate the integral from $0 + \epsilon$ to $1 - \epsilon$ where ϵ is an appropriately small positive number.

3 Question 3 by Souren Salehi

3.1 (3a)

From the definition of black body radiation we get,

$$W = \pi \int_0^\infty B d\nu = \pi \int_0^\infty \frac{2hc^2\nu^2}{e^{\frac{hc\nu}{kT}} - 1} d\nu. \tag{7}$$

We can then do a substitution of,

$$x = \frac{hc\nu}{kT}, \tag{8}$$

so that

$$\frac{dx}{d\nu} = \frac{hc}{kT}$$

which substituting back to equation 7

$$W = \pi 2hc^2 \int_0^\infty \frac{k^3T^3x^3/h^3c^3}{e^x - 1} \cdot \frac{kT}{hc} dx \tag{9}$$

$$W = \frac{2\pi k^4T^4}{h^3c^2} \int_0^\infty \frac{x^3}{e^x - 1} dx, \tag{10}$$

therefor the constant C_1 is given by

$$C_1 = \frac{2\pi k^4T^4}{h^3c^2} \tag{11}$$

3.2 (3b)

Numerical integration allows us to calculate bounded integrals, therefore to calculate the improper integral shown in equation 7 we need to split up the integration as such.

$$\int_0^\infty \frac{x^3}{e^x-1}dx = \int_0^{\frac{1}{2}} \frac{x^3}{e^x-1}dx + \int_{\frac{1}{2}}^\infty \frac{x^3}{e^x-1}dx \tag{12}$$

The first part of the integral can be easily calculated numerically (note that the integration can't start at 0, rather at a very small number as the function does not exist at 0). For the second part of the integral we may use the substitution of $x = 1/t$ to change the bounds of integration.

$$\int_{\frac{1}{2}}^\infty \frac{x^3}{e^x-1}dx = -\int_2^0 \frac{1}{t^5(e^{\frac{1}{t}}-1)}dt = \int_0^2 \frac{1}{t^5(e^{\frac{1}{t}}-1)}dt \tag{13}$$

Equation 13 can then be solved using numerical methods and the sum of the two segments would allow us to calculate the improper integral (similar to the 0-1/2 segment the integral cannot start directly as 0 but rather a very small number as function is not defined at 0).

To calculate the error in the integral we can use the practical error formula for the Simpson rule from equation 5.29 in the textbook. We can rerun the integral for half the number of steps to find the error in our integral. In our case 50 steps were used for the $0 - 1/2$ segment and 200 steps for the $1/2 - \infty$ segment. Therefore we rand the program for 25 and 100 steps respectively and using the difference calculated the error in our integral.

3.3 (3c)

The Stefan-Boltzmann constant is given by,

$$\sigma = \frac{2\pi k^4}{h^3c^2}\int_0^\infty \frac{x^3}{e^x-1}dx. \tag{14}$$

Therefore using our calculated value of the integral and its error we can compute our estimate for the Stefan-Boltzmann constant. Using our value of the integral we get our constant to be $5.67037 \cdot 10^{-8} \pm 8 \cdot 10^{-13}$ where the error is from the propagation of our error in the integral.

4 Question 4 by Souren Salehi and Idil Yaktubay

(a) Figure 4 shows the graphs of functions $p(u)$ and $q(u)$ given by equations 15 and 16 at the vicinity of $u = 1$. As we can see, the plot of $q(u)$ is much noisier than the plot of $p(u)$. This is because for $p(u)$, the rounding errors only come from taking the difference $1 - u$ and raising it to the power of 8. However, $q(u)$ has a rounding error associated with each term due to the operations involved with them (multiplication, raising to a power etc.), as well as a rounding error associated with the summation/subtraction of these eight terms. Therefore, a much larger amount of error accumulates when calculating the values of $q(u)$.

$$p(u) = (1-u)^8 \tag{15}$$

$$q(u) = 1 - 8u + 28u^2 - 56u^3 + 70u^4 - 56u^5 + 28u^6 - 8u^7 + u^8 \tag{16}$$

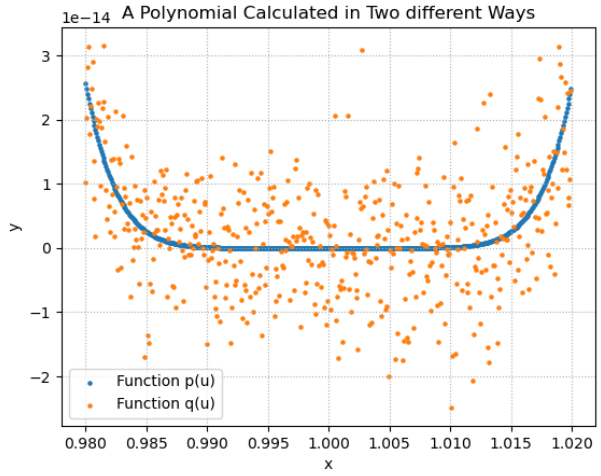

Figure 4: Plots of functions from equations 15 and 16 at the vicinity of $u = 1$. The orange plot represents $q(u)$, and the blue plot represents $p(u)$.

(b) Figure 5 shows the graph of $p(u) - q(u)$ near $u = 1$. As we can see, even though the two functions are mathematically equivalent, they differ due to rounding errors. A histogram of these differences is given by Figure 6. Using the `numpy.std()` function, we found the standard deviation of this distribution to be $\sigma \approx 0.8 \times 10^{-14}$. Further, equation 17 gives the error on the sum of N terms $x = x_1 + ... + x_N$. Here, we take the error constant C to be 10^{-16}. Therefore, the error associated with the sum $p(u) - q(u)$ for values close to $u = 1$ can be calculated using equation 17. We calculated this quantity for $u = 0.999$ and found that $\sigma \approx 1.1 \times 10^{-14}$, which is of the same order as the standard deviation of the $p(u) - q(u)$ distribution with a $O(40\%)$ difference. This is because each value from the distribution of $p(u) - q(u)$ comes from the error distribution with a standard deviation given by equation 17.

$$\sigma = C\sqrt{N}\sqrt{\bar{x^2}} \tag{17}$$

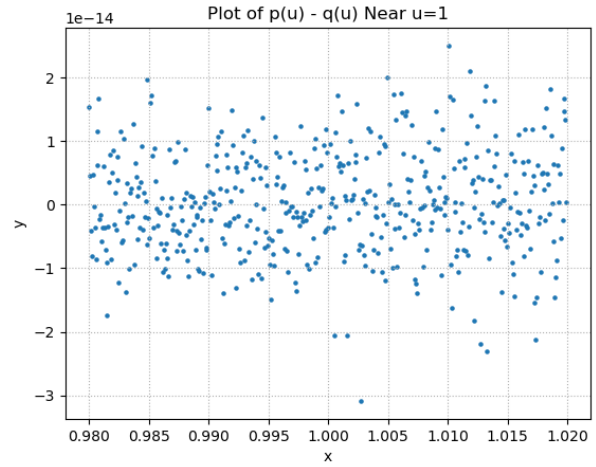

Figure 5: Plot showing the difference $p(u) - q(u)$ at values close to $u = 1$.

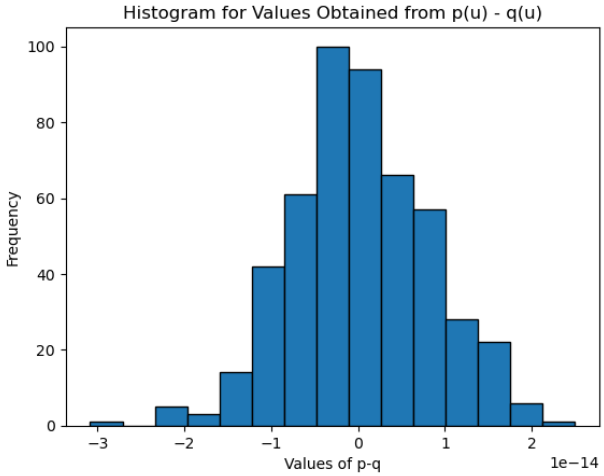

Figure 6: Histogram corresponding to the plot depicted by Figure –. The standard deviation of this distribution is $\sigma \approx 0.8 \times 10^{-14}$ as estimated by the `numpy.std()` function.

(c) Equation 18 gives the fractional error on the sum of N terms $x = x_1 + ... + x_N$. As before, we take the error constant C to be approximately 10^{-16}. With this, we can show that for values of u greater than 0.980 but less than 1.0 the fractional error of $q(u)$ is around 100%. To do this, we created a numpy.array that starts at 0.980 and goes up to 0.984 with increments of 0.0001. With each value of u from this array, we calculated the fractional error as given by equation 18. We found that the values of u larger than about 0.9828, the fractional error approaches ≈ 1.0. Figure 7 shows the fractional error for several values of u between 0.980 and 0.984. Additionally, Figure 8 shows the error of $q(u)$ relative to $p(u)$. In this case, the error approaches 1.0 at about 0.9828 as well.

$$\frac{\sigma}{\sum_i x_i} = \frac{C}{\sqrt{N}} \frac{\sqrt{\bar{x^2}}}{\bar{x}} \tag{18}$$

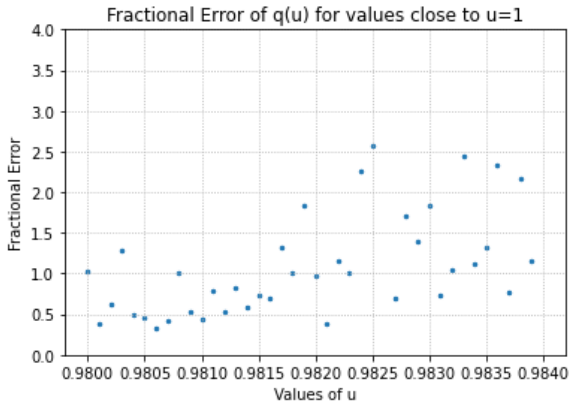

Figure 7: Fractional Error of $q(u)$ calculated with equation 18. As we can see, this quantity is very noisy. Apart from the noise, we see that it approaches 1.0 at about 0.9828.

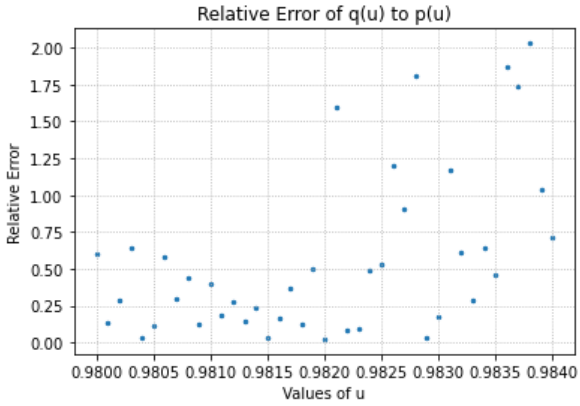

Figure 8: Error of $q(u)$ relative to $p(u)$. We see that the relative error approaches to 1.0 at about 0.9828.

(d) To show how rounding error comes up in products and quotients, we defined a quantity `f = u**8/((u**4)*(u**4))` and plotted it for 500 points in the range $0.98 < u < 1.02$. This quantity showed a range of values around 1.0, as shown on Figure 9. However, each of these values has a rounding error. To show this, we plotted `f-1` versus `u`, which is shown on Figure 10. Further, we found the standard deviation of $f(u)$ for values of u within this range to be `1.4678475218444484e-16`. We also found an estimate of the error on quantity f to be `1.414213562373095e-16` using equation 19. These values are of the same order, just as our values from **(b)**.

$$\sigma = \sqrt{2}Cx \tag{19}$$

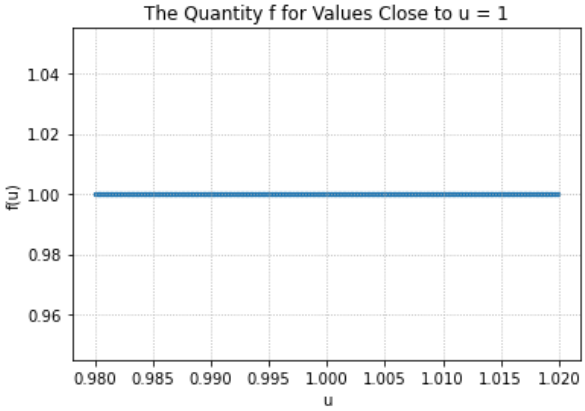

Figure 9: A graph of the quantity f at the vicinity of $u = 1$.

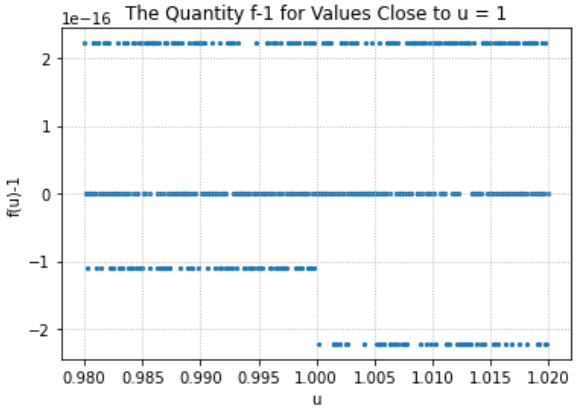

Figure 10: A graph of the quantity $f - 1$ at the vicinity of $u = 1$.