Jiachen Wan N19996964

Homework 1

Computaitonal Physics
September 26, 2019

Problem 1

good! 10/10

1. The code for forward, central and extrapolated difference methods are in the code folder. The equations used are:

$$f'(x) = \frac{f(x+h) - f(x)}{h} \text{ for forward method}$$

$$f'(x) = \frac{f(x+\frac{h}{2}) - f(x-\frac{h}{2})}{h} \text{ for central difference method}$$

$$f'(x) = \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-h)}{h} \text{ for extrapolated difference method}$$

2. Firt I made sure that all numbers involved in the calculations are in single precision, using the float32 function in numpy. So I tested with f(x)=cos(x), x=0.1

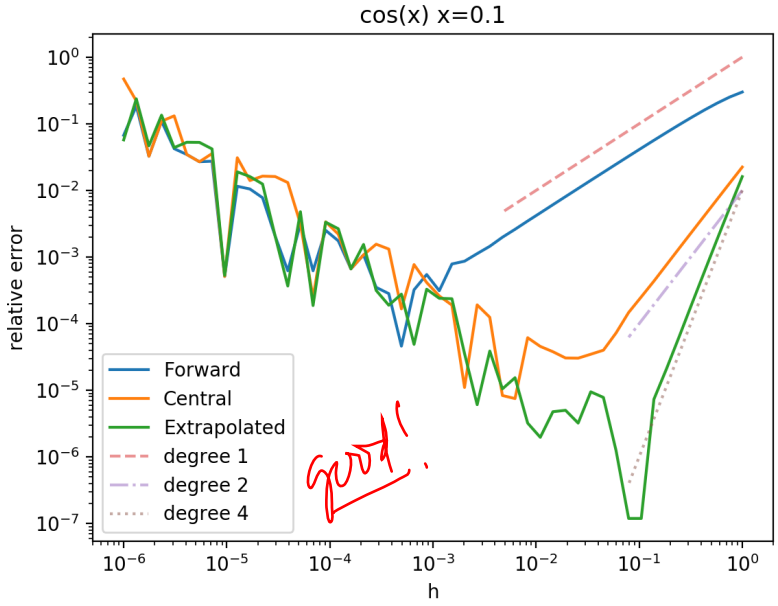

Figure 1: Derivative of cos(x) using different methods at x=0.1

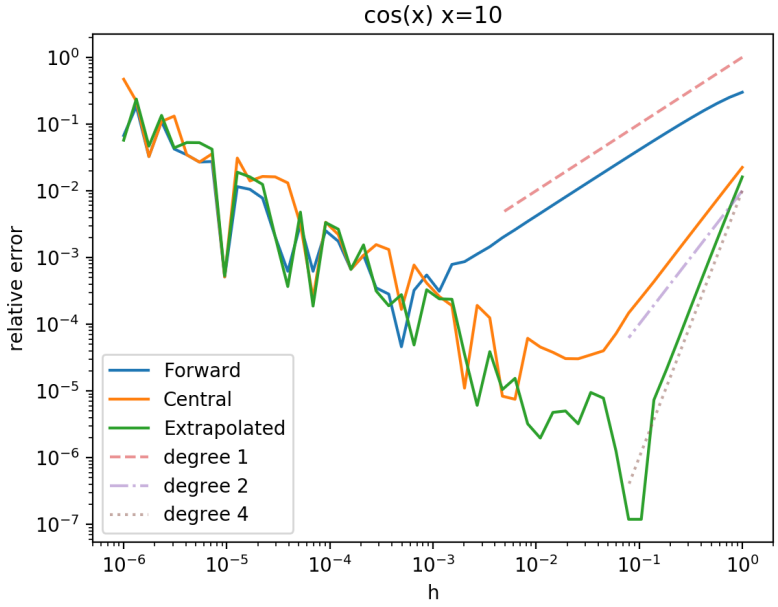

Figure 2: Derivative of cos(x) using different methods at x=10

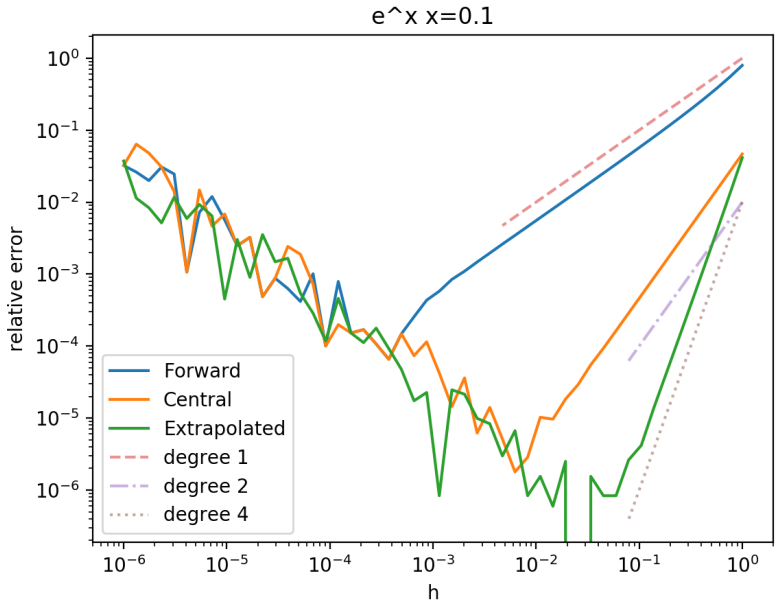

Figure 3: Derivative of e^x using different methods at x=0.1

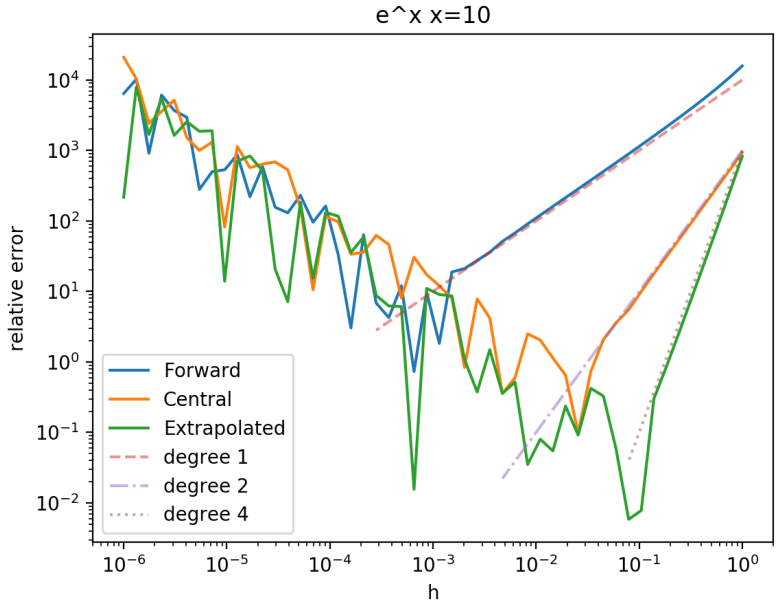

Figure 4: Derivative of e^x using different methods at x=10

Since we know that the relative error for Forward difference is $O(h^1)$, for Central difference is $O(h^2)$, and for Extrapolated difference is $O(h^4)$, it is clear that the plot matches my expectation. As for the number of significance, all results' significant digit were at most 8. It is also expected because I was using single precision for my calculations.

3. Truncation error refers to the error originated from algorithm. In short, better algorithm has smaller truncation error. When the steps are small, truncation error dominants. This is the part of the graph where the relation between h and error is consistent. For the Extrapolated difference method, truncation error dominates until roughly $h = 10^{-1}$. When the steps get too small, roundoff error becomes relevant. It occurs when the significant figures in each number(in the calculation) differ by more then what the machine can store. In case of singal precision, the machine precision is 10^{-7}. So when the step gets too small, a lot of roundoff error compile up, and such error increases when steps decrease further. That's why we see that the relative error start to increases when h becomes too small.

Problem 2

1. The equation I used are:

$$\int_a^b f(x)dx = \sum_{i=0}^{N-1} \frac{f(a+(i+\frac{1}{2})h)}{2} h \text{ for midpoint rule}$$

$$\int_a^b f(x)dx = (\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{N-1} f(a+ih))h \text{ for trapezoid rule}$$

$$\int_a^b f(x)dx = \frac{h}{3}[f(a)+f(b)+4\sum_{i=1}^{N/2} f(a+(2i-1)h)+2\sum_{i=1}^{N/2-1} f(a+2ih)] \text{ for Simpson's rule}$$

   The codes are in the folder.

2. The function to calculate is:

$$\int_0^1 e^x dx$$

   and I have used the float32 function to make sure it is single precision. All of the chosen N are even numbers so that the Simpson's Rule will work properly. The resulting graph is:

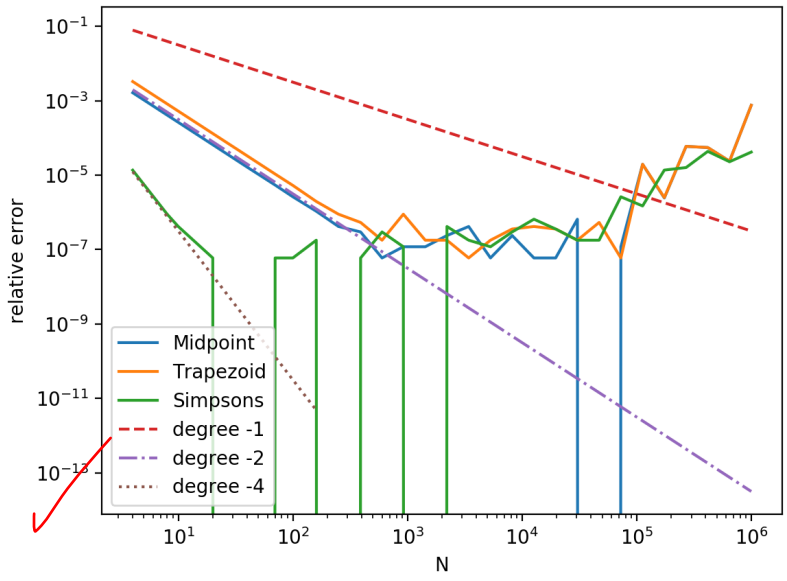

Figure 5: Integration of e^{-x} using different methods from 0 to 1

3. First of all, we can clearly see that the relative error matches expectation. The error for midpoint rule scales with N^{-2}, the error for Trapezoid rule scales with N^{-2}, and the error for Simpson's rule scales with N^{-4}. These can all be seen on the graph. We also know that the error for midpoint rule is half of the error for trapezoid rule, so that is why the curve for midppint rule is right under that of trapezoid rule. Secondly, we also observe the effect of roundoff error. When the relative error decrease to around 10^{-7} it stops decreasing, because when N gets too large, the steps are too small such that some digits cannot be stored on the machine, and it accumulates to produce error. Thus when N gets too big, we can see that the relative error actually increases, due to the accumulated roundoff error.

Problem 3

1. To start with, here is the P(k) obtained from the file:

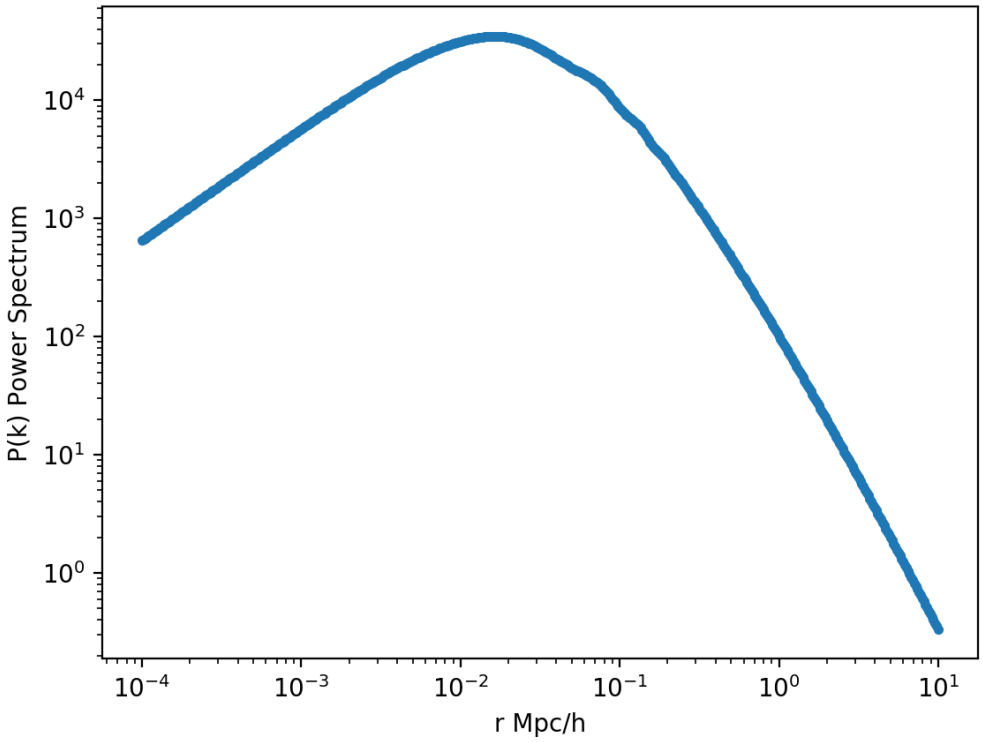

Figure 6: P(k) plot against k in a log-log scale

and I do observe an oscillation at r=0.1, as stated in the document. The next step is

to create an interpolation. I used the cubic spline interpolation function from scipy. To check the function worked well, I have plotted the result to check if it is smooth:

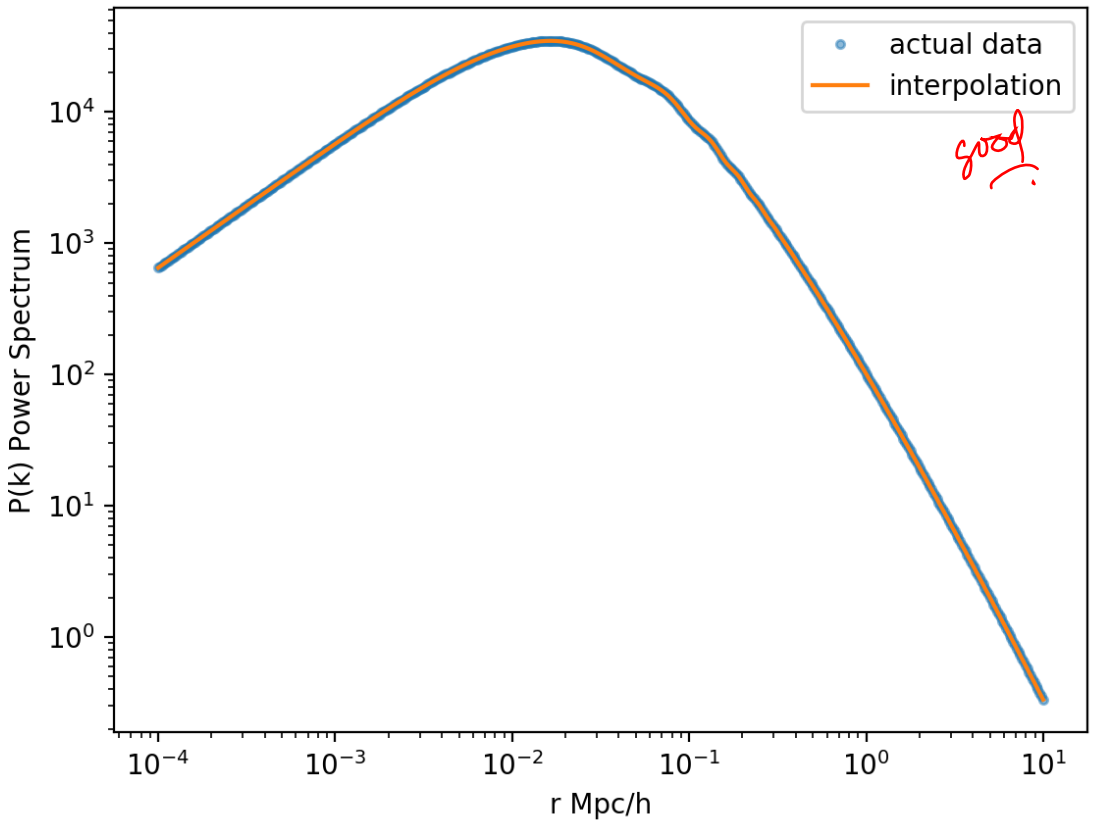

Figure 7: Interpolation and the origional data. It is clear that the interpolation works fine.

Looking at the graph, it is safe to assume that the interpolation is behaving normally. Now I must determine the range of the integration. The range is from 0 to infinite, but we know that P(0)=P(inf)=0. However, the interpolation of P(k) do not behave well at P(0) and P(inf). P(0) = -7.09780581, and P(inf)=-inf. So I must find position a and b for my integration, such that a is around zero and P(a)=0, while b is bigger than 0 and P(b)=0. To do so I ran the binary search around k=0 and k=14:

$$P(9.8562 * 10^{-7}) = -9.63994736 * 10^{-7}$$

$$P(14.4864) = 1.19436671 * 10^{-6}$$

so these are my chosen boundaries. Using the boundary and the interpolation to integrate using N = 1000, I have created a plot of $r^2\xi(r)$:

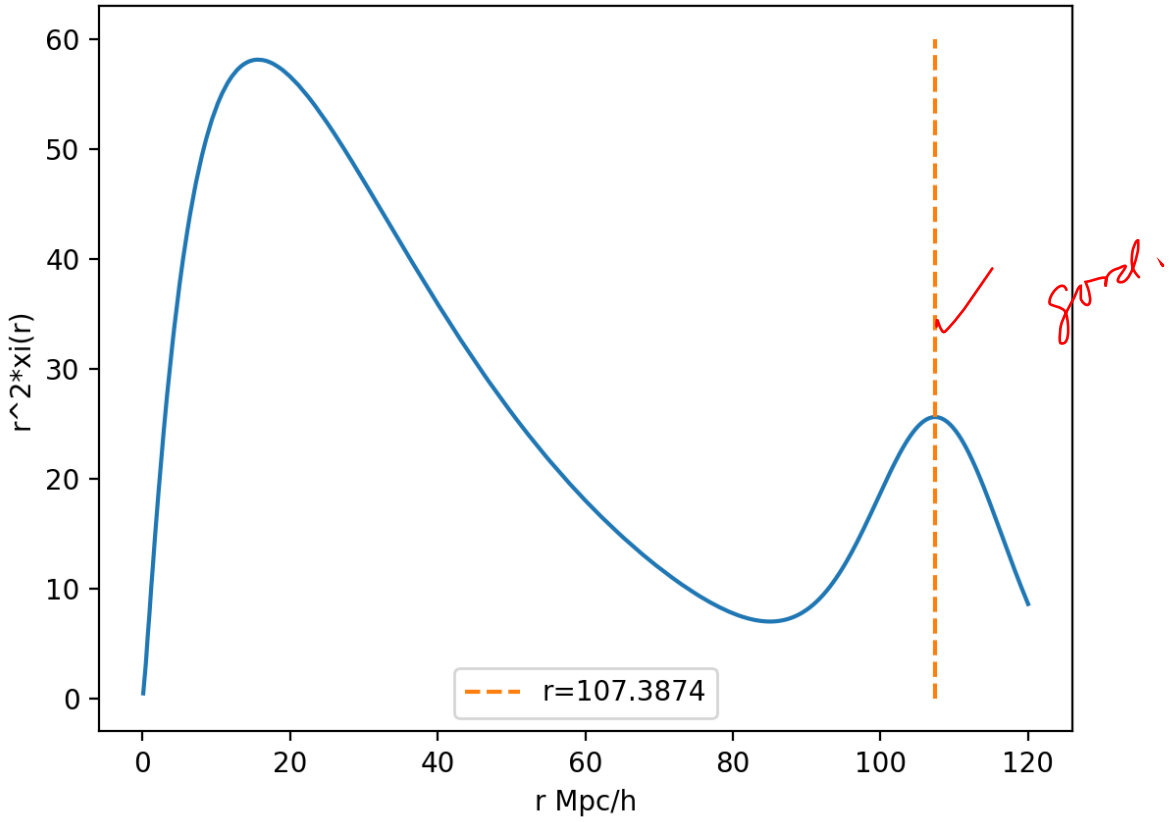

Figure 8: plot of $r^2\xi(r)$ against r. We can observe the baryon acoustic oscillation peak at around r=100 Mpc/h

In the paper 'DETECTION OF THE BARYON ACOUSTIC PEAK IN THE LARGE-SCALE CORRELATION FUNCTION OF SDSS LUMINOUS RED GALAXIES (2005)'[1], it clearly states that The bump at $100h^1Mpc$ is the acoustic peak. So I searched for a local maximum in the range from r=100 to r=120. The local maximum is found at:

$$r = 107.3874Mpc/h$$

which is the scale of baryon acoustic oscillation peak.

Citation

Eisenstein, D. J.; et al. (2005). "Detection of the Baryon Acoustic Peak in the LargeScale Correlation Function of SDSS Luminous Red Galaxies". The Astrophysi-

cal Journal. 633 (2): 560574. arXiv:astro-ph/0501171. Bibcode:2005ApJ...633..560E. doi:10.1086/466512