Computational Physics Homework I

Link to GitHub repository

Excellent work. 10/10

1 Errors in numerical derivatives

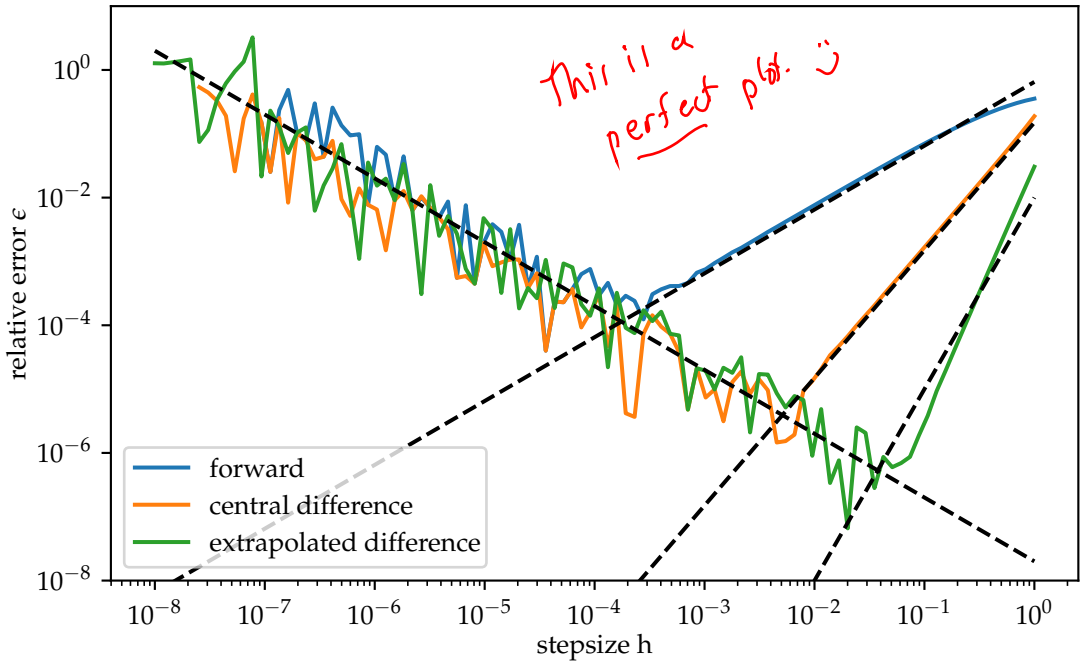

Figure 1: Relative error of different numerical differentiation techniques in dependence of the step size. Dashed lines indicate the power laws discussed in the main text and serve as a guide to the eye.

Fir this exercise I differentiate $f(x) = \cos(x)$ and $g(x) = \exp(x)$ using the forward, central difference, and extrapolated difference methods we discussed in the lecture. For each method, I plot the relative error ϵ, comparing the numerical derivatives to the analytical ones. The results shown here are for $f'(x = 10)$, but they are largely independent of the specific value of x and the chosen function.

Since python normally works with doubles, I set all values to single precision manually (same for the second exercise).

The results of the calculation for all three methods are shown in Figure 1. In all three graphs we see that roundoff error dominates for small stepsize h, then an optimal value

h_{opt} for h is reached, and when h increases further, the relative error increases again due to truncation error. The value of the optimal stepsize h_{opt} depends on the method used for differentiation.

We can see that the roundoff error scales similarly for all three methods, but for the relative error we obtain different power law behaviours depending on the method. As expected, the more advanced the method is, the lower the error is that can be reached for the optimal stepsize h_{opt}, with the best performance obtained by the extrapolated difference method.

For the forward method, shown in blue, we expect $\epsilon_{\text{r}} \propto 1/h$ as the scaling for for the relative roundoff error, and $\epsilon_{\text{t}} \propto h$ for the truncation error. As we discussed in the lecture, we expect an optimal stepsize h_{opt} on the order of $\sqrt{\epsilon_{\text{m}}} \approx 3 \times 10^{-3}$, and an optimal relative error of the same order of magnitude, which is what we can see in the plot. Here, ϵ_{m} is the machine error of about 10^{-7}.

The results of the central difference method are shown in orange. Here, we expect the truncation error to scale as $\epsilon_{\text{t}} \propto h^2$. We see that, as expected, a larger h_{opt} of $\epsilon_{\text{m}}^{1/3} \approx 5 \times 10^{-3}$, and a correspondingly smaller optimal relative error of about $\epsilon_{\text{m}}^{2/3} \approx 10^{-5}$.

The extrapolated difference method shows the best optimal performance out of the three methods examined. Due to the cancellation of h^2-terms in the calculation of the derivative, the truncation error scales as $\epsilon_{\text{t}} \propto h^4$. We reach the smallest relative error of about $\epsilon_{\text{m}}^{3/5} \approx 10^{-6}$ for a step size of $h_{\text{opt}} \approx \epsilon_{\text{m}}^{1/5} \approx 4 \times 10^{-2}$.

excellent.

2 Errors in numerical integrals

I evaluate the integral

$$I = \int_0^1 \exp(-t)\,\mathrm{d}t\,, \tag{1}$$

using the midpoint rule, trapezoid rule, and Simpson's rule as discussed in the lecture, for different numbers of steps N. For each number of steps, I calculate the absolute error of the evaluation by comparing to the analytic result $I = 1 - 1/\mathrm{e}$. The results for all three methods are depicted in Figure 2.

We see that all three methods converge to an absolute error of $\approx 10^{-8}$, limited by the machine precision. I was not able to see the effects of roundoff error in my calculations.

it's there, it just doesn't accumulate like expected in C.

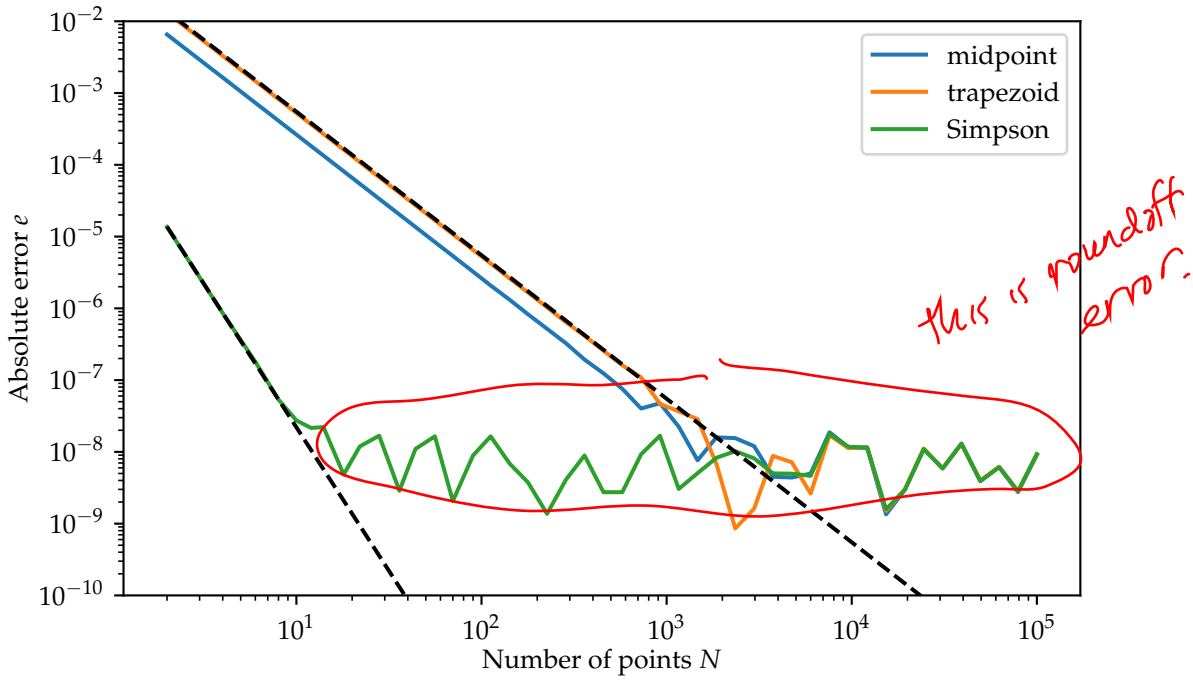

Figure 2: Absolute error of different numerical integration techniques in dependence of the number of steps used. Dashed lines correspond to the power laws discussed in the text.

The midpoint and trapezoid method follow the same power law behaviour of $e \propto N^{-2}$, however the midpoint method performs slightly better since the evaluation of the function in the middle of the interval yields a small advantage. Simpson's method follows the expected $e \propto N^{-4}$ scaling.

3 Density fluctuations in matter

I start from the given power spectrum of the density fluctuations in the universe, plotted in Figure 3. I use the numpy interpolation function, and sample at closely and evenly spaced points. I check that the interpolation resembles the initial data well, and proceed to calculate the power spectrum

$$\xi(r) = \frac{1}{(2\pi)^2} \int k^2 P(k) \frac{\sin(kr)}{kr} \, \mathrm{d}k \,. \tag{2}$$

I use the trapezoid method for the evaluation of the integral. The resulting correlation function, multiplied with r^2, is shown in Figure 4. I find the peak maximum at roughly $r \approx 106\,\mathrm{Mpc/h}$.

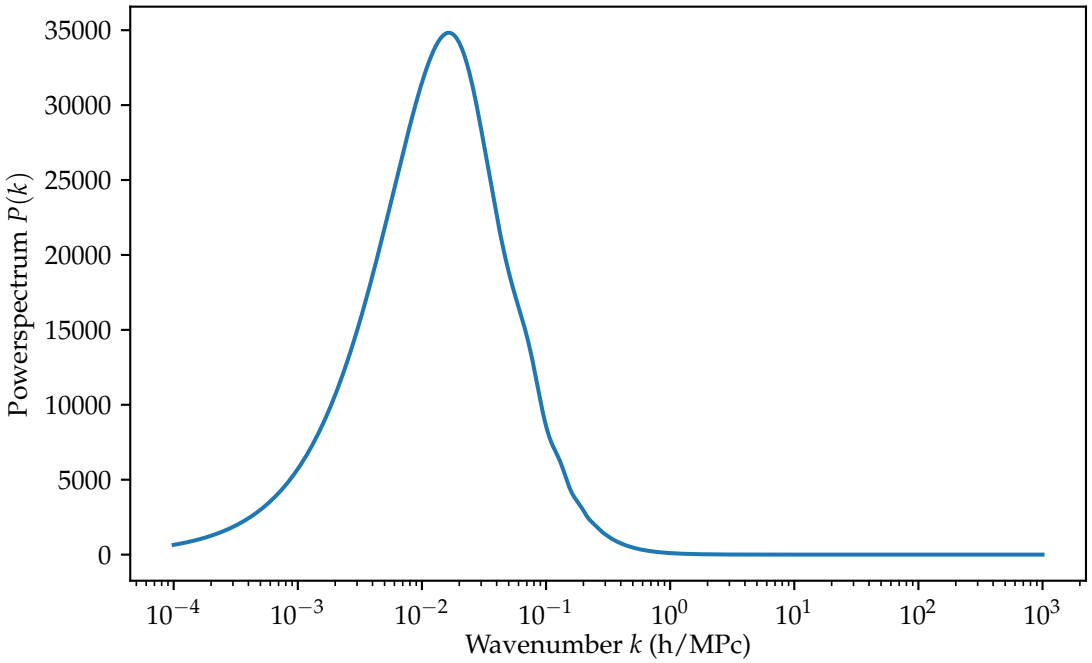

Figure 3: Power spectrum of density fluctuations in the universe. Semilogarithmic plot of given initial data.

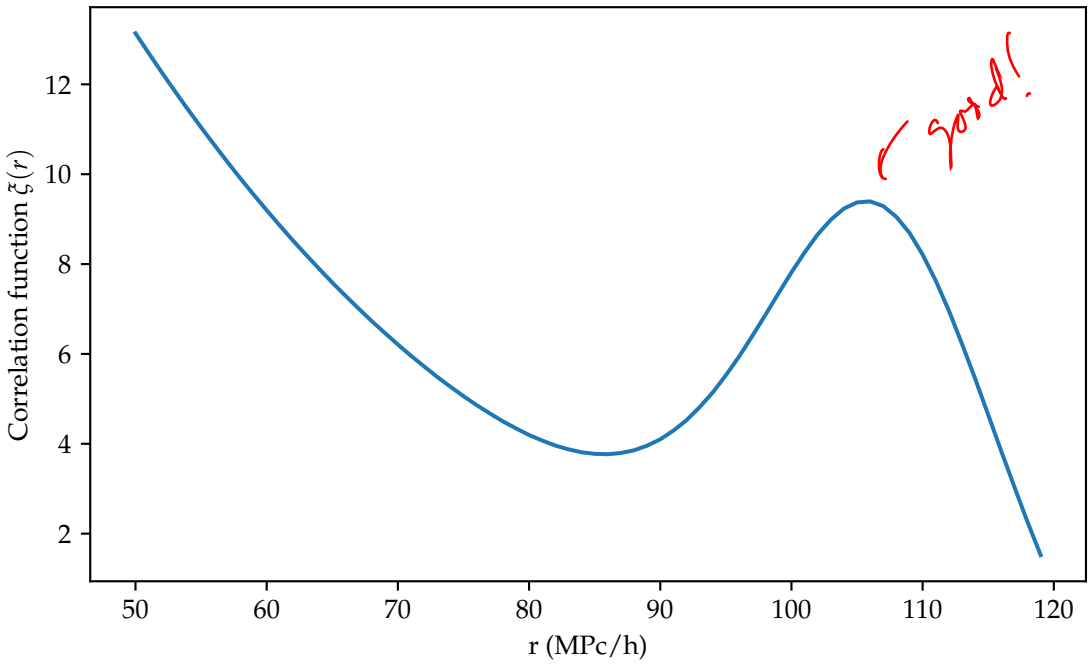

Figure 4: Correlation function of the power spectrum shown in Figure 3. The function has been multiplied with r^2 in order to visually enhance the peak.