Computational Physics II Report 3

Sebastian Mendoza

November 2021

1 Creating the functions

1.1 The Rectangular Method

To create the rectangular integration function, an analogue to a grid was first created: instead of creating an N dimensional array, containing for each dimension a total of n_p/N steps, an N-dimensional array was assigned to each number $j \in \{1, 2, ..., n_p\}$. In order to do this, a new quantity 'num' (which we will now call κ) was defined equal to $n_p^{1/N}$, which represents the number of points per dimension. Then, the k'th coordinate of j was assigned according to the formula

$$j_k = \frac{j}{\kappa^k} \mod \kappa$$

There is a bijection between this representation of each number from 1 to n_p and the grid itself (the mapping is injective since it is unique, and surjective since it covers all the possible grid points). Therefore, this two representations are isomorphic and we can justify the usage of the method. With this, we have N coordinates for each value in $\{1, ..., n_p\}$, and so the radius of each one was calculated. *Remark*: each coordinate was shifted by $(\kappa - 1)/2$, as this shifting is required to have the center of the sphere be located at $(0, 0, ..., 0)$. Then, the magnitude (squared) of the radius vector 'r' was calculated by the standard euclidean norm $j_1^2 + ... + j_N^2$ for each j. If the radius squared corresponding to a j was less than $((\kappa - 1)/2)^2$ (the radius of the sphere squared), then this point was considered as inside the sphere, and thus was considered by adding 1 to a variable denominated as 'count'. The end result was the final value of 'count' divided by n_p, as this is the total number of of points inside the sphere in relation to the total number of points in the cube. This volume was scaled by 2^N, as the radius was $\kappa/2$ instead of κ.
Remark: This function only works if $n_p = k^N$ for some $k \in N$

1.2 Monte Carlo Method

For the random sampling, a function was created for arbitrary dimensions N and an arbitrary number of samples n_p. For each sample point, N random values between -1 and 1 were created and then their squares were summed. This resulted in the radius squared of the sample point, which added 1 to count if it was less than 1 and 0 else. Then, count/n_p was returned (no scaling needed for this function as radius of the sphere was 1).

1.3 Analytic Recursive Method

To implement this function, a standard recursive method was applied with the initial parameter given being the dimension N. Two stopping cases were defined: $V_0 = 1$ and $V_1 = 2$. Then, the recursive step called this function again, giving it it $N - 2$ as its parameter and multiplying by $\frac{2\pi}{N}$.

2 Comparison

2.1 Rectangular Error

For this section, the difference between the analytic recursive function and the rectangular approximation was computed for different orders of magnitude, and different dimensions. Then, the error was plotted in a log-log scale in order to analyze the power rule for the error. This resulted in Fig. 1. If we let $\eta \coloneqq |y_{\text{rect}} - y_{\text{analyt}}|$, we can notice that the equations of the curve are linear and have the form of:

$$\log(\eta) = m \log(n_p) + b \implies \eta = \tilde{b} \cdot n_p^m$$

This is a decreasing exponential equation, for the error η in dependence of the the number of points n_p, which is what we expected from the theory. We can also see that, as the number of dimension increases, the exponential becomes less 'steep', which also agrees with the literature.

2.2 Monte Carlo variance

For this section, Monte Carlo method was applied for different orders of magnitude of n_p and different dimensions. The method was performed 20 times for each fixed parameter, and then the sample variance $\tilde{\sigma}^2$ of each set of parameters was computed using in-built functions. The results are shown in Fig. 2.

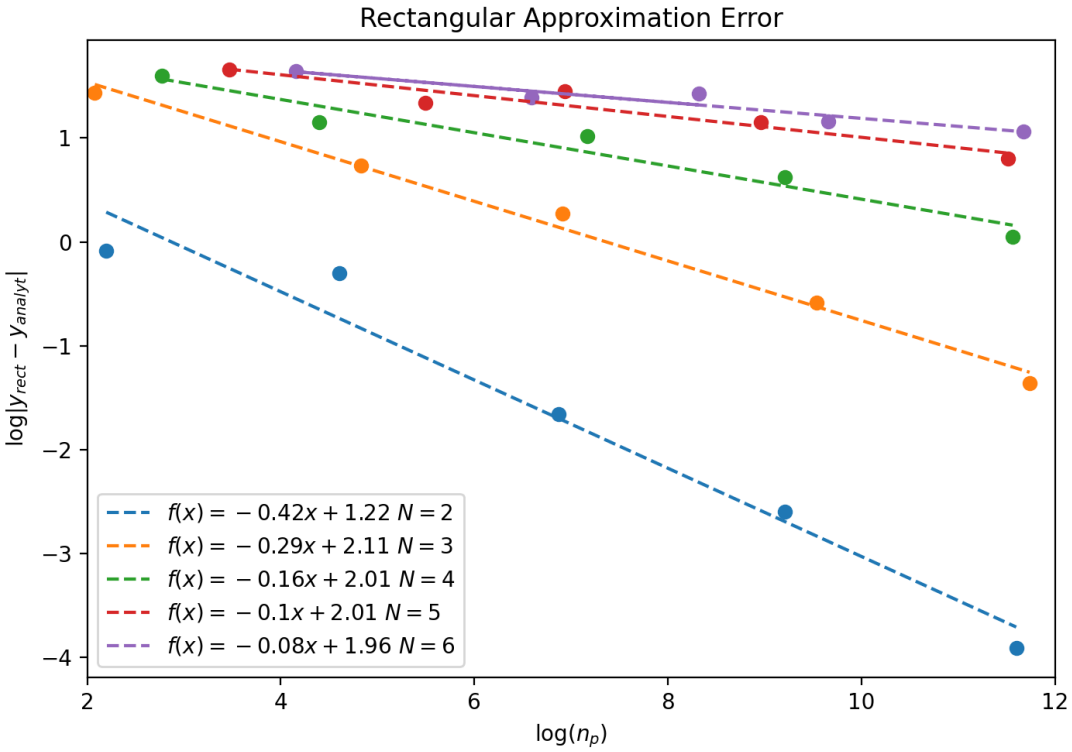

Figure 1: Rectangular error vs n_p for $N \in \{2, ..., 6\}$

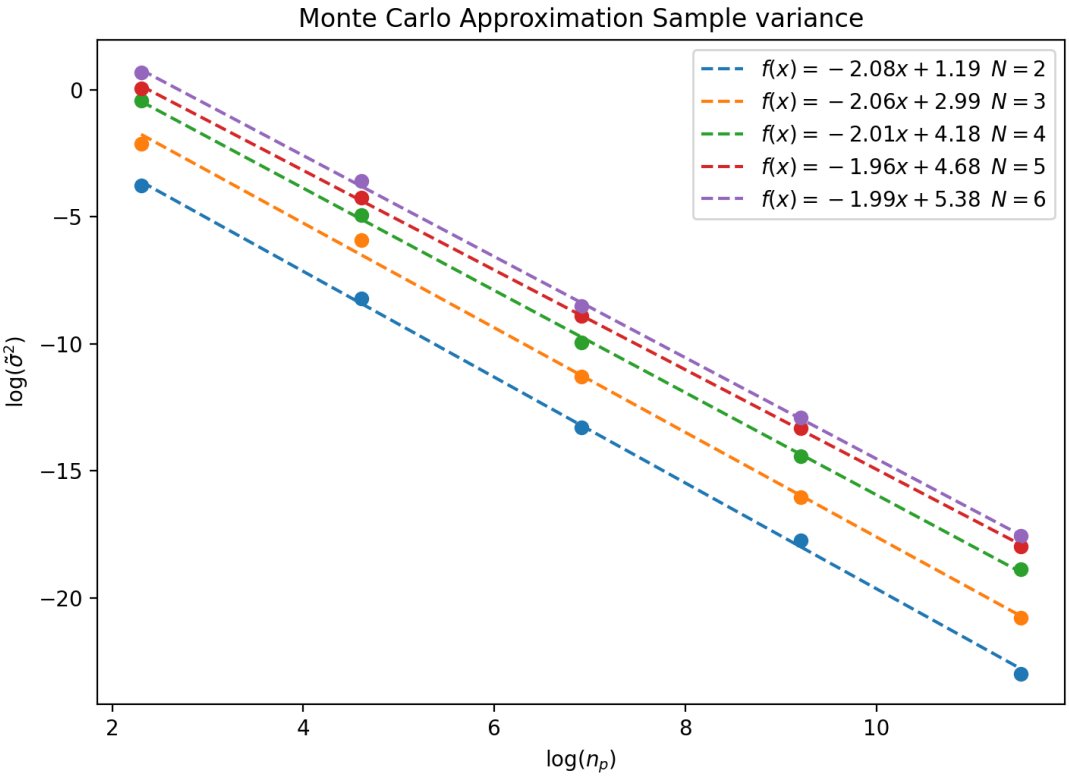

Figure 2: Variance for Monte Carlo method vs n_p for $N \in \{2, ..., 6\}$

We can notice than, in contrast to the rectangular method, this approach yields a slope that is nearly independent of the dimension N being used. This agrees with theory, as it was stated that the numerical precission should be independent of the number of dimensions for random sampling (and although we can see a slight decrease of the slope for increasing N, it is almost negligible in comparison to the rectangular method). In general, a good estimate for large-n_p would be to use Monte-Carlo for $N > 2$, as the slope clearly outperforms the one from the rectangular approach. Overall, the results obtained in this project are in much agreement with literature, which advocated for the validity of the code.