Computational Physics II — Project 3

Jan O. Haerter

March 4, 2022

Hand-in date: 6 a.m., Friday, Mar 18, 2022

In total, **100 points** can be achieved by solving the problems below (see annotation).

Volume of a hypersphere

Monte Carlo integration is a powerful tool when dealing with high-dimensional integrals. The following project compares numerical integration using grid-like discretization and Monte Carlo (random) sampling to achieve approximations to the volume of hyperspheres of various dimensions. The volume of a unit sphere in N dimensions is defined as

$$V_N \equiv \int_{\mathbb{R}^N} dx_1 dx_2 \cdot \cdots \cdot dx_N \, \theta(1 - \mathbf{r}^2) \,, \tag{1}$$

where θ is the Heaviside step function, defined as

$$\theta(x) \equiv \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \,, \end{cases} \tag{2}$$

and $\mathbf{r} \equiv (x_1, x_2, \ldots, x_N)$ is the N-dimensional position vector.
Please implement several functions that each compute the volume of the hypersphere in N dimensions. Assume only that the hypersphere is contained within a hypercube, that is, you could constrain the integration volume to the ranges $[0, 1]^N$ (using the symmetry of the sphere) or equivalently, $[-1, 1]^N$.

1. Carry out the rectangular approximation. The corresponding Python function should take the dimension N and the total number of distinct integration points n_p as input. Note that the n_p will have to constitute a regular grid in N dimensions, thus in each dimension there will be $n_p^{1/N}$ distinct points. *Hint*: The function $\theta(x)$ is a

binary function, and the volume estimate results from assessing, which fraction of the hypercube is occupied by the hypersphere. The Python function should return the numerically-approximate volume of the hypersphere. (**20 pts**)

2. Making the same assumptions on the integration volume and using the same number of points n_p, write a function that randomly samples n_p positions $\mathbf{r}$ from the integration volume. Thus, approximate the integral V_N by a Monte Carlo integration. (**20 pts**)

3. The analytical solution to the value V_N can be shown to obey the two-term recurrence relation: $V_0 = 1$, $V_1 = 2$, $V_N = \frac{2\pi}{N} V_{N-2}$. Implement a recursive Python function that computes V_N for any N from this relation. (**10 pts**)

4. For $n_p \in \{10^1, 10^2, 10^3, 10^4, 10^5\}$ (approximate values are fine, see "Hint" below.) and $N \in \{2, \ldots, 6\}$ obtain V_N via the rectangular approximation and plot the error, i.e., the absolute value of discrepancy from the exact result as function of n_p for each value of N on a log-log plot. Please describe the results and try to fit straight lines to the curves to extract the power-law exponent for each value of N. *Hint*: To get integer values of the numbers of steps in each dimension, you may want to depart somewhat from the suggested values n_p above, e.g., for $N = 3$, choosing $n_p = 5^3$ instead of $n_p = 100$ is more useful. The main idea is just to span several orders of magnitude for n_p. (**25 pts**)

5. For each combination of N and n_p, used in the previous exercise, produce a Monte Carlo integration to estimate the value of V_N. For each combination of N and n_p repeat this Monte Carlo integration perhaps 20 times to get an ensemble of results (using distinct seeds for the RNG). From the resuluting set of estimates compute the sample variance $\tilde{\sigma}^2 \approx \frac{1}{n_p}(\langle f(x)^2 \rangle - \langle f(x) \rangle^2)$. Plot $\tilde{\sigma}^2$ vs. n_p for each value of N on a log-log plot. Again try to fit straight lines to the different curves and extract the power-law exponents. Compare with the results from the rectangular approximation and try to estimate the lowest dimension where a Monte Carlo simulation is preferable in the large-N limit. (**25 pts**)

General remarks for all Projects. You will have to (i) analyze the problem, (ii) select an algorithm (if not specified), (iii) write a Python program, (iv) run the program, (v) visualize the data numerical data, and (vi) extract an answer to the physics question from the data. Which checks did you perform to validate the code? State the results you got for these tests. For each project you will submit a short report describing the physics problem, your way of attacking it, and the results you obtained. Provide the documented Python code in such a form that we can run the code. A Jupyter Notebook including the code and report is fine but not necessary.