V. Larson
November 7, 2022

Praktikum 1
Math modeling of the atmosphere

Computing a 1D integral by simple Monte Carlo integration

Assume that you wish to integrate the function

$$f(x) = H(x)x^m, \tag{1}$$

where $H(x)$ is the Heaviside step function, over a normal probability density function with zero mean:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2\right]. \tag{2}$$

Here, σ is the standard deviation. In other words, we wish to do the integral

$$I = \frac{1}{\sqrt{2\pi}\sigma} \int_{x=0}^{x=\infty} x^m \exp\left[-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2\right] dx. \tag{3}$$

We'll solve the integral in two ways: 1) analytically, and 2) using Monte Carlo integration.

1. Solve the integral analytically in terms of the gamma function. Use the fact that

   $$\int_{y=0}^{y=\infty} y^n \exp\left[-\frac{1}{2}\frac{y}{\sigma^2}\right] dx = \Gamma(n+1)\left(2\sigma^2\right)^{n+1}, \tag{4}$$

   where Γ is the gamma function.

2. Draw N random points from a uniform distribution spanning (0,1). Then transform those points to a univariate standard normal distribution, $\mathcal{N}(0,1)$, by using the inverse cumulative distribution function of a standard normal. To do so in python, you can use, e.g., the `norm.ppf` function in the `scipy.stats` library.

3. Transform the N points from the standard normal distribution to a normal distribution with mean 0 and standard deviation σ.

4. Write a python function that computes $f(x)$.

5. Compute the integral I using simple Monte Carlo integration with N sample points. Call the estimator $\hat{I}_N$.

6. Write a python function that computes the root-mean-square error in the integration,

$$\mathrm{RMSE}_N = \sqrt{\mathbb{E}((\hat{I}_N - I)^2)} \tag{5}$$

   To do this, re-compute $\hat{I}_N$ 1024/N times, with a different random seed each time.

7. Write a python driver function that inputs the number of samples, N, and outputs RMSE_N.

8. Run the driver function for the following values of N: 4, 16, 64, 256, 1024.

9. Use matplotlib to write a function that plots RMSE_N as a function of N on log-log axes. Does the convergence rate go as $1/\sqrt{N}$?