Euler-Maruyama Methods

Caleb Kilonzi

September 2024

Report

The provided Python code focuses on simulating a Geometric Brownian Motion (GBM) process and analyzing the error between the exact solution of this process and the numerical approximation given by the Euler-Maruyama method. The goal of this analysis is to evaluate how the absolute error behaves as the number of time steps used in the approximation increases, and to explore the relationship between the size of the time step and the accuracy of the Euler-Maruyama method through a log-log plot.

The code begins by defining the initial parameters for the simulation. The initial value of the process is set to $X_0 = 1.0$, the drift coefficient $\mu = 0.06$ represents the expected rate of return, and the volatility $\sigma = 0.2$ represents the randomness or fluctuation in the process. The time horizon $T = 1.0$ specifies that the simulation runs for a single unit of time. To explore how the error behaves with different time steps, the code uses a range of values for the number of time steps (num_steps), ranging from 10 to 1000.

To compute the exact solution of the Geometric Brownian Motion, the code defines the `exact_solution` function. This function uses the closed-form solution of the stochastic differential equation governing GBM. The solution is based on the well-known formula for GBM: $X_T = X_0 \exp((\mu - 0.5\sigma^2)T + \sigma W_T)$, where W_T is the Wiener process (random normal variable representing the noise in the system). This function computes the drift term $\mu - 0.5\sigma^2$ and adds a stochastic term that introduces randomness based on the normal distribution. The exact solution serves as the benchmark against which the Euler-Maruyama approximation will be compared.

The Euler-Maruyama method is a numerical technique used to approximate solutions to stochastic differential equations. In this case, the code defines the `euler_maruyama` function to approximate the GBM process. This method divides the time horizon T into discrete steps, and at each step, the value of the process is updated using both a deterministic drift term and a stochastic term that introduces noise. For each time step, the process is incremented by $\mu X_t \Delta t$ (representing the drift) and $\sigma X_t \Delta W_t$ (representing the noise from the Wiener process). The function calculates this update iteratively for the entire number of time steps, returning the final approximation of the GBM at time T.

The main purpose of the code is to investigate how the error between the exact solution and

the Euler-Maruyama approximation changes with different time step sizes. For each value in `num_steps_values`, the exact solution is computed, and the Euler-Maruyama method is used to compute an approximation. The absolute difference between the two values is recorded as the error. This process is repeated for different values of `num_steps`, which controls the size of each time step. Smaller time steps (larger values of `num_steps`) are expected to yield more accurate approximations, reducing the error.

Once the errors are computed for different time steps, the code proceeds to analyze the relationship between the size of the time step Δt and the error. It does this by performing a log-log transformation. Specifically, the logarithm of the time step size Δt is computed (`log_Delta`), as well as the logarithm of the errors (`log_errors`). This transformation is useful for analyzing power-law relationships, which are common in numerical methods where the error is often proportional to some power of the time step size.

The code then uses linear regression to fit a line to the log-log data. The slope of this line represents the rate at which the error decreases as the time step size decreases. In numerical methods, this rate is often referred to as the convergence rate. The linear regression is performed using the `np.polyfit` function, which fits a straight line to the log-transformed data and returns the slope and intercept of the best-fit line.

Finally, the code generates a plot to visually represent the relationship between the time step size and the error. The plot is a log-log plot, where the x-axis represents $\ln(\Delta t)$ (the logarithm of the time step size) and the y-axis represents $\ln(error)$ (the logarithm of the absolute error). The computed errors are shown as blue points, while the red dashed line represents the best-fit line from the linear regression. This plot helps illustrate how the error decreases as the time step size decreases. A steep negative slope would indicate that the error decreases rapidly with smaller time steps, while a flatter slope suggests that the error decreases more slowly.

Based on the provided image of the plot, the slope of the fitted line is approximately -0.10812, indicating that the error decreases very slowly as the time step size decreases. Ideally, for methods like Euler-Maruyama, we would expect a stronger dependence of the error on the time step size, with a slope closer to -1. The small slope in this case suggests that reducing the time step size leads to only modest improvements in accuracy, meaning the Euler-Maruyama method is not converging very quickly for this particular problem setup.

In conclusion, the code successfully demonstrates the relationship between the time step size and the error in the Euler-Maruyama approximation of the Geometric Brownian Motion process. However, the analysis shows that for the given parameters, the convergence is slow, and reducing the time step size alone may not be sufficient to significantly improve accuracy. This suggests that other numerical methods or optimizations may be necessary to achieve more accurate results when simulating GBM processes.