Jacobs University
Fall 2022

October 18, 2022

Stochastic Methods Lab

Assignment Sheet 5

Due on October 25, 2022

Note: The work is to be submitted via `git`, as discussed in class. The coding language is Python. Please make sure that your code actually runs and produces the requested output. Please make your code readable for the instructor and TA, and include comments wherever necessary. Please submit .py source code, not jupyter notebooks. Theoretical questions may be submitted as a scan of handwritten notes or typed up (e.g., using LaTeX). The submission deadline is midnight of the stated due date.

Problem 1 [6 points]

The price of a European Call option with current stock price S, strike price K, annualized volatility σ, annual risk-free interest rate r, and maturity time T can be computed explicitly with the Black-Scholes formula

$$C = S\,\Phi(x) - K\,\mathrm{e}^{-rT}\,\Phi(x - \sigma\sqrt{T})\,,$$

where

$$x = \frac{\ln(S/K) + (r + \sigma^2/2)\,T}{\sigma\sqrt{T}},$$

and Φ denotes the cumulative distribution function of the standard normal distribution with mean zero and variance one. Compare your call option prices from the binomial tree model with n steps from Assignment Sheet 3 against those computed from the Black-Scholes formula. Plot the logarithm of the error vs. n (loglog-plot). Do you roughly obtain a straight line? If so, with what power of n does the error scale?

Choose parameters $S = 1$, $K = 1.2$, $\sigma = 0.5$, $T = 1$, and $r = 0.03$ (for which the option price is 0.1410).

Hint: With "from scipy.stats import norm" you can use the cumulative normal distribution function "norm.cdf(x)".

Problem 2 [4 points]

Let us investigate the Stirling approximation numerically, i.e., consider

$$f(n) = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

to approximate $n! \approx f(n)$. Now do a logarithmic plot of the relative error $\frac{n!-f(n)}{f(n)}$. Do you obtain a straight line? If so, what is the slope? From that deduce what the next order in the Stirling approximation is.

Hint: With "from scipy import special" you can use "special.factorial(n)".

Problem 3 [4 points]

Plot the binomial distribution

$$b(j, n; p) = \binom{n}{j} p^j \, q^{n-j},$$

where $q = 1 - p$, into a coordinate system where the values on the x-axis correspond to j according to

$$x_j = \frac{j - np}{\sqrt{npq}}$$

and the y-values are given by $\sqrt{npq}\, b(j, n; p)$. Compare the graphs for $n = 10$, $n = 100$, and the graph of the standard Gaussian

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \, \mathrm{e}^{-x^2/2}$$

in the same plot. Do one plot with $p = 0.2$ and another one with $p = 0.5$. Comment briefly on what you see.

Hint: With "from scipy import special" you can generate the binomial coefficients with "special.binom(n,j)".

Problem 4 [4 points]

Generate $N = 10\,000$ samples of the binomial distribution (number of successes in n independent trials). Rescale the samples via

$$X = \frac{J - \mathbb{E}[J]}{\sqrt{\mathrm{Var}[J]}}$$

where you use the sample mean to approximate $\mathbb{E}[J]$ and the sample variance to approximate $\sqrt{\mathrm{Var}[J]}$. (These can be computed via the `numpy`-functions `mean()` and `std()`.) Then generate $N = 10\,000$ samples of the standard normal distribution. Plot the sorted samples for X vs. the sorted samples for the standard normal distribution. Comment briefly on what you see.

Note: This is called a QQ-plot and is more generally used to empirically compare two probability distributions.

Hint: Take a look at the functions "binomial", "normal", and "sort".