Finite Difference Approach for the Greeks

20247 Applied Numerical Finance

Group 7 – Brage Bakken, Dominik Meyer, Enrico Giannelli, Emanuele Chiarini, Mikhail Borovkov

1. Problem Description

This project deals with the parameter sensitivity of European options through forward (FDE) and central difference estimators (CDE). We implement the two methods and look at their performance in practice.

2. Forward/Central Difference Estimators

The two critical estimators used in this work are forward and central difference estimators, both finite difference estimators. The finite difference approach is a mathematical method that can be used to estimate the derivatives of a function (the discounted price of the option in our case) at a specific point simply by computing values of the function with it inputs slightly changed. More in detail, we have the Forward difference estimator (FDE):

$$\Delta_F \equiv \Delta_F(n,\, h) = \frac{\overline{Y_n}(\theta+h) - \overline{Y_n}(\theta)}{h}.$$

where $\overline{Y_n}(.)$ is the sample average of n present values of the option payoff.

The central difference estimator (CDE) is instead:

$$\Delta_C \equiv \Delta_C(n,\, h) = \frac{\overline{Y_n}(\theta+h) - \overline{Y_n}(\theta-h)}{2h}.$$

Calculating the bias, with the assumption that the present value of the option is twice differentiable, yields the following for the FDE:

$$Bias(\Delta_F) = \frac{1}{2}\alpha''(\theta)h + o(h), \text{ where } \alpha(\theta) = E[Y(\theta)].$$

Whereas doing the same on the CDE yields a smaller bias:

$$Bias(\Delta_C) = o(h).$$

and the result is even stronger if we assume the existence of a third derivative.

However, we note that the CDE is more costly as we need to compute two sample means of the discounted prices (at $\theta + h$ and $\theta - h$) instead of the one we need in the FDE's case. We ignore the computation of the sample mean of the actual present value of the option as that is usually computed regardless of this procedure

3. Implementation

The estimation was done in Python using Jupyter Notebook. We implemented the following functions:

- generate_path: generates Monte Carlo estimates of the price of a European option
- expected_price_option: computes the expected price of an option using the Black Sholes model formula
- compute_greek: computes the value of a specified Greek using the formulas in Table 1.
- estimate_greeks: computes the correct value of the Greek, using compute_greek, and the estimated one applying the finite difference approach (FDE/CDE) to either the exact price from expected_price or the simulated one from generate_path.
- variance_estimation: computes an estimation of the variance of a finite difference estimator.
- plot_error: plots the exact bias of the finite difference estimator
- plot_greek_mc: plots the greek estimations of the Monte Carlo simulation and the closed form solution
- plot_variance_greeks: plots the variance of the Greeks' Monte Carlo simulation

Greek	Closed-form formula (call)	Closed-form formula (put)
Delta	$\Delta_C = e^{-qT} \cdot N(d_1)$	$\Delta_P = N(d_1) - 1$
Gamma	$\Gamma_{C\ or\ P} = \frac{\partial \Delta_C}{\partial S} = e^{-qT} \cdot \frac{N'(d_1)}{S\sigma\sqrt{T}}$	
Theta	$\theta_C = \frac{\sigma S e^{-qT} N'(d_1)}{2\sqrt{T}} - -qSe^{-qT}N(d_1) + r\,e^{-rT}N(d_2)$	$\theta_P = \frac{\sigma S e^{-qT} N'(d_1)}{2\sqrt{T}} + + qSe^{-qT}N(d_1) - re^{-rT}N(d_2)$
Vega	$\upsilon_{C\ or\ P} = Se^{-qT}N'(d_1)\sqrt{T}$	
Rho	$\rho_C = T\,e^{-rT}N(d_2)$	$\rho_P = -T\,e^{-rT}N(-d_2)$

Table 1. BS closed-form formulas for the Greeks.

All the functions are fully documented and commented. We also developed a couple of other auxiliary functions handling the actual plots creation and the bias computation, that are not explained here as they are not expected to be accessed directly by users.

4. **Programming challenges**

The biggest challenge we faced in using the finite difference approach was to ensure that the calculation was efficient and fast, even when using very large values of n (such as 10000 or more). To address this challenge, we had to rewrite the code for the finite difference approach using NumPy. NumPy allowed us to handle very large values of n without sacrificing speed or accuracy, thus, greatly improving our ability to do interesting simulations as shown afterwards.

In order to further improve the efficiency and performance of the code we tried to avoid redundancies by reusing as much of the code as possible. This involved identifying common patterns and operations in the code, and then combining or refactoring them to reduce the overall number of lines of code and the amount of duplication.

In addition, the calculation of gamma required specific modifications to the code compared to the calculation of the other Greeks, which made it difficult to integrate the two calculations. To address this challenge, we had to carefully design the code for the gamma calculation so that it could be easily integrated into the rest of the calculation process.

To solve the issue of code redundancy we resorted to the creation of parameters dictionaries. Each parameters dictionary stored the input parameters provided by user, such as the underlying asset's price, volatility, and time to expiration. The dictionaries then easily allowed us to update, with +*h*/-*h*/…, the values of the correct input parameter depending on the Greek that was being calculated, thus, simplifying the code and making it more readable and maintainable.

5. Findings

We used the following parameters in the estimations that follow: Spot price S=100, Strike price K=100, Volatility σ=30%, Time to expiry T=1, Interest rate r=1%, Dividend yield q=0%. Note that whenever variances are plotted, we're estimating them using 200 estimators. In addition, whenever n and h are not varied they're set to 10000 and 0.01 respectively

5.1. Discretization error

The first result of our analysis are graphs like Figure 1 illustrating the bias of this approach, by evaluating the difference between the closed-form formula of the Greeks and the first difference approach applied to the present values of the options obtained with closed formulae.

5.2. The role of h

We then looked at the role of h. We find that as the value of h increases, the derivative estimate becomes less accurate. This is because a larger value of h results in a coarser approximation of the derivative, thus causing the estimates to consistently deviate from the true value of the derivative. Using a very small value of h, instead, can lead to more accurate estimates of the derivative, but it can also increase the amount of random error or uncertainty in the estimates especially if n is small. These effects are clearly showcased by the Figure 2 and 3.

5.3. The role of n

Next, we find that the larger the value of n, the more accurate the derivative estimate will be. However, using a very large value of n will also increase the computational cost. In general, the choice of n is a trade-off between accuracy and computational cost. Again, Figure 4 clearly shows this. Ideally, in practice we would choose an n that is large enough and that doesn't increase the computational cost too much and we would then choose h so as to have a variance and bias that are as low as possible.

5.4. FDE vs CDE

We can see in Figure 5 and 6 that the central difference estimator is more accurate than the forward difference estimator, as we discussed before. We have to keep in mind, though, that the CDE requires us to perform more calculations.

5.5. The effect of using the same seed

If we look at Figure 7 and 8 it becomes obvious, when focusing in on the y-axis, that the variance of the sampled Greeks using the same seed for $\Upsilon(\theta + h)$ and $\Upsilon(\theta - h)$ is way lower than the one when we generate them from different sets of random numbers. This is reasonable as we're approximating a derivative using a difference of present values and in the case in which the present values are estimated using different seeds we would have that a portion of the difference would be caused by these different realizations and not just by the change in the parameter.

6. Numerical/Financial challenges

Interestingly we found that the value of the h parameter needs to be adapted to the specific Greek being estimated. This is because, as we just discussed, the h value should be small enough to accurately approximate the derivative of the parameter with respect to the underlying inputs, but not so small that the approximation is dominated by numerical errors. This means that for example, the h value may need to be larger when estimating the sensitivity of an option's price with respect to the underlying stock price than when we estimate it with respect to the risk-free interest rate. This is obviously caused the stock price typically having a larger value then the risk-free interest rate and, thus, the perturbation needing to be bigger to be meaningful in this case.

7. Optimization

Given the results we just found, in particular on the role of h, we decided to implement an optimizer that minimizes the MSE based on it. To do so we relied on SciPy minimize function and its COBYLA method. COBYLA works by iteratively constructing a quadratic approximation of the objective function (in this case, the mean squared error) and solving the resulting quadratic optimization problem to determine the next set of inputs to try. To do so it uses a trust region approach to explore the search space and find the set of inputs that minimize the mean squared error.

The result is the optimizer function present in the Jupyter notebook which once provided with the option's parameters and the largest n that leads to reasonable computational times on the user's machine provides the value of h that minimizes the MSE.

8. Next Steps

The presented framework for Greek estimation can be expanded in several ways that are beyond the scope of the project. An example of this would be to allow for the simulation of the price path, the paths of interest rate (r), dividend yield (q), and volatility (σ). This would allow us to consider the computation of Greeks, through the finite difference approach, for more general cases.

9. References

Glasserman, P. (2003). Monte Carlo Methods in Financial Engineering. Springer-Verlag New York.

10. Pictures

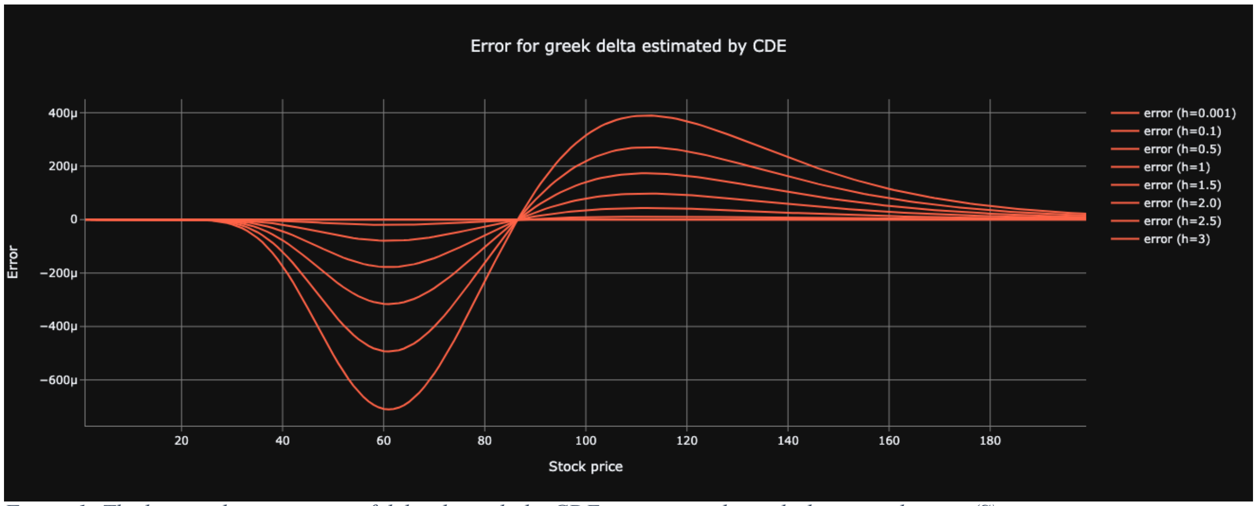

Figure 1: The bias in the estimation of delta through the CDE as we vary the underlying stock price (S)

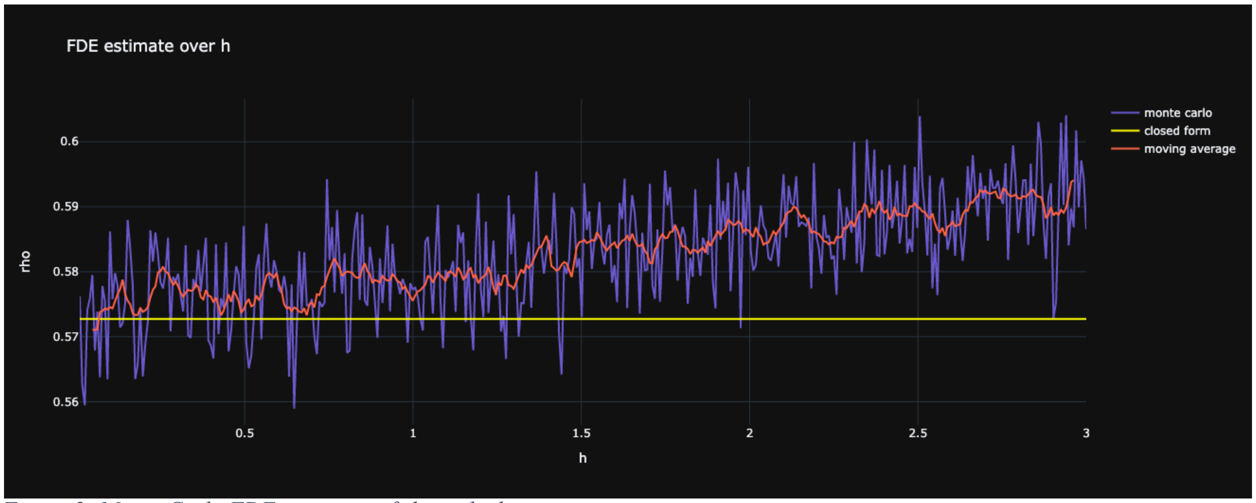

Figure 2: Monte Carlo FDE estimates of rho as h changes

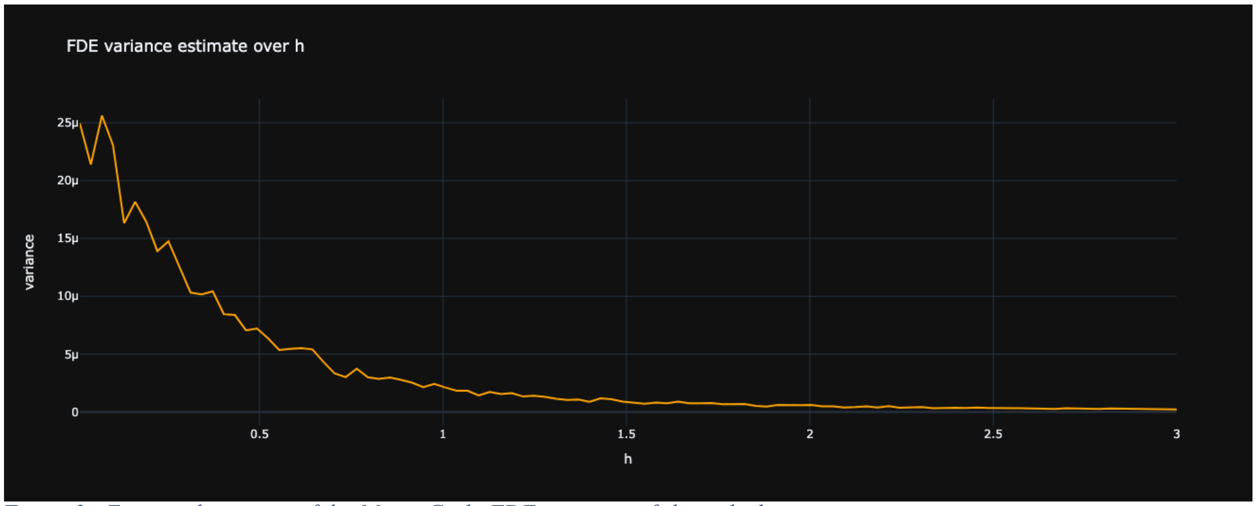

Figure 3: Estimated variance of the Monte Carlo FDE estimates of rho as h changes

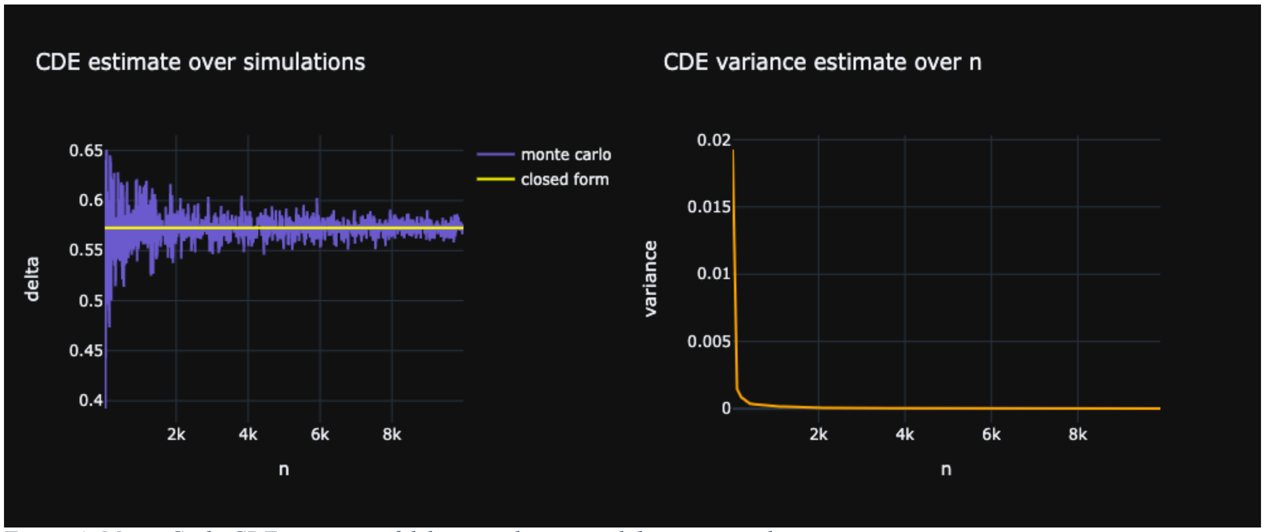

Figure 4: Monte Carlo CDE estimates of delta as n changes and their empirical variance

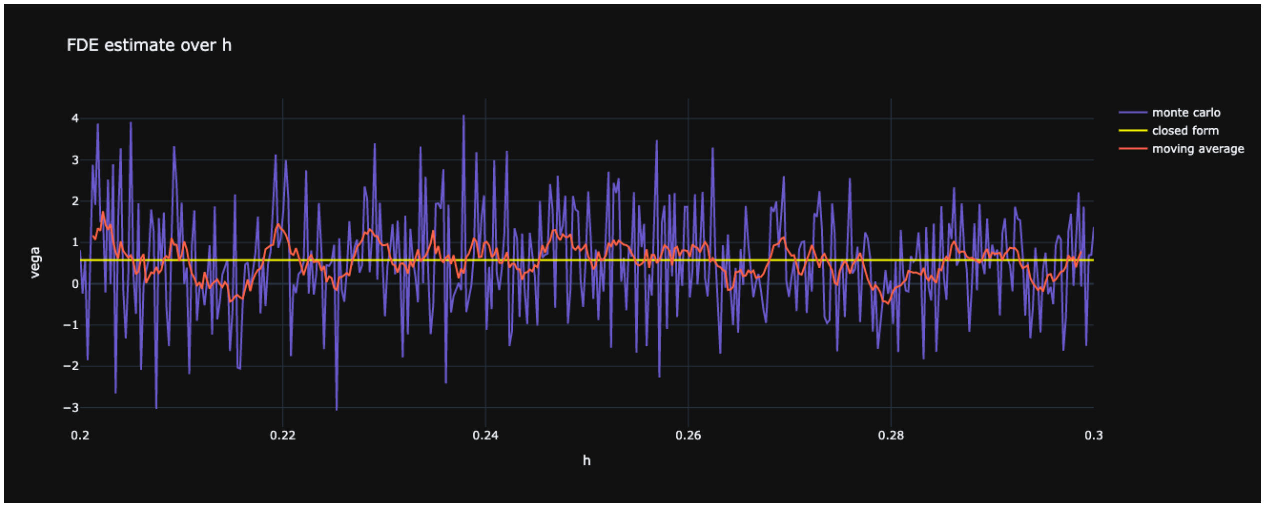

Figure 5: Monte Carlo FDE estimates of vega as h varies

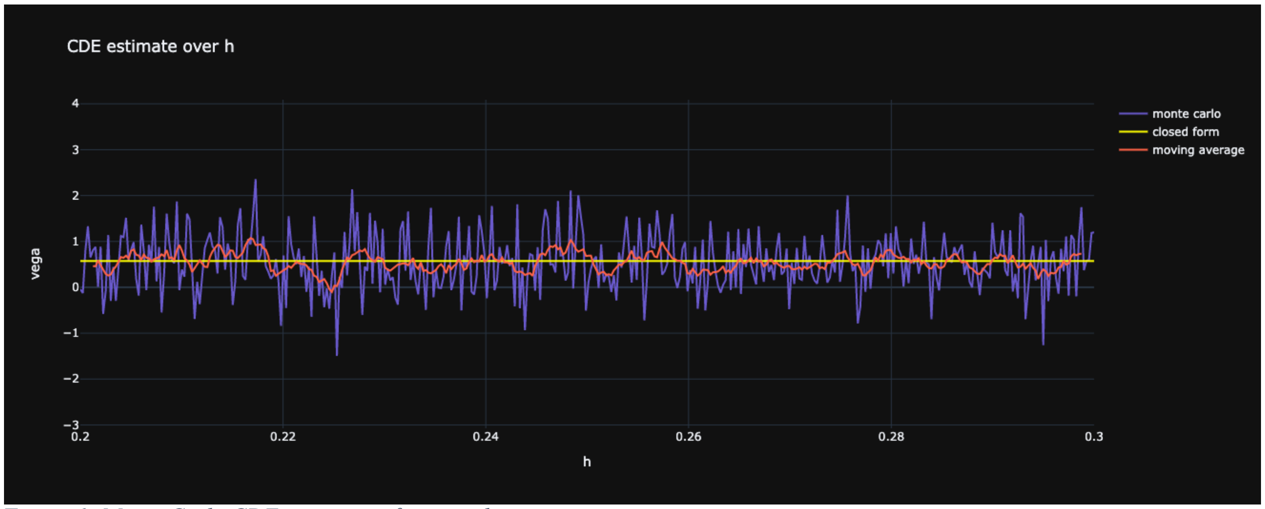

Figure 6: Monte Carlo CDE estimates of vega as h varies

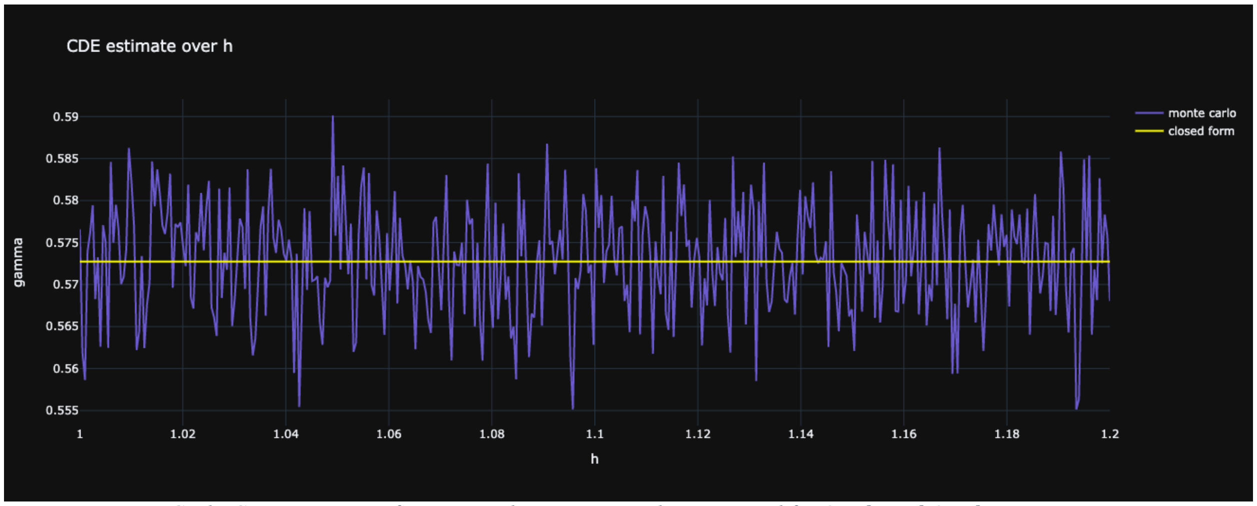

Figure 7: Monte Carlo CDE estimates of gamma as h varies using the same seed for $\theta + h$ and $\theta - h$

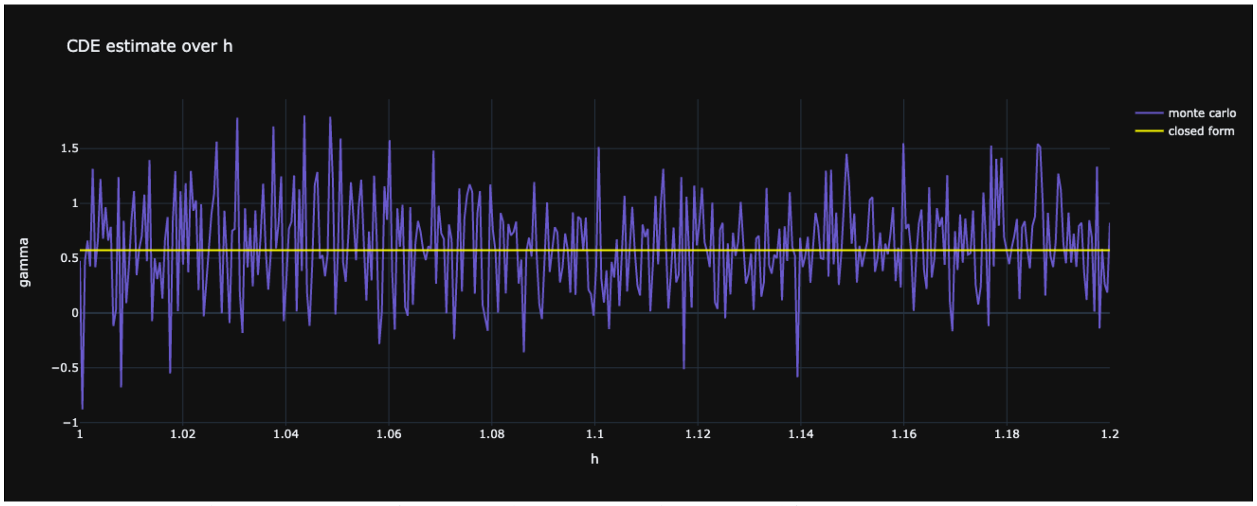

Figure 8: Monte Carlo CDE estimates of gamma as h varies using the same seed for $\theta + h$ and $\theta - h$