From Brownian Paths to the Heat Equation

Two Numerical Routes to Black–Scholes

Dorian Przetakiewicz

06. February 2026

Abstract

This project prices vanilla and exotic European options in two independent ways: a Monte Carlo simulation based on the Feynman–Kac representation, and a numerical solution of the Black–Scholes equation cast as a heat equation via an operator-splitting scheme. Both approaches are cross-checked against known analytical formulae. The Monte Carlo section covers European, Asian, binary and lookback options. The diffusion section presents the reduction to the heat equation, a Strang-split time evolution, and a comparison to the analytical call surface both for a short-maturity market option and for a longer-maturity benchmark.

Contents

1 Monte Carlo Simulation of the Black–Scholes Formula

In order to do a Monte Carlo simulation with the goal to calculate option prices, we make use of the so-called Feynman-Kac formula. The Black Scholes formula is given in 31. The Feynman Kac formula applies to PDE's (like the B-S) of type,

$$\frac{\partial C}{\partial S}a(S,t) + \frac{1}{2}\frac{\partial^2 C}{\partial S^2}b(S,t)^2 + \frac{\partial C}{\partial t} - r(S,t)C = 0 \tag{1}$$

with final condition $C(S,T) = C(S)$ and $dS = a(S(t),t)dt + b(S(t),T)d\tilde{W}$. Applying that to the Black-Scholes formula, we receive:

$$C(0) \equiv C(S,0) = \langle e^{-rT}C(S(T))|S(0) = S_0\rangle_{\tilde{\mathcal{P}}} \tag{2}$$

where S_0 is the current share price and $\tilde{\mathcal{P}}$ the martingale probability measure, under $C_{B-S} = \langle PV(C(T))\rangle_{\tilde{\mathcal{P}}}$.
From this one could derive the formula to calculate the current (european) option price like,

$$C(0) = \int_0^\infty dS e^{-rT}C(S,T)\frac{1}{S\sqrt{2\pi\sigma^2 T}}e^{-\frac{\left[ln(\frac{S}{S_0})-(r-\frac{1}{2}\sigma^2 T)\right]^2}{2\sigma^2 T}} \tag{3}$$

meaning that for european options the payoff function $V(S,T)$ depends only on the share price $S(T)$ and we can calculate this numerically.
In the case of european options, put and call, the payoffs do not depend on the share price trajectories, thus inside a Monte Carlo simulation, we do not have to do x time-steps, but can immediately calculate the sample $S(t+T)$ instead of x steps á $S(t+\Delta t)$.
We assume the price of the stock to behave like Geometric Brownian motion with drift r and volatility σ:

$$S(t+\Delta t) = S(t)e^{(r-\frac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\xi}, \quad \xi \propto \mathcal{N}(0,1) \tag{4}$$

Using this assumption and formula, we can also price exotic options like asian or lookback options.
So to realize the Monte Carlo simulation, we start at $t = 0$ with $S(0) = S_0$ then we split the time T into *steps* (an integer) of size Δt. That way we generate N_{MC} trajectories $\{S(t)\}$ in the range $0 \le t \le T$. For each trajectory we compute the payoff, according to the options payoff functions as defined later, and finally we estimate:

$$\langle e^{-rT}C(S(T))|S(0) = S_0\rangle_{\tilde{\mathcal{P}}} \approx \frac{1}{N_{MC}}\sum_{i=1}^{N_{MC}} e^{-rT}C(S(T))_i \tag{5}$$

In each step we generate a random number ξ. This random number is generated using the Mersenne Twister algorithm (`std::mt19937_64`) under a normal distribution with $\mu = 0$, $\sigma = 1$.

The way i implemented this in C++ is, i created a class called `BlackScholes` which holds the following members:

`N`	$= N_{MC}$, number of Monte Carlo trajectories
`stepSize`	number of Δt time-steps per trajectory
`type`	option type (asian, binary, european, lookback)
`X`, `r`, `sig`, `T`	strike, risk-free rate, volatility σ, maturity
`S0`	initial stock price S_0
`S`	array of stock prices along the current trajectory
`payoffs`	array of payoffs, one entry per trajectory
`discounted`	discounted payoffs, $e^{-rT} \cdot$ `payoffs`
`dt`	$= \Delta t$, single sub-step size
`lastPrice`	pointer to the last entry of `S`

The member `type` selects the payoff function F, which differs both by option type (asian, binary, european, lookback) and by put/call. `S` stores the stock prices of the current trajectory and is used inside path-dependent payoff functions (e.g. the Asian options). For the European and binary options it is sufficient to use `lastPrice`, a pointer to the last entry of `S`. Finally `payoffs` and `discounted` store the (undiscounted and discounted) payoffs of each trajectory, from which we recover today's option price $C(S, 0)$. Finally we take the mean of all discounted option prices to calculate an estimate of the true option price:

$$C(0) \approx \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} e^{-rT} C(S(T))_i \tag{6}$$

with the standard error:

$$\sigma = \sum_{i=1}^{N_{MC}} \sqrt{\frac{(e^{-rT} C(S(T))_i - C(0))^2}{n(n-1)}} \tag{7}$$

1.1 Option Types

In my Monte Carlo simulation i simulate the following option types and compare them with their analytical solution.
European Call: Right to buy at strike X.
Payoff function:

$$F(S, T) = \max(S_T - X, 0) \tag{8}$$

Analytical function:

$$C_{\text{Euro}}(S, T) = S_0 N(d_1) - X e^{-rT} N(d_2) \tag{9}$$

with

$$d_1 = \frac{\ln(S_0/X) + \left(r + \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T} \tag{10}$$

and (N is the CDF of the standard gaussian):

$$N(x) = \Phi(x) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \tag{11}$$

European Put: Right to sell at strike X.
Payoff function:

$$F(S, t) = \max(X - S_T, 0) \tag{12}$$

Analytical function:

$$P_{\text{Euro}}(S, T) = X e^{-rT} N(-d_2) - S_0 N(-d_1) \tag{13}$$

Asian Call (arithmetic average): Depends on average stock price over the path.
Payoff function:

$$F(S, T) = \max(\bar{S} - X, 0), \quad where(\bar{S} = \frac{1}{n} \sum_{i=1}^{n} S_{t_i}). \tag{14}$$

Analytical function (taken from [2]):

$$C_{Asian}(S_0, T) = \exp\left[((\rho - r)T)\right] \left(S_0 \Phi(\tilde{d1}) - X \exp\left[(-\rho T)\Phi(\tilde{d_2})\right)\right] \tag{15}$$

where (m is equal to the number of steps per trajectory)

$$\sigma_z = \sigma\sqrt{\frac{2m+1}{6(m+1)}} \quad \rho = \frac{r - \frac{\sigma^2}{2} + \sigma_Z^2}{2} \tag{16}$$

and

$$\tilde{d_1} = \frac{\log \frac{S_0}{X} + (\rho + \frac{1}{2}\sigma_z^2)T}{\sqrt{T}\sigma_Z}, \quad \tilde{d_2} = \frac{\log \frac{S_0}{X} + (\rho - \frac{1}{2}\sigma_z^2)T}{\sqrt{T}\sigma_Z} \tag{17}$$

Asian Put (arithmetic average):
Payoff function:

$$F(S,T) = \max(X - \bar{S}, 0), \quad where(\bar{S} = \frac{1}{n}\sum_{i=1}^{n} S_{t_i}). \tag{18}$$

Analytical function:

$$P_{asian}(S_0,T) = \exp\left[(\rho - r)T\right]\left(X \exp\left[-\rho T\right]\Phi(-\tilde{d_2}) - S_0\Phi(-\tilde{d_1})\right) \tag{19}$$

Binary Call: Fixed cash if in-the-money.
Payoff function:

$$F(S,T) = \Theta(S_T - X) \tag{20}$$

Analytical function:

$$C_{\text{bin}}(S_0,T) = e^{-rT}\Phi(d_1 - \sigma\sqrt{T}) \tag{21}$$

Binary Put:
Payoff function:

$$F(S,T) = \Theta(X - S_T) \tag{22}$$

Analytical function:

$$P_{\text{bin}}(S_0,T) = e^{-rT}\Phi(-d_2) \tag{23}$$

Lookback Call (floating strike): Payoff is the difference between price at maturity and the minimum value of the asset price over the duration of the option lifetime; formulae taken from [3].

Payoff function:

$$F(S,T) = \max(S_T - S_{min}, 0) \tag{24}$$

Analytical function:

$$C_{\text{LB}}(S_0,T) = S_0\Phi(a_1(S,m)) - me^{-rT}\Phi(a_2(S,m)) - \frac{S_0\sigma^2}{2r}\left(\Phi(-a_1(S,m)) - e^{-rT}\left(\frac{m}{S_0}\right)^{\frac{2r}{\sigma^2}}\Phi(-a_3(S,m))\right) \tag{25}$$

with:

$$M = \max_{0\le\tau\le T} S_\tau, \quad m = \min_{0\le\tau\le T} S_\tau \tag{26}$$

where:

$$a_1(S,H) = \frac{\ln(S/H) + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \tag{27}$$

$$a_2(S,H) = a_1(S,H) - \sigma\sqrt{T}, \qquad a_3(S,H) = a_1(S,H) - \frac{2r\sqrt{T}}{\sigma} \tag{28}$$

Lookback Put (floating put):
Payoff function:

$$F(S,T) = \max(S_{max} - S_T, 0) \tag{29}$$

Analytical function:

$$P_{\text{LB}} = -S_0\Phi(-a_1(S,M)) + Me^{-rT}N(-a_2(S,M)) + \frac{S_0\sigma^2}{2r}\left(\Phi(a_1(S,M)) - e^{-rT}\left(\frac{M}{S_0}\right)^{\frac{2r}{\sigma^2}}N(a_3(S,M)\right) \tag{30}$$

1.2 Results

In the following i will present the results of the Monte Carlo simulations, compared with their analytical values.
In all simulations, except the lookback option simulations and the european call option simulation we use $N_{MC} = 100000$ and $n_{steps} = 250$ Δt time steps per trajectory.
In Fig. 1 we see that the trajectory shows strong spikes in the beginning, this is because the

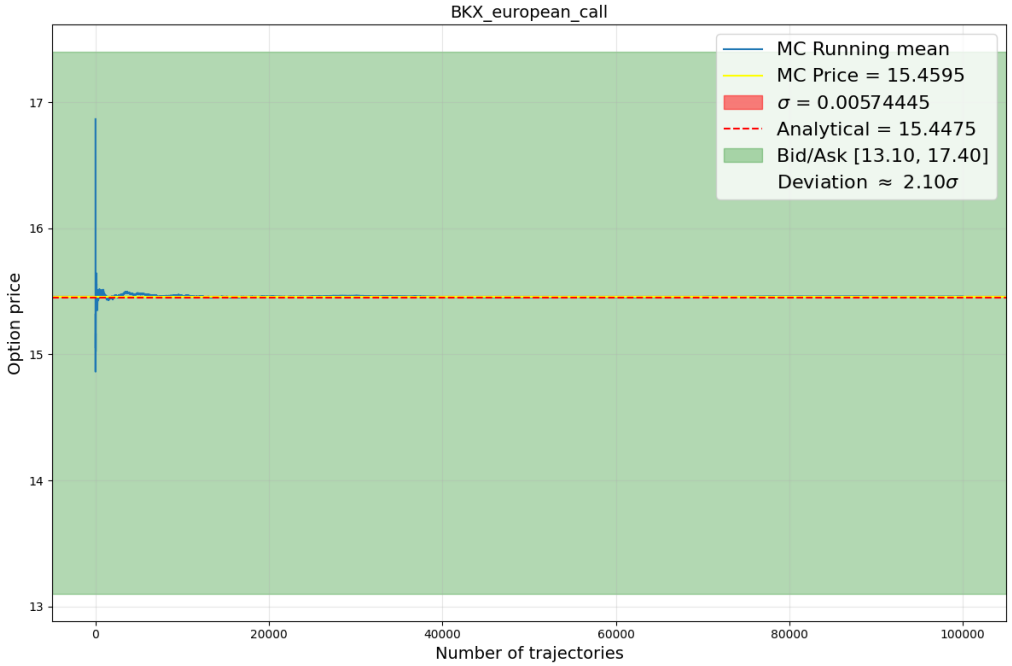

Figure 1. BKX European Call option with maturity date 2013-03-16, quoted on 2013-01-02. Bid: 13.1, Ask: 17.4; $N_{MC} = 10000$, $S_0 = 52.93$, $X = 37.5$, $r = 0.01$, $\sigma = 0.1586$, $T = 17/365$; data from [1].

running mean has less values to be computed on and thus shows a higher variance. In the next plots i will remove the first 20% of data points to showcase the motion of the option price depending on the number of trajectories with a more stable running mean.
We see that the simulation returns an option price that is within the price regime (ask-bid) of the actual option, and has a deviation of 0.94σ to the analytical price. In Fig. 2 we see that for the given european put option we get a sensible price from the monte carlo simulation within 1.54σ of the estimated analytical solution, and well within the quoted bid/ask price of that day.

We realise that the Monte Carlo simulations seem to provide reasonable results; the results from both lookback options are poor, as one can see by the high deviation (in σ). However, [5] reports that floating-strike lookback options do perform poorly in Monte Carlo simulations, with similar deviations from the analytical value. One interesting fact is that the LookBack simulations seemed to perform better if the step size was increased, the bigger the better it seems.

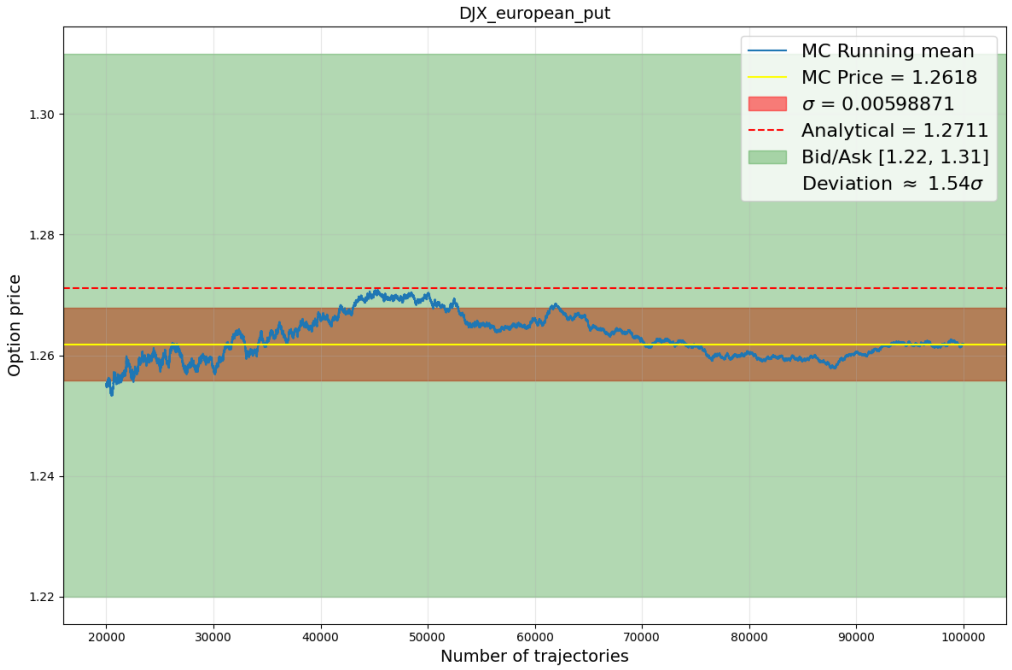

Figure 2. DJX European put option with maturity date 2013-01-19, quoted on 2013-01-02; $S_0 = 134.13$, $X = 134.0$, $r = 0.01$, $\sigma = 0.1183$, $T = 17/365$; data from [1].

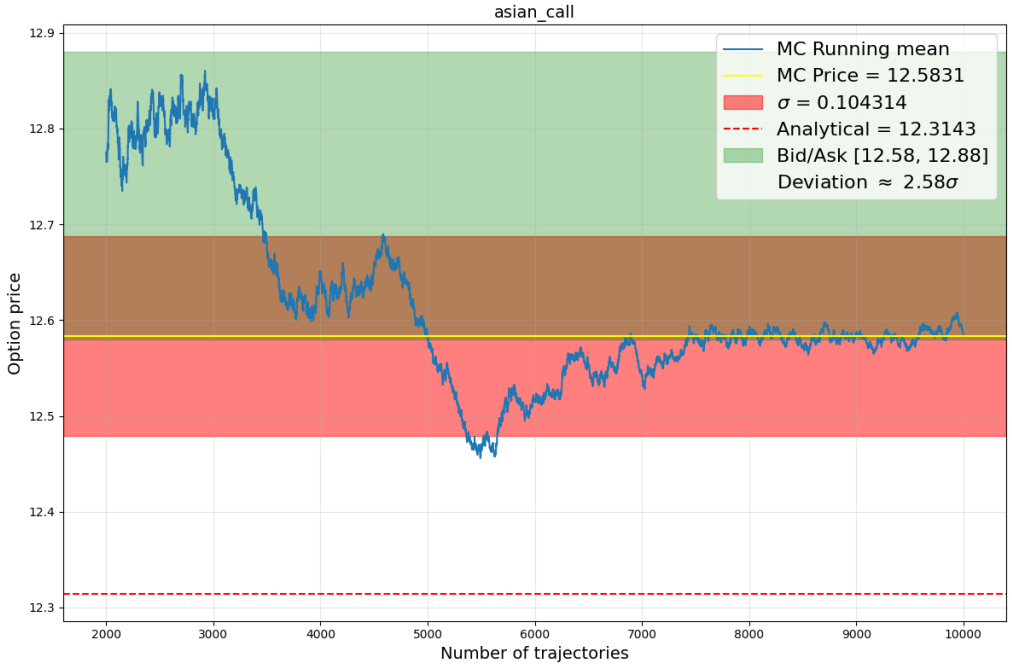

Figure 3. Asian call option; fictive bid/ask values are the max/min from the Monte Carlo reference in [2]; $S_0 = 100.0$, $X = 0.90$, $r = 0.05$, $\sigma = 0.20$, $T = 1$.

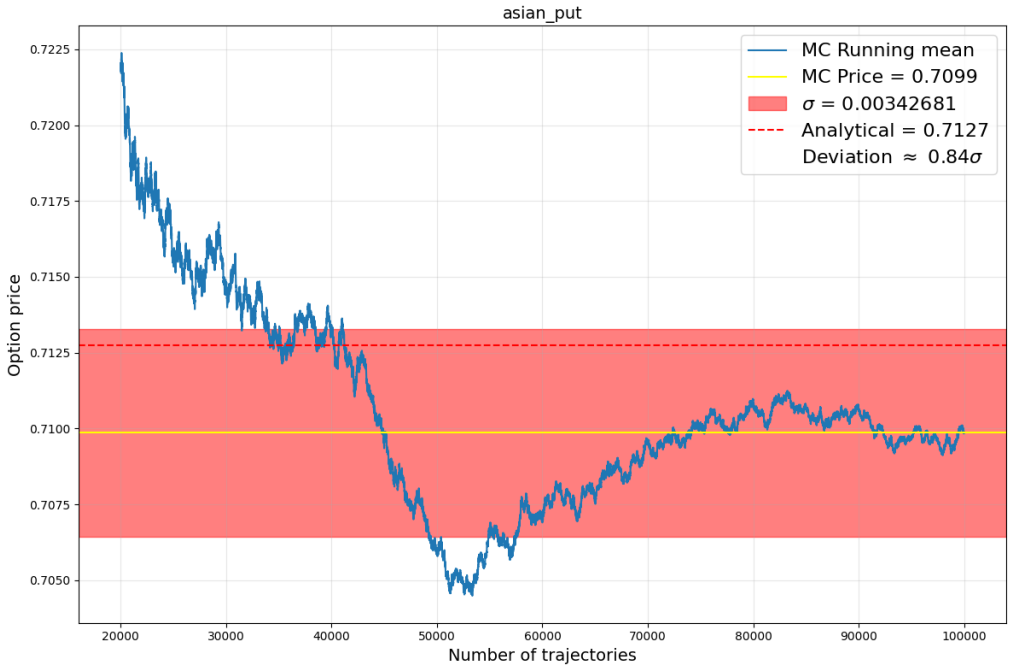

Figure 4. Asian put option, with parameters being the same ones as for the European Put but calculated as an asian option (i could not find free data), $S_0 = 134.13$, $X = 134.0$, $r = 0.01$, $\sigma = 0.1183$, $T = \frac{17}{365}$

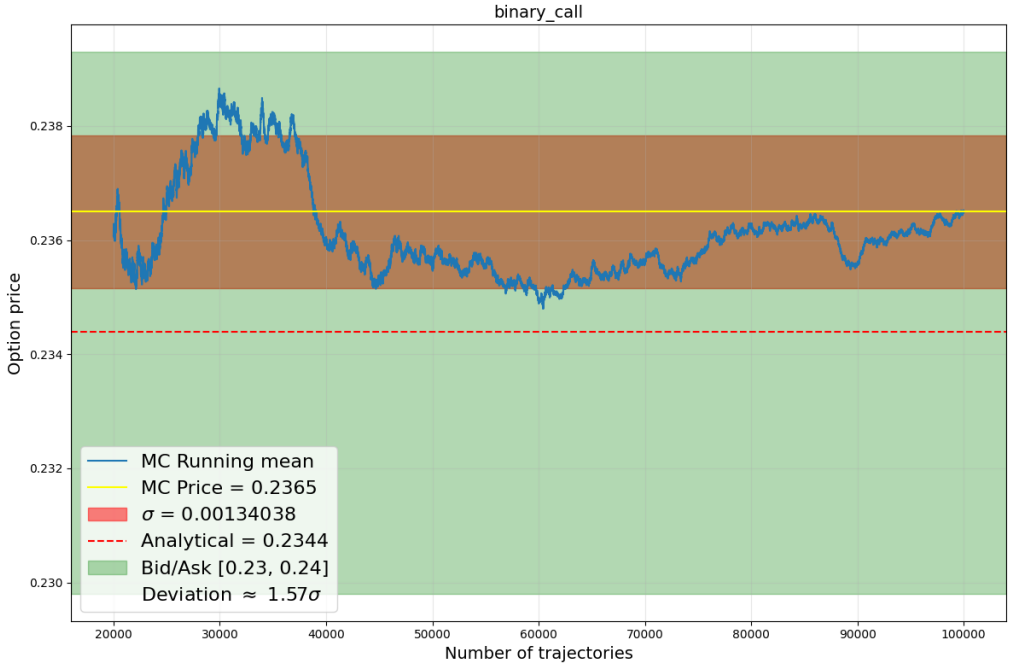

Figure 5. Binary call option; parameters taken from [4], whose result is $0.23458 \pm 2\%$ (indicated as the fictive bid-ask range). $S_0 = 121.675$, $X = 150$, $r = 0.0156$, $\sigma = 0.7860$, $T = 90/365$; risk-free-rate is the 5-year treasury bill yield.

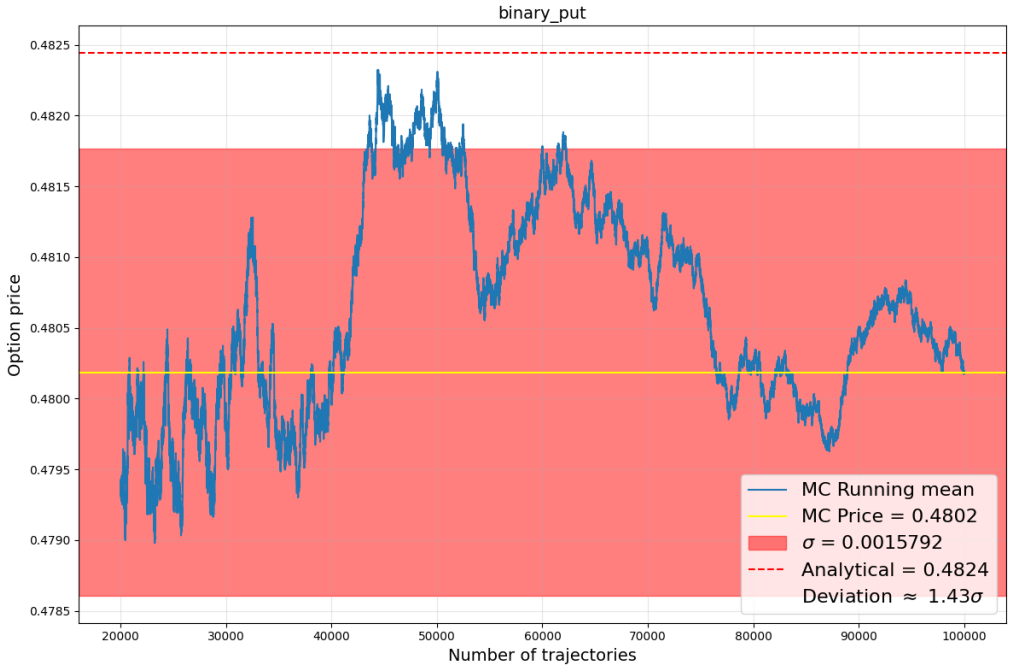

Figure 6. Binary put option, with parameters being the same ones as for the European Put but calculated as a binary put option (i could not find free data), $S_0 = 134.13$, $X = 134.0$, $r = 0.01$, $\sigma = 0.1183$, $T = \frac{17}{365}$

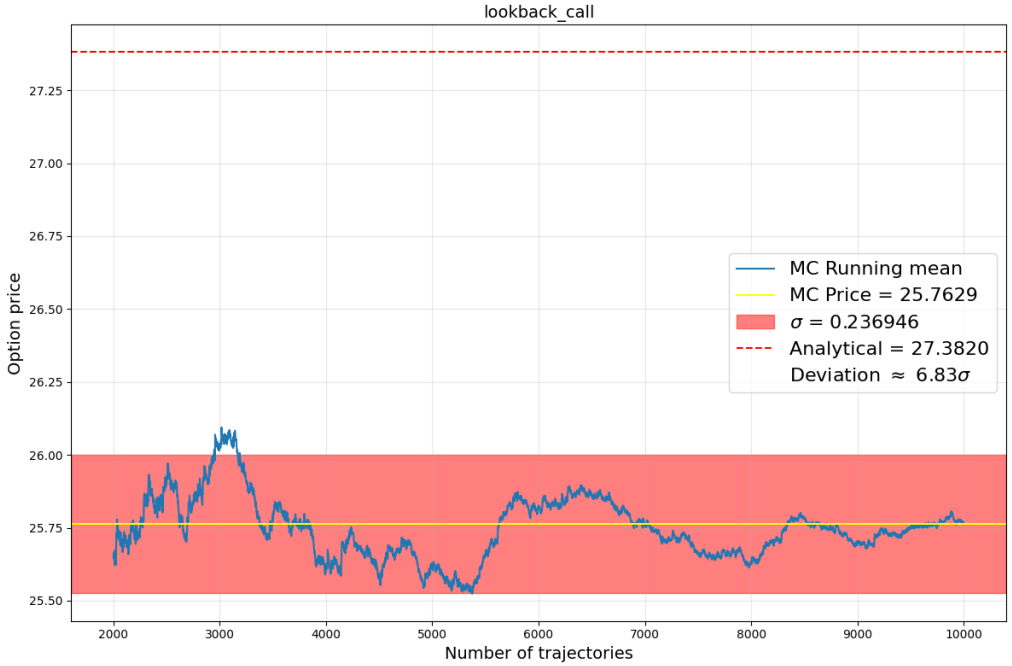

Figure 7. Lookback call option with parameters from [3]; the analytical formula assumes a minimum price 10% below S_0. $S_0 = 100$, $r = 0.1$, $\sigma = 0.3$, $T = 1$, $N_{MC} = 10000$, $n_{steps} = 10000$.

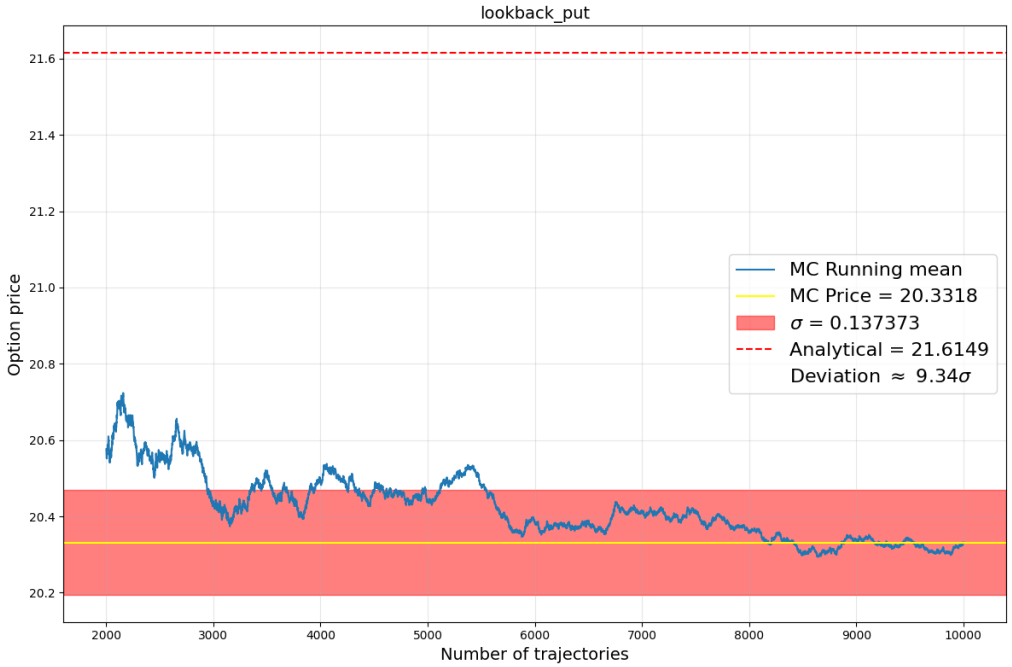

Figure 8. Lookback put option with the same parameters as the above call option, the analytical equation assumes a max price of 10% higher than S_0, $N_{MC} = 10000$, $n_{steps} = 10000$

2 Black–Scholes as a Diffusion Equation: Derivation

We have the Black-Scholes equation:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0 \tag{31}$$

with the following boundary conditions:

$$C(S,T) = \max(S - X, 0) \tag{1}$$

$$C(0,t) = 0, \qquad \lim_{S\to\infty} C(S,t) \propto S. \tag{32}$$

We rewrite Eq. 31 in the following way:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 (S\frac{\partial}{\partial S})^2 C + (r - \frac{1}{2}\sigma^2) S\frac{\partial C}{\partial S} - rC = 0 \tag{33}$$

We introduce the following change of variables:

$$S = e^y \qquad t = T - \tau \tag{34}$$

This change of variables is motivated by the fact, that the underlying process described by S is a geometric Brownian motion (that's how the stock price behaves). That way $y = \log S$ describes a Brownian motion (with a possible drift).
Secondly the evolution of the system is backwards from the terminal state of the system, thus we have to reverse time.
This results in

$$S\frac{\partial}{\partial S} \to \frac{\partial}{\partial y}, \qquad \frac{\partial}{\partial t} \to -\frac{\partial}{\partial \tau} \tag{35}$$

giving us a new form of Eq. 33:

$$\frac{\partial C}{\partial \tau} - \frac{1}{2}\sigma^2 \frac{\partial^2 C}{\partial y^2} - (r - \frac{1}{2}\sigma^2)\frac{\partial C}{\partial y} + rC = 0 \tag{36}$$

Now we see that if we were to get rid of the last two terms, we would have a diffusion equation. Next we discount the option price C with (remember $\tau \propto -t$)

$$u(y,\tau) = e^{r\tau} C(y,\tau) \tag{37}$$

This substitution then leads (due to the chain rule) $\frac{\partial e^{-r\tau} u}{\partial \tau} = -re^{-r\tau}u + e^{-r\tau}\frac{\partial u}{\partial \tau}$ and helps us to get rid of the previous rC term. We receive the following equation:

$$\frac{\partial u}{\partial \tau} - \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial y^2} - (r - \frac{1}{2}\sigma^2)\frac{\partial u}{\partial y} = 0 \tag{38}$$

Now we are left with a single partial derivative with respect to y. This one does not cancel yet, unless $r = \frac{1}{2}\sigma^2$ because we have not taken into account the drift of the Brownian motion. The drift is linear in time τ, thus we can make the following substitution:

$$x = y + (r - \frac{1}{2}\sigma^2)\tau, \qquad \tau = t \tag{39}$$

We then receive an equation of form:

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x^2} \tag{40}$$

The function $u(x,\tau)$ is defined for $-\infty < x < \infty$ and $0 \leq \tau \leq T$. The terminal condition Eq. 1 of the original Black-Scholes problem becomes the initial condition of the heat equation at $\tau = 0$. The substitutions we did so far ($u = e^{r\tau}C$ and $x = y + (r - \frac{1}{2}\sigma^2)\tau$) reduce at $\tau = 0$ to the identity ($u = C$, $x = y = \log S$), so the initial condition is simply the call payoff written in the new variable x:

$$u(x,0) = u_0(x) = \max(e^x - X,\, 0). \tag{41}$$

This is just the standard $\max(S - X, 0)$ payoff, evaluated at $S = e^x$.
The solution of the equation is given by the known solution of the heat equation:

$$u(x,\tau) = \frac{1}{\sigma\sqrt{2\pi\tau}} \int_{-\infty}^{\infty} u_0(s) e^{-\frac{(x-s)^2}{2\sigma^2\tau}} ds \tag{42}$$

We can evaluate this integral and then return to the former variables $(x, \tau, u) \to (S, t, C)$ via

$$C(S,t) = e^{-r\tau}\, u\big(\log S + (r - \tfrac{1}{2}\sigma^2)\tau,\ \tau\big), \qquad \tau = T - t. \tag{43}$$

3 Numerical Approach for the Diffusion Equation

Now we want to solve the diffusion equation numerically, for the Black Scholes model.
We choose a spatial domain $x \in [x_{min}, x_{max}]$ and discretize it into L grid points with spacing Δx.
Let

$$u_i(\tau) = u(x_i, \tau), \qquad i = 1, \cdots, L \tag{44}$$

so that essentially it becomes a vector like

$$\vec{u}(\tau) = \begin{pmatrix} u_1(\tau) \\ \vdots \\ u_L(\tau) \end{pmatrix} \tag{45}$$

The time evolution now happens by updating this vector within the numerical simulation.
Looking at Eq. 40, we realize that we have a second derivative, we use the standard central difference to rewrite it as:

$$\frac{\partial^2 u(x,\tau)}{\partial x^2} \approx \frac{\frac{u_{i+1}-u_i}{\Delta x} - \frac{u_i - u_{i-1}}{\Delta x}}{\Delta x} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} \tag{46}$$

We can go further by writing out the full diffusion equation:

$$\frac{\partial u(x,\tau)}{\partial \tau} = D \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} \tag{47}$$

where we defined $D = \frac{\sigma^2}{2}$.
Now for the full grid we can rewrite this equation using a matrix, calling it $\hat{H}$

$$\hat{H} = \begin{bmatrix} 2 & -1 & 0 & . & . & . & 0 \\ -1 & 2 & -1 & & & & . \\ 0 & -1 & 2 & & & & . \\ . & & & . & & & . \\ . & & & & . & & 0 \\ . & & & & & 2 & -1 \\ 0 & . & . & . & 0 & -1 & 2 \end{bmatrix} \tag{48}$$

to be of form:

$$\frac{\partial \vec{u}(\tau)}{\partial \tau} = -\frac{D}{(\Delta x)^2}\hat{H}\vec{u}(\tau) \tag{49}$$

The matrix $\hat{H}$ can then be decomposed into the two matrices

$$\hat{A} = \begin{bmatrix} 1 & -1 & 0 & . & . & . & 0 \\ -1 & 1 & 0 & & & & . \\ 0 & 0 & 1 & -1 & & & . \\ . & & -1 & . & & & . \\ . & & & & . & & 0 \\ . & & & & & 1 & -1 \\ 0 & . & . & . & 0 & -1 & 1 \end{bmatrix} \text{ and } \hat{B} = \begin{bmatrix} 1 & 0 & 0 & . & . & . & 0 \\ 0 & 1 & -1 & & & & . \\ 0 & -1 & 1 & & & & . \\ . & & & . & & & . \\ . & & & & . & -1 & 0 \\ . & & & & -1 & 1 & 0 \\ 0 & . & . & . & 0 & 0 & 1 \end{bmatrix}.$$

So it means now that $\hat{H} = \hat{A} + \hat{B}$ and that the solution to our diffusion Eq. 49 is then (with $\alpha = -\frac{D}{(\Delta x)^2}$):

$$\vec{u}(\tau + \Delta_\tau) = e^{\alpha \hat{H} \Delta_\tau}\vec{u}(\tau) \tag{50}$$

where Δ_τ is the size of one numerical time step.
Now since $\hat{H}$ is neither diagonal nor block-diagonal, we do not really know how the exponential looks, and also since these are matrices and not numbers, generally $e^{\hat{H}} \neq e^{\hat{A}}e^{\hat{B}}$, but using the Lie–Trotter product formula, to first order (see [6]):

$$e^{t(\hat{A}+\hat{B})} = \lim_{n\to\infty}(e^{t\hat{A}/n}e^{t\hat{B}/n})^n \tag{51}$$

we can approximate to second order the exponential of the matrices (the proof is done via some steps involving the Baker-Campbell-Hausdorff formula since the matrices don't commute) such that we get the second order product formula (Strang splitting):

$$e^{\alpha \hat{H}\,\Delta_\tau} \;\approx\; e^{\frac{1}{2}\alpha \hat{A}\,\Delta_\tau}\; e^{\alpha \hat{B}\,\Delta_\tau}\; e^{\frac{1}{2}\alpha \hat{A}\,\Delta_\tau}. \tag{52}$$

Finally the full time evolution for our system over the total interval T is then done by applying this time-stepping formula m times, with $\Delta_\tau = T/m$:

$$e^{\alpha \hat{H}\,T} \;\approx\; \left(e^{\frac{1}{2}\alpha \hat{A}\,\Delta_\tau}\; e^{\alpha \hat{B}\,\Delta_\tau}\; e^{\frac{1}{2}\alpha \hat{A}\,\Delta_\tau}\right)^m. \tag{53}$$

The diffusion equation can then be solved by 50, with the usage of 53. The matrix exponential of a block diagonal matrix, where each block is of the form $\hat{X} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$, can be easily determined by computing

$$e^{q\hat{X}} = \sum_{k=0}^{\infty}\frac{q^k}{k!}\hat{X}^k = \mathbf{1} + \frac{\hat{X}}{2}(e^{2q} - 1) \tag{54}$$

where $\mathbf{1}$ defines the identity matrix.
Practically in my simulation $\hat{A}$ and $\hat{B}$ are block-diagonal $L \times L$ matrices where the first block of $\hat{B}$ and the last block of $\hat{A}$ are 1×1 unity blocks, and the rest are 2×2 blocks of the form $\hat{X}$ above. Reading off the half-step exponent from Eq. 52, the half-step factor for $\hat{A}$ is $e^{\frac{1}{2}\alpha\Delta_\tau\hat{X}}$ and the full-step factor for $\hat{B}$ is $e^{\alpha\Delta_\tau\hat{X}}$. Defining the dimensionless number

$$\beta \;=\; \alpha\,\Delta_\tau \;=\; -\frac{D\,\Delta_\tau}{\Delta x^2}, \tag{55}$$

the per-block matrices that appear in the Strang substep are

$$e^{(\beta/2)\hat{X}} = \frac{1}{2}\begin{bmatrix} 1+e^{\beta} & 1-e^{\beta} \\ 1-e^{\beta} & 1+e^{\beta} \end{bmatrix} \quad \text{(half-step for } \hat{A}\text{)}, \tag{56}$$

$$e^{\beta\hat{X}} = \frac{1}{2}\begin{bmatrix} 1+e^{2\beta} & 1-e^{2\beta} \\ 1-e^{2\beta} & 1+e^{2\beta} \end{bmatrix} \quad \text{(full-step for } \hat{B}\text{)}, \tag{57}$$

and the lone 1×1 blocks at the corners pick up $e^{\beta/2}$ (for the last block of $\hat{A}$) and e^{β} (for the first block of $\hat{B}$), respectively. In the code these matrices are constructed once, and the Strang substep is applied m times in the time loop.
Finally in order to be able to write our simulation, we need to define the correct intial conditions with the transformed variables. From Eq. 41 we already have the initial condition in closed form,

$$u_i(0) = \max(e^{x_i} - X,\, 0), \tag{58}$$

which is just the call payoff evaluated at the grid points $S_i = e^{x_i}$. This means that each grid point will have the value $u_i(0)$ assigned to it at the beginning of the simulation. The variable x is directly related to S as it is defined as $x = \log S + (r - \frac{1}{2}\sigma^2)\tau$, which directly denies the possibility of having a stock price $S = 0$. So i chose S_{min} and S_{max} to lie well away from the strike X, with S_{min} a small fraction of X and S_{max} several multiples of X. With these values for S i calculate the max and min value for x and space them equally on a grid of $L = 1001$ points, where Δx is the difference between each point.
During the simulation we also have to make sure that the boundary conditions are correct. For the left boundary ($S \to 0$) we have

$$C(0,t) = 0 \quad \to \quad \lim_{x \to -\infty} u(x,\tau) = 0, \tag{59}$$

so $u_1(\tau) = 0$ throughout the simulation.
For the right boundary ($S \to \infty$), we know that for large S the call price behaves as $C(S,t) \sim S - Xe^{-r(T-t)}$. Plugging this into $u = e^{r\tau}C$ together with $S = e^{x-(r-\sigma^2/2)\tau}$ gives

$$u(x_{\max},\tau) \;\sim\; e^{\,x_{\max}+\frac{1}{2}\sigma^2\tau} - X. \tag{60}$$

At every time step we set $u_L(\tau)$ to this asymptote.

3.1 Results

With the initial condition Eq. (58), the Strang split Eq. (53), the β from Eq. (55) and the boundary conditions above, the numerical solver reproduces the analytical Black–Scholes price to within the spatial discretisation error of $\mathcal{O}(\Delta x^2)$.

We test it on the BKX European call from the Monte Carlo section, using the exact same parameters ($S_0 = 52.93$, $X = 37.5$, $r = 0.01$, $\sigma = 0.1586$, $T = 17/365$). Fig. 9 shows the numerical curve $C(S,0)$ from the diffusion solver overlaid on the analytical Black–Scholes price. The two curves are visually indistinguishable; at $S = S_0$ the numerical price is 15.388 versus an analytical 15.448, a relative deviation of 0.4%. This is well within the quoted bid–ask spread (13.1/17.4) for that day.

Because the BKX maturity is only $T = 17/365 \approx 0.047$, the heat-equation evolution barely has time to smooth the kink of the payoff, and the surface $C(S,t)$ is almost the payoff itself. To actually see the diffusion at work, i also ran the same solver on a textbook-style set of parameters $T = 1$, $\sigma = 0.30$, $r = 0.05$, $X = 100$, where the kink has plenty of time to smooth out. Fig. 10 shows the resulting numerical surface side-by-side with the analytical Black-Scholes surface, and

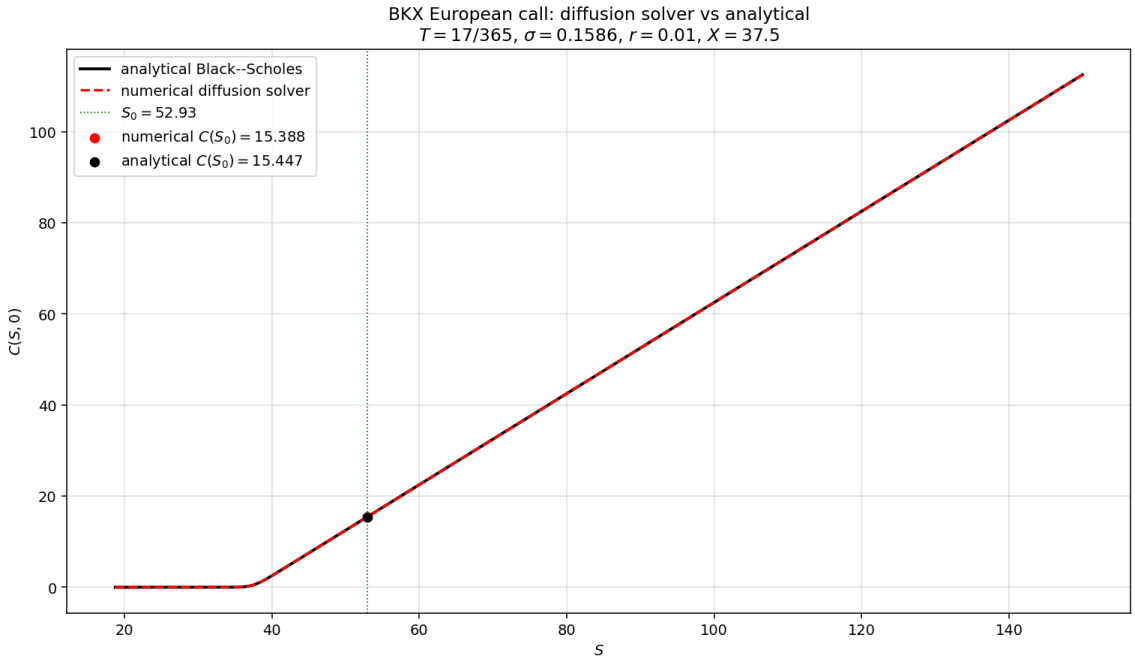

Figure 9. BKX European Call: numerical price $C(S, 0)$ from the diffusion solver (red dashed) overlaid on the analytical Black–Scholes price (black). Markers indicate the value at S_0. Bid: 13.1, Ask: 17.4; $S_0 = 52.93$, $X = 37.5$, $r = 0.01$, $\sigma = 0.1586$, $T = 17/365$; data from [1].

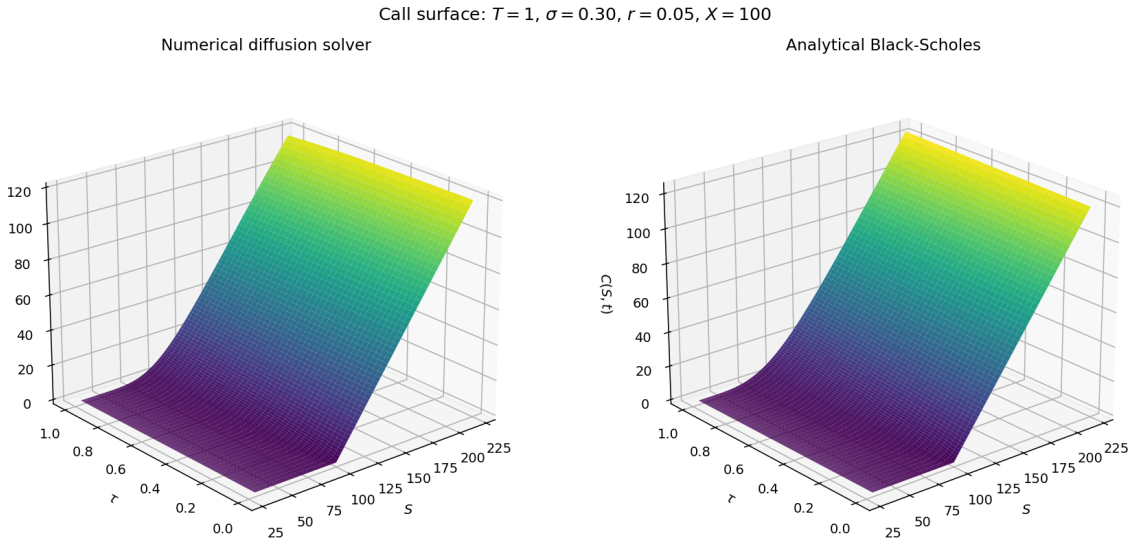

Figure 10. Call-price surface $C(S, t)$ versus S and $\tau = T - t$ from the diffusion solver (left) compared with the analytical Black-Scholes surface (right). Parameters: $T = 1$, $\sigma = 0.30$, $r = 0.05$, $X = 100$.

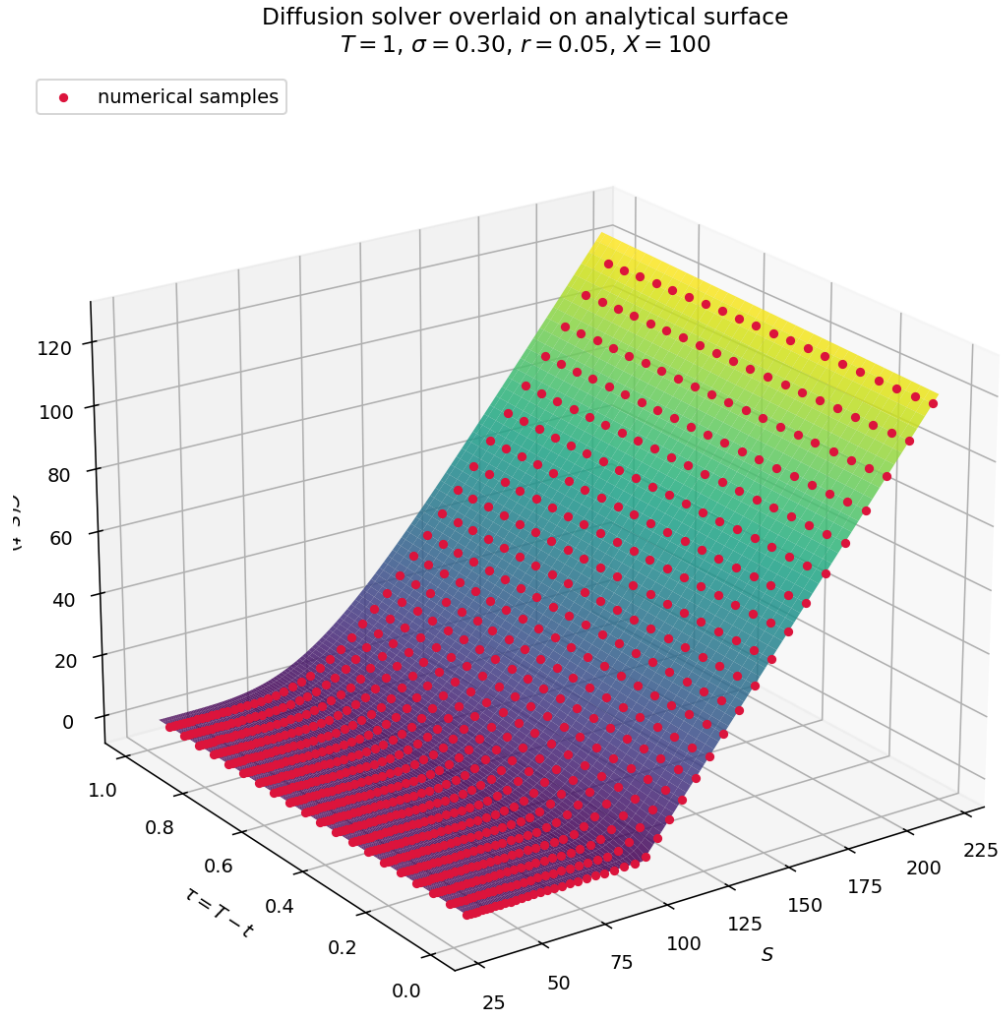

Figure 11. Same data as Fig. 10: numerical samples (red dots) overlaid on the analytical surface. The samples sit on the surface, which is what we want.

Fig. 11 overlays the numerical samples (red dots) on the analytical surface (smooth colormap) to emphasise that the two agree pointwise.

To get a quantitative feel for the agreement, Fig. 12 shows three slices at $\tau = 0$ (the payoff), $\tau = 0.25$ and $\tau = 1$ (deep into maturity). At $\tau = 0$ the numerical curve sits exactly on the kinked payoff, as it should – this is the initial condition Eq. 58 verbatim. At intermediate τ the curves agree to the eye. At $\tau = 1$ there is a small undershoot at large S; this comes from imposing the finite-$x_{\max}$ asymptote Eq. 60 on a truncated grid, and shrinks if one pushes $x_{\max}$ further out. Fig. 13 shows the pointwise error $C_{\text{num}} - C_{\text{BS}}$ on the same grid; the error is everywhere small compared to the price level and is concentrated near the right boundary at late τ, exactly where one expects it.

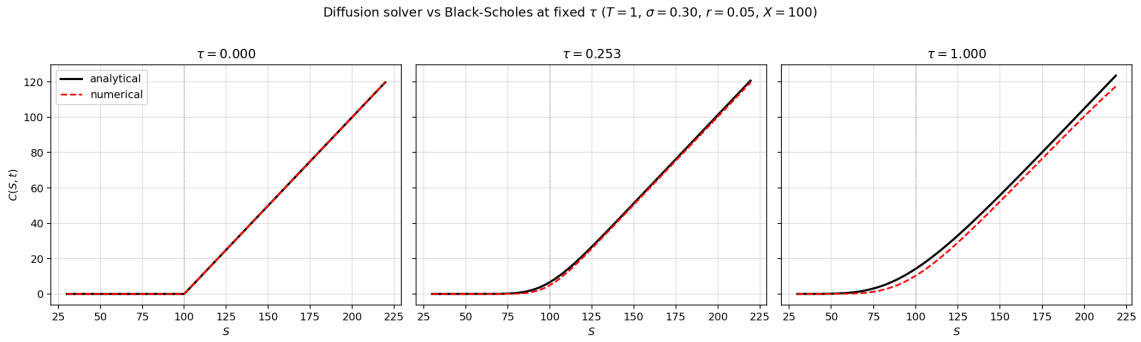

Figure 12. Slices of $C(S, t)$ at three values of τ for the $T = 1$ benchmark. Black: analytical Black-Scholes; red dashed: diffusion solver. The dotted vertical line marks $S = X$.

Overall the diffusion-equation solver agrees with the analytical Black-Scholes price across the whole (S, τ) surface within the expected $\mathcal{O}(\Delta x^2)$ discretisation error, both for the short-maturity BKX option used in the Monte Carlo section and for the longer-maturity benchmark $T = 1$.

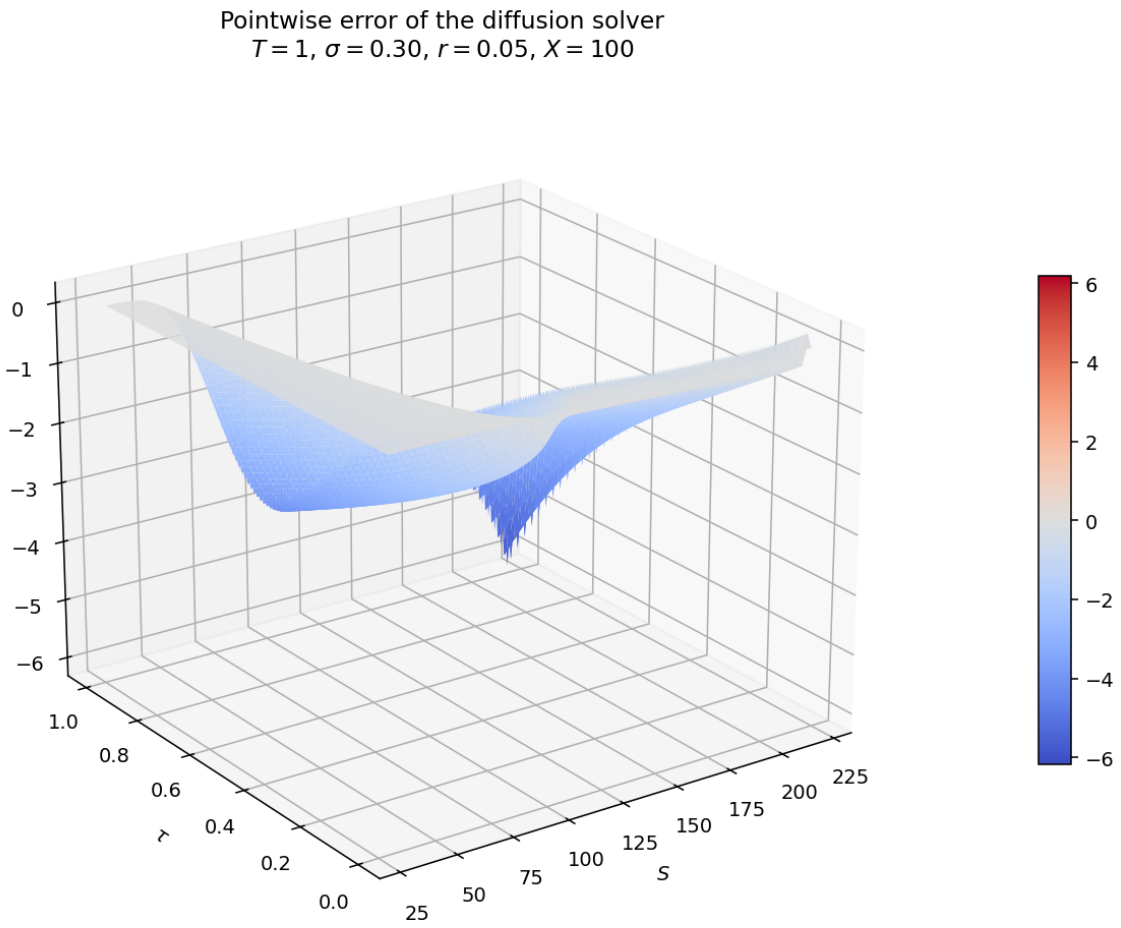

Figure 13. Pointwise error $C_{\text{num}} - C_{\text{BS}}$ on the same (S, τ) grid as Fig. 10. The error is small in magnitude and concentrated near the truncated right boundary at large τ.

AI Usage

Claude Opus (Anthropic) was used to (i) refine the writing style of this report, (ii) resolve a numerical issue in the diffusion-equation solver where the discretised time step in the Strang splitting was mis-scaled, and (iii) generate the three-dimensional comparison plots (Figs. 10, 11 and 13) as well as the slice plot in Fig. 12. The physics content, the derivations, the Monte Carlo implementation and the analytical formulae are my own; the AI tool acted as an editing and plotting assistant.

References

[1] Historical option data, https://optiondata.org.

[2] T. Aspenberg and V. Nord, *The Discrete Arithmetic Asian Option and a Comparison of Novel Methods for its Valuation*, Bachelor Thesis, Uppsala Universitet, 2010. https://www.diva-portal.org/smash/get/diva2:301070/FULLTEXT01.pdf.

[3] QuantStart, *Floating-Strike Lookback Option Pricing with C++ via Analytic Formulae*. https://www.quantstart.com/articles/Floating-Strike-Lookback-Option-Pricing-with-C-via-Analytic-Formulae.

[4] T. L. Nguyen and V. T. Pham, *On Binary Option Pricing under the Black–Scholes Model*, EAI Endorsed Transactions, 2022. https://eudl.eu/pdf/10.4108/eai.28-10-2022.2328425.

[5] S. Yadav, *Estimating Lookback Price Using Monte Carlo Simulation and Binomial Lattice*, 2021. https://www.researchgate.net/publication/351788645.

[6] Strang splitting (background reference). https://grokipedia.com/page/Strang_splitting.