Level 9 HW: Part D

Name: David Leather

Date: 4/2/2025

Part D: Advanced Monte Carlo

Question D.a

> *Create generic functions to compute the standard deviation and standard error based on the above formulae. The inputs are a vector of size `M` (`M = NSIM`), the interest-free rate and expiry time `T`. Integrate this newcode into `TestMC.cpp`. Make sure that the code compiles.*

I implemented two functions. One generic function computes the standard deviation, `cmptSD()` , and one computes the standard error, `cmptSE()`, which calls `cmptSD()` using the specified inputs. These functions are generic in that they allow for different numeric types, `NumType`.

```
// Global function for computing std. dev.
template<typename NumType>
NumType cmptSD(vector<NumType> draws, NumType r, NumType T)
{
    // This function computes the standard deviation of the Monte Carlo
    // pricing algorithm based on a vector of draws, the risk-free rate,
    // and time-to-expiry.

    // Extract M
    unsigned M = draws.size();

    // Compute sum_C := \sum_j C_{T,j}, and sum_sq_C := \sum_j C_{T,j}^2
    NumType sum_sq_draws = 0;
    NumType sum_draws = 0;
    for (unsigned i = 0; i < M; ++i)
    {
        sum_draws += draws[i];
        sum_sq_draws += draws[i] * draws[i];

    }
    return sqrt((sum_sq_draws - (pow(sum_draws, 2) / M)) / (M - 1)) * exp(-r * T);
}

// Global function for computing standard error
template<typename NumType>
NumType cmptSE(vector<NumType> draws, NumType r, NumType T)
{
    // This function computes the standard error of the Monte Carlo
    // pricing algorithm based on a vector of draws, the risk-free rate,
    // and time-to-expiry.

    // Extract M
    unsigned M = draws.size();

    // Compute standard deviation
    NumType std_dev = cmptSD(draws, r, T);

    return std_dev / sqrt(M);
}
```

Additionally I needed to change part of the main loop to first compute and assign the payoff at time T, or equivalently $C_{T,j}$, assign it to variable `payoff`, store that in a vector of payoffs `Cs` to then pass at the end of the loop to the aforementioned functions.

```
// Initalize the vector of Calls
vector<double> Cs(NSim);

// Main MC loop
for (long i = 1; i <= NSim; ++i) {
    if (i % 10000 == 0) {
        std::cout << "Simulation number: " << i << std::endl;
    }

    VOld = S_0;
    for (size_t index = 1; index < x.size(); ++index) {
        dW = myNormal->getNormal();

        VNew = VOld + (k * drift(x[index - 1], VOld))
            + (sqrk * diffusion(x[index - 1], VOld) * dW);

        VOld = VNew;
        if (VNew <= 0.0) coun++;
    }

    payoff = myOption.myPayOffFunction(VNew);
    Cs.push_back(payoff);     // Push C_{T,j} to vector
    price += payoff / double(NSim);
}

price *= exp(-myOption.r * myOption.T);
```

Question D.b

> *Run the MC program again with data from Batches 1 and 2. Experiment with different values of `NT` (time steps) and `NSIM` (simulations or draws). How do SD and SE react for these different run parameters, and is there any pattern in regards to the accuracy of the MC (when compared to the exact method)?*

Batch 1: Accuracy of European Call: $T = 0.25$, $K = 65$, $\sigma = 0.30$, $r = 0.08$, $S = 60$

Reference Call Price = 2.13337

Table 1: The **absolute error** of the MC simulation across grid points (`NT`) and number of simulations (`NSIM`) for Batch 1

`NT`, `NSIM`	`NSIM` =100	`NSIM` =1,000	`NSIM` =10,000	`NSIM` =100,000	`NSIM` =1,000,000
`NT` = 2	2.632417e-01	9.203648e-02	1.200790e-01	8.510446e-02	8.340218e-02
`NT` = 3	5.035559e-01	3.235333e-01	1.224027e-01	7.813884e-02	5.147937e-02
`NT` = 4	5.987387e-01	9.473616e-02	1.460946e-02	3.346980e-02	3.233861e-02
`NT` = 5	5.466716e-01	7.456874e-02	9.855942e-03	3.847193e-02	1.890097e-02
`NT` = 6	1.061755e+00	1.753611e-01	5.521700e-02	1.889628e-02	1.844404e-02
`NT` = 7	7.457166e-01	2.418215e-01	1.316571e-01	3.636494e-02	1.242936e-02
`NT` = 8	2.669690e-01	2.127838e-01	1.708850e-02	2.627710e-02	2.159024e-02
`NT` = 9	6.084229e-02	5.314298e-02	2.094094e-02	5.702243e-03	1.804855e-02
`NT` = 10	3.212444e-01	1.475987e-01	7.430848e-04	1.402769e-02	2.317850e-02
`NT` = 100	4.266600e-01	1.503501e-01	5.603340e-03	1.537361e-02	1.758639e-03
`NT` = 1,000	4.071797e-01	2.320302e-02	3.167530e-02	1.191878e-02	4.454782e-03

Table 2: The **standard deviation** of the MC simulation across grid points (NT) and number of simulations (NSIM) for Batch 1

NT, NSIM	NSIM =100	NSIM =1,000	NSIM =10,000	NSIM =100,000	NSIM =1,000,000
NT = 2	3.702759e+00	4.205938e+00	4.134713e+00	4.149745e+00	4.170402e+00
NT = 3	4.891625e+00	3.732387e+00	4.185014e+00	4.262213e+00	4.294542e+00
NT = 4	3.517116e+00	4.434011e+00	4.459829e+00	4.364515e+00	4.353021e+00
NT = 5	5.118550e+00	4.516054e+00	4.395748e+00	4.365230e+00	4.393537e+00
NT = 6	2.414266e+00	4.509912e+00	4.366411e+00	4.410379e+00	4.415072e+00
NT = 7	4.981660e+00	4.830431e+00	4.227897e+00	4.394543e+00	4.430881e+00
NT = 8	5.199024e+00	4.446559e+00	4.464114e+00	4.411672e+00	4.424697e+00
NT = 9	4.213768e+00	4.579448e+00	4.453109e+00	4.476104e+00	4.443539e+00
NT = 10	4.057807e+00	4.294339e+00	4.455329e+00	4.449267e+00	4.435539e+00
NT = 100	3.731941e+00	4.548375e+00	4.430099e+00	4.492470e+00	4.515502e+00
NT = 1,000	4.623455e+00	4.741437e+00	4.470627e+00	4.502637e+00	4.516729e+00

Table 3: The **standard errors** of the MC simulation across grid points (NT) and number of simulations (NSIM) for Batch 1.

NT, NSIM	NSIM =100	NSIM =1,000	NSIM =10,000	NSIM =100,000	NSIM =1,000,000
NT = 2	3.702759e-01	1.330034e-01	4.134713e-02	1.312265e-02	4.170402e-03
NT = 3	4.891625e-01	1.180285e-01	4.185014e-02	1.347830e-02	4.294542e-03
NT = 4	3.517116e-01	1.402157e-01	4.459829e-02	1.380181e-02	4.353021e-03
NT = 5	5.118550e-01	1.428102e-01	4.395748e-02	1.380407e-02	4.393537e-03
NT = 6	2.414266e-01	1.426159e-01	4.366411e-02	1.394684e-02	4.415072e-03
NT = 7	4.981660e-01	1.527516e-01	4.227897e-02	1.389677e-02	4.430881e-03
NT = 8	5.199024e-01	1.406125e-01	4.464114e-02	1.395093e-02	4.424697e-03
NT = 9	4.213768e-01	1.448149e-01	4.453109e-02	1.415468e-02	4.443539e-03
NT = 10	4.057807e-01	1.357989e-01	4.455329e-02	1.406982e-02	4.435539e-03
NT = 100	3.731941e-01	1.438322e-01	4.430099e-02	1.420644e-02	4.515502e-03
NT = 1,000	4.623455e-01	1.499374e-01	4.470627e-02	1.423859e-02	4.516729e-03

Batch 2: Accuracy of European Call: $T = 1.0$, $K = 100$, $\sigma = 0.20$, $r = 0$, $S = 100$

Reference Call Price = 7.96557

Table 3: The **abs error** of the MC simulation across grid points (NT) and number of simulations (NSIM) for Batch 2.

	NSIM =100	NSIM =1,000	NSIM =10,000	NSIM =100,000	NSIM =1,000,000
NT = 2	3.485391e-01	1.202511e-02	6.247254e-02	2.999940e-02	2.521862e-02
NT = 3	1.540124e+00	7.635447e-01	1.544779e-01	4.850612e-02	3.448957e-02
NT = 4	1.722926e+00	4.451495e-01	2.034288e-01	3.689535e-02	4.353653e-02
NT = 5	1.804811e+00	1.569835e-01	8.584167e-02	1.678313e-02	5.679862e-02
NT = 6	3.267576e+00	6.546431e-01	1.049989e-01	3.365332e-02	3.851598e-02
NT = 7	2.364024e+00	6.423566e-01	3.295264e-01	2.332860e-02	4.026738e-02
NT = 8	4.974667e-01	4.276399e-01	1.248284e-01	7.118566e-03	1.174670e-02
NT = 9	5.017132e-01	1.037566e-01	1.213223e-01	6.574340e-02	9.758878e-03
NT = 10	9.372743e-01	2.957527e-01	5.235354e-02	1.579178e-02	1.312259e-02
NT = 100	1.204283e+00	3.692603e-01	1.715665e-02	3.035763e-02	7.707332e-03
NT = 1000	1.055245e+00	9.063880e-02	7.579229e-02	2.864089e-02	1.631538e-02

Table 5: The **standard deviation** of the MC simulation across grid points (NT) and number of simulations (NSIM) for Batch 2.

	NSIM =100	NSIM =1,000	NSIM =10,000	NSIM =100,000	NSIM =1,000,000
NT = 2	1.133303e+01	1.242954e+01	1.226964e+01	1.232857e+01	1.237038e+01
NT = 3	1.416083e+01	1.139988e+01	1.239817e+01	1.257496e+01	1.265163e+01
NT = 4	1.082217e+01	1.300745e+01	1.300898e+01	1.281460e+01	1.278670e+01
NT = 5	1.455594e+01	1.309936e+01	1.288798e+01	1.281749e+01	1.288042e+01
NT = 6	8.381529e+00	1.319100e+01	1.281277e+01	1.291724e+01	1.292694e+01
NT = 7	1.441636e+01	1.394607e+01	1.249895e+01	1.287796e+01	1.296443e+01
NT = 8	1.477896e+01	1.288287e+01	1.303940e+01	1.291955e+01	1.294673e+01
NT = 9	1.248044e+01	1.331490e+01	1.303436e+01	1.307194e+01	1.298788e+01
NT = 10	1.205227e+01	1.261498e+01	1.302500e+01	1.300213e+01	1.297001e+01
NT = 100	1.135694e+01	1.313115e+01	1.295705e+01	1.309141e+01	1.314985e+01
NT = 1000	1.362096e+01	1.362321e+01	1.303816e+01	1.311199e+01	1.315371e+01

Table 6: The **standard errors** of the MC simulation across grid points (NT) and number of simulations (NSIM) for Batch 2.

NT , NSIM	NSIM =100	NSIM =1,000	NSIM =10,000	NSIM =100,000	NSIM =1,000,000
NT = 2	1.133303e+00	3.930564e-01	1.226964e-01	3.898636e-02	1.237038e-02
NT = 3	1.416083e+00	3.604959e-01	1.239817e-01	3.976550e-02	1.265163e-02
NT = 4	1.082217e+00	4.113318e-01	1.300898e-01	4.052333e-02	1.278670e-02
NT = 5	1.455594e+00	4.142380e-01	1.288798e-01	4.053246e-02	1.288042e-02
NT = 6	8.381529e-01	4.171362e-01	1.281277e-01	4.084791e-02	1.292694e-02

NT = 7	1.441636e+00	4.410136e-01	1.249895e-01	4.072369e-02	1.296443e-02
NT = 8	1.477896e+00	4.073921e-01	1.303940e-01	4.085522e-02	1.294673e-02
NT = 9	1.248044e+00	4.210541e-01	1.303436e-01	4.133711e-02	1.298788e-02
NT = 10	1.205227e+00	3.989208e-01	1.302500e-01	4.111634e-02	1.297001e-02
NT = 100	1.135694e+00	4.152434e-01	1.295705e-01	4.139866e-02	1.314985e-02
NT = 1000	1.362096e+00	4.308036e-01	1.303816e-01	4.146374e-02	1.315371e-02

Conclusion

Looking at Table 1 and Table 4, which displays the **pricing error** in absolute terms we see both increases in grid size and number of simulations tends to decrease error. While increasing the number of simulations does not bias the estimate, we are much more likely to get a smaller draw from the distribution of abs. err. when the number of simulations is smaller. Increasing the number of grid points does tend to result in lower bias (smaller error), but the results are minimal. Even with `NT=2` we get a good approximation to prices with very large number of simulations.

Looking at Table 2 and Table 5 which shows the **standard deviation** of MC prices we no discernable pattern with the number of grid points or simulations. This is to be expected. The distribution of prices itself should converge to a stable distribution. But as we see in Table 3 and Table 6 which shows the **standard error** (volatility around the mean estimate) we see that the standard error decreases by a factor of roughly $\sqrt{NSIM}$ as we increase the order of magnitude of $NSIM$.