Advanced Monte Carlo

This part is focused on the advanced monte carlo where Standard Deviation and Standard Error are calculated as measures to better infer the accuracy of a Monte Carlo run.
Since the NSim and NT values greatly affect the accuracy of a Monte Carlo run, these measures help understand how the error will be impacted by change in these values.

$$\mathrm{SD} = \sqrt{\frac{\sum C_{T,j}^2 - \frac{1}{M}\left(\sum C_{T,j}\right)^2}{M-1}} \times \exp\left(-rT\right)$$

$$SE = \frac{SD}{\sqrt{M}}$$

where

$C_{T,j} = \text{call output price at } t = T \text{ for the } j\text{th simulation}, 1 \leq j \leq M,$

$M = \text{number of simulations}.$

Code Changes

A generic function has been added to the code to calculate SD and SE. Same function is being used to optimize recomputation of SD whenever SE is required. The function therefore returns a tuple of <SD, SE> values per a monte carlo run.

```
// Calculate SD and SE
// Returns a tuple of <SD, SE>
template <typename Type>
boost::tuple<Type, Type> SDSE(const std::vector<Type>& prices, Type r, Type T)
```

Main function is modified to run both the batches in a loop and for every batch, predefined NSim and NT values are passed to the simulation from the experience of Group C.
Put and call values are smartly calculated simultaneously as the stock price is the one being simulated and this saves recomputation for the two types. SDSE function is then called on the prices vector for t=T and output is saved to a .csv file

Since this was computationally very intensive, I had started the program and left it to run. Results were neatly documented when I came back after a couple of hours. Sweet!
Now, let us see the results from the Monte Carlo runs on batch 1 and batch 2.

Batch 1

T = 0.25, K = 65, sig = 0.30, r = 0.08, S = 60

Call Option

NSim	NT	Call Price (MC)	Call Price (Exact)	Error (Exact)	Call SD	Call SE
10000	100	2.1378	2.13337	-0.00443	0.625836	0.00625836
10000	1000	2.13162	2.13337	0.00175	0.623019	0.00623019
100000	100	2.13043	2.13337	0.00294	0.616161	0.00194847
100000	1000	2.14465	2.13337	-0.01128	0.619007	0.00195747
1000000	100	2.13271	2.13337	0.00066	0.615599	0.000615599
1000000	1000	2.13456	2.13337	-0.00119	0.616498	0.000616498

Put Option

NSim	NT	Put Price (MC)	Put Price (Exact)	Error (Exact)	Put SD	Put SE
10000	100	5.90807	5.84628	-0.06179	0.625836	0.00625836
10000	1000	5.93793	5.84628	-0.09165	0.623019	0.00623019
100000	100	5.87321	5.84628	-0.02693	0.616161	0.00194847
100000	1000	5.87066	5.84628	-0.02438	0.619007	0.00195747
1000000	100	5.85125	5.84628	-0.00497	0.615599	0.000615599
1000000	1000	5.83585	5.84628	0.01043	0.616498	0.000616498

Batch 2

T = 1.0, K = 100, sig = 0.20, r = 0.0, S = 100

Call Option

NSim	NT	Call Price (MC)	Call Price Exact)	Error (Exact)	Call SD	Call SE
10000	100	7.94097	7.96557	0.0246	2.31603	0.0231603
10000	1000	7.92226	7.96557	0.04331	2.31407	0.0231407
100000	100	7.94362	7.96557	0.02195	2.29788	0.00726652
100000	1000	7.98184	7.96557	-0.01627	2.30462	0.00728786
1000000	100	7.9625	7.96557	0.00307	2.29908	0.00229908
1000000	1000	7.97529	7.96557	-0.00972	2.30419	0.00230419

Put Option

NSim	NT	Put Price (MC)	Put Price (Exact)	Error (Exact)	Put SD	Put SE
10000	100	8.06336	7.96557	-0.09779	2.31603	0.0231603
10000	1000	8.12311	7.96557	-0.15754	2.31407	0.0231407
100000	100	8.0079	7.96557	-0.04233	2.29788	0.00726652
100000	1000	8.00674	7.96557	-0.04117	2.30462	0.00728786
1000000	100	7.97439	7.96557	-0.00882	2.29908	0.00229908
1000000	1000	7.94982	7.96557	0.01575	2.30419	0.00230419

Observations

We see a general rule that large NSim and large NT lead to better results with decreasing SD and SE.
While increasing NSim, SD first falls and then might rise a little. SE however seems to be falling with increasing NSim.
SE decreases with increasing NSim, but the rate of decrease substantially slows down with growing NSim. To further decrease SE, NSim might have to be increased with even higher orders.
With larger NSim (1,000,000), increasing NT (100 => 1000) increases SE slightly.