Black-Scholes option pricing

Victor Podlozhnyuk
vpodlozhnyuk@nvidia.com

July 2012

Document Change History

Version	Date	Responsible	Reason for Change
0.9	2007/03/19	vpodlozhnyuk	Initial release
1.0	2007/04/06	mharris	Minor clarity / grammar edits for initial release

Abstract

The pricing of options is a very important problem encountered in financial engineering since the creation of organized option trading in 1973. This sample shows an implementation of the Black-Scholes model in CUDA for European options.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com

Introduction

The most common definition of an *option* is an agreement between two parties, the *option seller* and the *option buyer*, whereby the option buyer is granted a right (but not an obligation), secured by the option seller, to carry out some operation (or *exercise* the option) at some moment in the future. The predetermined price is referred to as the *strike price*, and the future date is called the *expiration date*. (See Kolb & Pharr. [1])

Options come in several varieties:

A *call option* grants its holder the right to *buy* the *underlying asset* at a *strike price* at some moment in the future.

A *put option* gives its holder the right to *sell* the *underlying asset* at a *strike price* at some moment in the future.

There are several types of options, mostly depending on when the option can be exercised. European options can be exercised only on the expiration date. American-style options are more flexible as they may be exercised at any time up to and including expiration date and as such, they are generally priced at least as high as corresponding European options. Other types of options are path-dependent or have multiple exercise dates (Asian, Bermudian).

For a call option, the profit made at the exercise date is the difference between the price of the asset on that date and the strike price, minus the option price paid. For a put option, the profit made at the exercise date is the difference between the strike price and the price of the asset on that date, minus the option price paid.

The price of the asset at expiration date and the strike price therefore strongly influence how much one would be willing to pay for an option.

Other important factors in the price of an option are:

- **The time to the expiration date, *T***: Longer periods imply a wider range of possible values for the underlying asset on the expiration date, and thus more uncertainty about the value of the option.
- **The riskless rate of return, *r*,** which is the annual interest rate of bonds or other "risk-free" investments: Any amount P of dollars is guaranteed to be worth $P \cdot e^{rT}$ dollars T years from now if placed today in one of theses investments or in other words, if an asset is worth P dollars T years from now, it is worth $P \cdot e^{-rT}$ today.

This example demonstrates a CUDA implementation of the Black-Scholes model for European options.

Black-Scholes model.

The Black-Scholes model provides a partial differential equation (PDE) for the evolution of an option price under certain assumptions. For European options, a closed-form solution exists for this PDE. (See Black & Scholes, [2])

$$V_{call} = S \cdot CND(d_1) - X \cdot e^{-rT} \cdot CND(d_2)$$

$$V_{put} = X \cdot e^{-rT} \cdot CND(-d_2) - S \cdot CND(-d_1)$$

$$d_1 = \frac{\log(\frac{S}{X}) + (r + \frac{v^2}{2})T}{v\sqrt{T}}$$

$$d_2 = \frac{\log(\frac{S}{X}) + (r - \frac{v^2}{2})T}{v\sqrt{T}}$$

$$CND(-d) = 1 - CND(d)$$

where

V_{call} is the price for an option call,

V_{put} is the price for an option put,

$CND(d)$ is the Cumulative Normal Distribution function,

S is the current option price,

X is the strike price,

T is the time to expiration.

r is the continuously compounded risk free interest rate,

v is the implied volatility for the underlying stock,

The cumulative normal distribution function is computed with a polynomial approximation that provides six-decimal-place accuracy. The expansion uses a fifth-order polynomial. (See Hull, [3])

Implementation details

Choosing a data storage layout

The existence of a closed-form expression makes calculating option prices an easy task. The main problem is choosing the best data storage layout.

Here are some characteristics of the G80-class GPU:

- Many execution threads simultaneously perform the same instruction at hardware level.
- No memory caching for "raw" device memory operations. Data caching is handled explicitly by programmer in shared memory if required (i.e. if the same data element is accessed several times within the same thread and/or shared among several threads)
- Memory coalescing: reads and writes within each warp should be arranged sequentially, so that all memory requests can be coalesced into a single continuous block with base address aligned to 16 * <element size> byte boundaries for best performance. At this point it is important to understand that memory coalescing occurs between *threads of the warp* simultaneously executing *the same instruction*, not between successive (in time) memory operations.

A G80 device is capable of loading words of only 4, 8 or 16 bytes, so in the general case arrays of structures aligned on these boundaries can both increas memory waste and result in multiple non-coalesced memory accesses per structure read per memory. This can be very detrimental to performance on the GPU, so it is strongly preferred to arrange data into arrays of elementary types (also known as the "structure of arrays" strategy), since it enables memory coalescing for any number of elementary-type input data properties with no wasted memory.

Thread count and data size

A simple implementation of the Black-Scholes algorithm would assign each thread to a specific index of input data. But there are some hardware constraints to be taken into account:

- Block grid dimensions on G80 are only 16-bit (i.e. the maximum size of each dimension is 65536).
- The maximum number of threads per block is 512. Depending on the use of shared memory and registers, the optimal number of threads for maximum performance is in the 192-256 range.

Therefore, a one-to-one correspondence between thread count and input data size restricts the maximum input data size (around 33 millions options, if we stick to convenient 1D grid configuration) and also requires (re)configuring the execution grid in accordance to input data size.

To allow for arbitrary numbers of options, each thread should process more than one index if required; this is implemented with the code in Listing 1.

```
const int      tid = blockDim.x * blockIdx.x + threadIdx.x;
const int THREAD_N = blockDim.x * gridDim.x;

for(int opt = tid; opt < OptN; opt += THREAD_N)
    BlackScholesBody(
        d_CallResult[opt],
        d_PutResult[opt],
        d_StockPrice[opt],
        d_OptionStrike[opt],
        d_OptionYears[opt],
        Riskfree,
        Volatility
    );
```

Listing 1. This code shows how we break the dependency between the compute grid configuration and the input data size.

In Listing 1, no matter how small THREAD_N is or how large OptN is, exactly OptN indices will be processed with perfect memory coalescing. The only recommendation here is for OptN to be a multiple of the warp size (32 threads on G80 GPUs) to avoid hardware idling, which anyway is negligible in case of large OptN.

Bibliography

1. Craig Kolb and Matt Pharr (2005). "Option pricing on the GPU". *GPU Gems 2*. Chapter 45.
2. Fischer Black and Myron Scholes (1973). "The Pricing of Options and Corporate Liabilities". *Journal of Political Economy* **81** (3): 637-654.
3. John C. Hull (1997) "Options, Futures, and Other Derivatives"