Antoine Savine, Rolf Poulsen; October 17, 2018

Computational Finance: Hand-In #2

Deadline CET 23:59 on Sunday November 11, 2018. Hand-in via Absalon.

Each named exercise counts for 30 points, leading to a maximal cumulative point score of 120. The total point score for the course is calculated as

x_total = min(100*(0.5*x_HI1 /100 + 0.5*x_HI2/100),100),

where the x_'s are the respective scores on the two hand-ins. The total point score is then mapped to the final grade via following this semi-official table (where teacher discretion is used in borderline cases; "helhedsvurdering i grænsetilfælde"):

Grade	Intl' letter (US A-F-scale)	Distribution (given Pass)	Points
12	A	10	91-100
10	B	25	80-90
7	C	30	66-79
4	D	25	55-65
02	E	10	50-54
00	Fx	–	30-49
-3	F	–	0-29

The goal of this hand-in is to implement the valuation of European options in the so-called CEV (Constant Elasticity Volatility) model, a simple extension of the Black-Scholes model specified by the risk-free dynamics

$$dS = \lambda S^{\beta} dW \ ,$$

where λ is the CEV volatility, the elasticity $\beta$is a parameter between 0 and 1, and $W$is a standard Brownian Motion under the risk-neutral measure. For simplicity, we ignore interest rate and dividends so S is a martingale under the pricing measure.

Exercise I: Analytic solution

Code in C++ and export to Excel the value of a European call under the CEV dynamics as a function of the spot S_0, λ, β, expiry T, and strike K of the call. Choose one or more of the following strategies.

Strategy 1: Exact formula

Volatility Skews and Extensions of the Libor Market Model, Andersen and Andreasen, 1998 derives a formula a la Black-Scholes but with a displaced chi-square distribution in place of the classic normal distribution. An implementation of a displaced chi-square cumulative distribution is therefore necessary, which is nontrivial and requires personal research. The Andersen and Andreasen reference may be overkill since much of their material is not relevant to us here; you may find this note on CEV equivalence more helpful.

Strategy 2: Harmonic approximation

ZABR – *Expansions for the Masses, Andreasen and Huge, 2012* establishes a framework for the production of expansions (approximate closed-form solutions) in a wide variety of local and stochastic volatility models, including CEV as a simple case. The basic result for non-stochastic, time homogeneous local volatility models is that in a model of the form:

$$\frac{dS}{S} = g(S)\,dW$$

The equivalent Black-Scholes implied volatility for strike K (independently of expiry T) is approximately

$$\sigma(K) \approx \frac{\ln S - \ln K}{\int_K^S \frac{du}{u\,g(u)}}.$$

see also page 196-197 in the volatility-slides from FinKont2.

Strategy 3: Shifted-log approximation

On pages 80-86 in the above volatility-slides, it is suggested to approximate the CEV model by a model of the form

$$dS = (a + bS)\,dW\,,$$

with well-chosen a and b parameters mapped from the CEV λ and β. This particular model (called shifted-log or displaced Black-Scholes) has a simple closed-form solution for call-option prices. Note that in this solution, we do not approximate the solution in the exact model, we approximate the model by one that admits a simple exact solution.

Exercise II: FDM (Finite Difference Method) solution

Short version: Use the Crank-Nicolson method to calculate prices European call-options in the CEV model. Code in C++ and export to Excel. Verify and discuss the matching of analytical and FDM solutions.

Hint/instructions: This is a “throw everything and the kitchen sink at it" question. Things you need to think – more or less deeply – about are:

- Tridag
- Treatment of non-constant coefficients is spelled out here
- Variable transformation
- Boundary conditions
- Rannacher stepping
- Is the global convergence order $O(dt^2 + dx^2)$? Østerby, chapter 10

Exercise III: MC (Monte-Carlo) solution

1. Implement a solution with Monte-Carlo simulations in C++, exported to Excel. A generic solution (that easily extends to other models or options) is not necessary, although a modular, well-decoupled program are considered a plus. Target maximum algorithmic and computational efficiency as discussed in class and profile and optimize your code as explained in the curriculum. Implement your Monte-Carlo solution, along the analytic and FDM solution, in your own, minimal, library. Do not reuse the framework of the companion code of the curriculum. For random number generation, use L'Ecuyer's mrg32k3a RNG (random number generator) implemented in mrg32k3a.h in the gutHub repo www.github.com/asavine/CompFinance/wiki To turn uniform random numbers into standard Gaussians, use Zelen and Severo's approximation of the inverse cumulative normal distribution implemented in gaussians.h. (Copy and adapt the code in gaussians.h and mrg32k3a.h to your own library). Similarly to the Heston-model from Hand-In #1, the CEV-model cannot (easily) be simulated exactly, rather you will have to discretize using, say, an Euler-scheme. This means that thought needs to be given to the choice of discretization step-size and to retaining positivity of S, for instance by log-transforming.

2. Parallelize your solution with the techniques and constructs seen in class. Turn to the curriculum for help (it is acceptable to reuse the code of the thread pool and concurrent queue), but implement your own solution, do not reuse the framework in the companion code. Use mrg32k3a's skip ahead algorithm to guarantee that the parallel solution matches the serial solution in all cases. The parallel efficiency (speed-up over the number of physical cores) must be 100% or very close.

The serial and parallel solutions must exactly match in all cases, and the MC solution must be checked against FDM.

Exercise IV: Greeks and AAD

First: Extend the FDM and MC (or just one of them) to a time-dependent volatility model of the type

$$dS = \lambda(t) S^{\beta} dW ,$$

where, in practice, $\lambda(t)$ is given by interpolation from a discrete set (t_i, λ_i) where it is considered that the λ_i's are the parameters of the model. Implement the interpolation efficiently, making sure to use STL algorithms (correctly) instead of handcrafted loops and implement a logarithmic search for bracketing. Optional: Research the notion of "parameter averaging" to also extend (approximately) the analytic solution.

Second: Estimate the so-called model risk Greeks $\frac{\partial V}{\partial S_0}$, $\frac{\partial V}{\partial \lambda_i}$ and $\frac{\partial V}{\partial \beta}$ of the FDM and MC valuation functions previously implemented.

1. As a baseline, implement a finite difference estimation: Bump S_0, the λ_i's and β one by one and repeat valuation. This may be done in C++ or directly in an Excel spreadsheet.

2. Instrument your FDM and/or MC (both serial and parallel) implementation(s) with AAD. The AAD library, consisting of the files AAD*.*, gaussians.h and blocklist.h may be copied from the GitHub

repository directly to your solution. Your challenge is to instrument your own FDM and MC code with the AAD library.

3. Compare FD and AAD results and calculation times with varying numbers of λ_i's (i.e. various lengths of the λ-vector.) You should see very similar, nearly identical results for the Greeks in all cases. AAD speed should be similar or inferior to FD with constant lambda. With an increasing number of λ_i's, a correct implementation should see a (linear + logarithmic) growth in calculation time for FD and logarithmic growth for AAD, resulting in a massive improvement in speed with a large number of λ_i's (say a few hundred).