Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 179 (2006) 535–544
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Pricing multi-asset American-style options by memory reduction Monte Carlo methods

Raymond H. Chan, Chi-Yan Wong *, Kit-Ming Yeung

Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong

Abstract

When pricing American-style options on d assets by Monte Carlo methods, one usually stores the simulated asset prices at all time steps on all paths in order to determine when to exercise the options. If N time steps and M paths are used, then the storage requirement is $d \cdot M \cdot N$. In this paper, we give a simulation method to price multi-asset American-style options, where the storage requirement only grows like $(d+1)M+N$. The only additional computational cost is that we have to generate each random number twice instead of once. For machines with limited memory, we can now use larger values of M and N to improve the accuracy in pricing the options.

Keywords: Memory reduction method; Monte Carlo method; Multi-asset; American-style options; Random number

1. Introduction

Monte Carlo method is one of the main methods for computing American-style options, see for instance [12,2,3,9]. These algorithms are computationally inefficient because they require the storage of all asset prices at all simulation times for all simulated paths. Thus the total storage requirement grows like $O(dMN)$ where d is the number of underlying assets, M is the number of simulated paths and N is the number of time steps. The accuracy of the simulation is hence severely limited by the storage requirement.

The apparent difficulties in using Monte Carlo methods to price American-style options come from the backward nature of the early exercise feature. There is no way of knowing whether early exercise is optimal when a particular asset price is reached at a given time. One can look at this problem from another point of view. In Monte Carlo method, the simulated paths are all generated in the time-increasing direction, i.e. they start from the initial asset prices $\mathbf{x}_0$ and march to the expiry date according to a given geometric Brownian

* Corresponding author.
E-mail addresses: rchan@math.cuhk.edu.hk (R.H. Chan), cywong@math.cuhk.edu.hk (C.-Y. Wong), kmyeung@math.cuhk.edu.hk (K.-M. Yeung).

0096-3003/$ - see front matter
doi:10.1016/j.amc.2005.11.108

motion. But since the pricing of American options is a backward process starting from the expiry date back to $\mathbf{x}_0$, the usual approach is to save all the intermediate asset prices along all the paths.

In this paper, we use our simulation method in [4] for computing multi-asset American-style options that does not require storing of all the intermediate asset prices. The storage is reduced from $O(dMN)$ to $(d+1)M+N$ only. Our main idea is to generate the paths twice: one in a forward sweep to establish the asset prices at the expiry date, and one in a backward sweep that computes the intermediate asset prices only when they are needed. The only additional cost in our method is that we have to generate each random number twice instead of once. The resulting computational cost is less than twice of that of the methods where all the intermediate asset prices are stored.

The remainder of this paper is organized as follows. Section 2 recalls the usual full-storage approach for computing multi-asset American-style options. Section 3 gives some background about random number generators in computers. In Section 4, we introduce our memory reduction method. In Section 5, we show how to use our method to compute multi-asset American options by adopting it to the least-squares method proposed by Longstaff and Schwartz [9]. Section 6 gives some numerical results to illustrate the effectiveness of our method.

We will use the MATLAB language [11] to explain how the codes are to be written as the language is easier to comprehend. The corresponding commands in FORTRAN 90 [5] and MATHEMATICA [13] are given in Appendix A.

2. The full-storage method

As usual, we let the prices of d non-dividend paying assets $\mathbf{x}=(x^1,x^2,\ldots,x^d)^{\mathrm{T}}$ follow the geometric Brownian motion

$$\frac{dx^k}{x^k}=r\,\mathrm{d}t+\sigma_k\,\mathrm{d}W_k,\quad 1\leqslant k\leqslant d,$$

where r is the risk-free interest rate, σ_k is the volatility of asset k, and $\mathrm{d}W_k$ is the Wiener process for asset k. By Ito's Lemma, we have

$$\begin{aligned}x^k(t+\mathrm{d}t)&=x^k(t)\exp\left(\left(r-\frac{1}{2}\sigma_k^2\right)\mathrm{d}t+\sum_{j=1}^{d}v_{kj}\,\mathrm{d}W_k\right)\\&=x^k(t)\exp\left(\left(r-\frac{1}{2}\sigma_k^2\right)\mathrm{d}t+\sqrt{\mathrm{d}t}\sum_{l=1}^{d}\mathscr{V}(k,l)\mathbf{z}(l)\right),\end{aligned}\tag{1}$$

where $x^k(t)$ is the price of asset k at time t, $\mathbf{z}$ is a d-vector of standard normal random variables, and $\mathscr{V}=[v_{ij}]$ is the volatility matrix. The volatility matrix $\mathscr{V}$ is given by

$$\mathscr{C}=\mathscr{V}\mathscr{V}^{\mathrm{T}},$$

and $\mathscr{C}=[c_{ij}]$ is the d-by-d covariance matrix with $c_{ij}=\rho_{ij}\sigma_i\sigma_j$, where ρ_{ij} is the correlation coefficient between $\mathrm{d}W_i$ and $\mathrm{d}W_j$, see for instance [8].

In the Monte Carlo simulation, we divide the time horizon into N time steps with each having the length

$$\Delta t=\frac{T-t_0}{N},$$

where t_0 is the current time and T is the expiry date of the option. Thus the time horizon is discretized as $t_0<t_1<\cdots<t_N=T$ where $t_j=t_0+\mathrm{j}\,\Delta t$.

Let the asset prices at time t_0 be $\mathbf{x}_0=(x_0^1,x_0^2,\ldots,x_0^d)^{\mathrm{T}}$. Given $\mathbf{x}_0$, we can simulate the price paths using (1). More precisely, for asset k, $1\leqslant k\leqslant d$, if we are to simulate M paths, then the ith path can be defined by the recurrence:

$$\mathbf{S}_j^i(k)=\mathbf{S}_{j-1}^i(k)\exp\left(\left(r-\frac{\sigma_k^2}{2}\right)\Delta t+\sqrt{\Delta t}\sum_{l=1}^{d}\mathscr{V}(k,l)\mathbf{z}_j^i(l)\right),\quad 1\leqslant i\leqslant M,\ \ 1\leqslant j\leqslant N,\tag{2}$$