Journal of Computational and Applied Mathematics 274 (2015) 70–78

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Fast Greeks by simulation: The block adjoint method with memory reduction[☆]

Wenbin Hu, Shenghong Li [*]

Department of Mathematics, Zhejiang University, Hangzhou 310027, China

HIGHLIGHTS

- Propose the block simulation method which is an extension of the common Monte Carlo simulation.
- Can accelerate the adjoint method by several times.
- Simple to implement, without extra work and loss in accuracy.
- Memory reduction is conducted so that the block simulation can run with the adjoint method better.

ARTICLE INFO

Article history:
Received 23 December 2013
Received in revised form 15 May 2014

Keywords:
Monte Carlo
Block simulation
Sensitivity
Adjoint
Memory reduction

ABSTRACT

The pathwise method is one of the approaches to calculate the option price sensitivities by Monte Carlo simulation. Under the general diffusion process, we usually use the Euler scheme to simulate the path of the underlying asset, which requires small time spaces to assure the convergence. The adjoint method can be used to accelerate the calculation of the Greeks in such case. However, it needs to store the intermediate information along the path. In this paper, we propose to use the block simulation to further accelerate the calculation. Block simulation can be seen as an extension of the common Monte Carlo simulation. It is simple to implement, without any extra work and loss in accuracy. It also has the flexibility on the block division, fitting to the computation environment. Moreover, we use the extended forward-path method along with the real time calculation strategy to do the memory reduction so that it can be combined with the adjoint method better. The numerical tests show our method can accelerate the adjoint method by several times. Our method is relatively even more efficient in the high-dimensional case.

1. Introduction

Price sensitivities (or Greeks) of financial derivatives are of vital importance, more than the price itself. One reason is that the price can be observed in the market while the sensitivities cannot. More importantly, the sensitivities determine the hedging strategy of the derivatives. There are mainly three types of Monte Carlo methods for calculating sensitivities, i.e., the finite difference method, the pathwise method and the likelihood ratio method. Another method is Malliavin calculus, which can be seen as the extension of the likelihood ratio method. Extensive numerical evidence accumulated across many models and applications indicates that the pathwise method, when applicable, provides the best estimates of sensitivities (refer to [1] for details).

☆ This work is supported by National Natural Science Foundation of China (No. 11171304 and No. 71371168) and Zhejiang Provincial Natural Science Foundation of China (No. Y6110023).

* Corresponding author. Tel.: +86 130 1897 3561.
E-mail addresses: lshzju@126.com, shli@zju.edu.cn (S. Li).

http://dx.doi.org/10.1016/j.cam.2014.07.005
0377-0427/© 2014 Elsevier B.V. All rights reserved.

The adjoint method proposed in [2] and [3] is an approach to accelerate the calculation of sensitivities by the pathwise method. It applies when the underlying assets follow a general diffusion process. The key idea of this method comes from Algorithmic Differentiation [4]. In contrast to a forward pathwise calculation, it works backward recursively using adjoint variables. The adjoint method is advantageous for calculating the sensitivities of a small number of securities with respect to a large number of parameters. One of the major disadvantages of the adjoint method is that we need to store the necessary information along the path in the memory for the backward sweep. When the method is used path by path, the memory requirement is relatively minor. However, when multi-path simulation is required, e.g. in the block simulation which will be proposed in Section 3, it may go out of memory.

The forward-path method was first proposed in [5] to do the memory reduction when pricing multi-dimensional American options by Least Square Monte Carlo [6]. This method is also applied to exponential Levy process [7]. Hu and Li [8] extend the forward-path method and make it applicable to general diffusion processes. The extended forward-path method can be used conveniently to re-calculate the information needed in the backward sweep of the adjoint method. As a result, the whole path storage is not required, which significantly alleviates the memory issue.

In this paper, we propose to use the block simulation, combined with the adjoint method, to accelerate the pathwise sensitivity calculation. The block simulation can be seen as an extension of the common Monte Carlo simulation. It is simple to implement, without any extra work and loss in accuracy. It also has the flexibility on the block division fitting to the computation environment. We use the extended forward-path method as well as the real time calculation to do the memory reduction. Numerical tests show that our method significantly accelerates the calculation. By applying the extended forward-path method, the memory requirement can be reduced from $O(dMN)$ to $O(dM)$, where d is the number of underlying assets, M is the number of simulated paths and N is the number of timesteps.

The rest of this paper is organized as follows. Section 2 briefly reviews the adjoint method proposed in [2]. In Section 3, the block simulation is proposed and we explain in detail on how to combine the block simulation with the adjoint method. Section 4 shows how to do the memory reduction by the extended forward-path method as well as the real time calculation strategy. Section 5 summaries our method and gives the algorithm. Numerical tests are implemented in Section 6 on two specific examples. We conclude in Section 7.

2. Pathwise Greeks: adjoint method

We begin with a brief review of the adjoint method for computing the price delta in a general multi-dimensional process which satisfies the following stochastic differential equation (SDE)

$$d\mathbf{X}(t) = \mathbf{a}(\mathbf{X}(t))dt + \mathbf{b}(\mathbf{X}(t))d\mathbf{W}(t) \tag{1}$$

The process $\mathbf{X}$ takes values in R^d, $\mathbf{a}$ is a d-dimensional vector and $\mathbf{b}$ takes values in $R^{d\times m}$. $\mathbf{W}$ is an m-dimensional Brownian motion, whose components are independent from each other. In Monte Carlo simulation, we usually use the Euler approximation to make a time-discretized evolution for the general process $\mathbf{X}$. By denoting $\widehat{\mathbf{X}}$ as the approximation to $\mathbf{X}$, the Euler method with fixed spacing h can be expressed as

$$\widehat{\mathbf{X}}(n+1) = \widehat{\mathbf{X}}(n) + \mathbf{a}(\widehat{\mathbf{X}}(n))h + \mathbf{b}(\widehat{\mathbf{X}}(n))\mathbf{Z}(n)\sqrt{h} \tag{2}$$

where $\mathbf{Z}(1), \mathbf{Z}(2), \ldots$ are independent m-dimensional standard normal random vectors. Now we consider the problem of estimating the delta of a derivative with discounted payoff function g and the expiration T, i.e.

$$\frac{\partial}{\partial \mathbf{X}_j(0)} E[g(\mathbf{X}(T))]$$

which is the delta with respect to the jth underlying asset. Using the Euler scheme (2), we approximate the pathwise derivative estimate using

$$\sum_{i=1}^{d} \frac{\partial g(\mathbf{X}(N))}{\partial \mathbf{X}_i(N)} \boldsymbol{\Delta}_{ij}(N)$$

with $N = \lfloor T/h \rfloor$ an integer and

$$\boldsymbol{\Delta}_{ij}(n) = \frac{\partial \mathbf{X}_i(n)}{\partial \mathbf{X}_j(0)},\ i, j = 1, \ldots, d.$$

So besides the simulation of the asset prices at T, we need to simulate the $\boldsymbol{\Delta}_{ij}(N)$. By differentiating Eq. (2) on both sides we get the evolution approximation of $\boldsymbol{\Delta}_{ij}(n)$:

$$\boldsymbol{\Delta}_{ij}(n+1) = \boldsymbol{\Delta}_{ij}(n) + \sum_{k=1}^{d} \frac{\partial \mathbf{a}_i(\widehat{\mathbf{X}}(n))}{\partial \mathbf{X}_k} \boldsymbol{\Delta}_{kj}(n)h + \sum_{l=1}^{m}\sum_{k=1}^{d} \frac{\partial \mathbf{b}_{il}(\widehat{\mathbf{X}}(n))}{\partial \mathbf{X}_k} \boldsymbol{\Delta}_{kj}(n)\mathbf{Z}_l(n)\sqrt{h}.$$

The matrix recursion is

$$\boldsymbol{\Delta}(n+1) = \mathbf{D}(n)\boldsymbol{\Delta}(n) \tag{3}$$