Monte Carlo Pricing using Operator Methods and Measure Changes

Claudio Albanese [*], Hongyun Li [†]

August 25, 2009, last revised September 7, 2009

Abstract

A large class of generic stochastic processes which are not necessarily analytically solvable but are still numerically tractable can be described by giving transition probability kernels over a contiguous set of time intervals. From the numerical viewpoint, this procedure is highly effective on current microchip architectures as kernels can be conveniently evaluated using GPU co-processors and then used for scenario generation while storing them in CPU caches. This paper describes the pricing methodology and a mathematical framework for Finance based on direct kernel manipulations, i.e. operator methods. We also discuss a number of techniques based on measure changes to accomplish tasks such as variance reduction and sensitivity calculations. Numerical experiments are included along with performance benchmarks. Source code is distributed separately online under GPL license in a library named OPLib.

Contents

[*]Department of Mathematics, King's College London, claudio@albanese.co.uk
[†]Department of Mathematics, Imperial College London, hongyun.li06@ic.ac.uk

1 Introduction

This article discusses the optimal design of Monte Carlo pricing algorithms on current and emerging microchip architectures, in particular multi-core CPUs and GPUs and outlines a natural mathematical framework within which to approach the technical issues in the most direct and efficient way.

Recent advances in computer technology motivate us to be ambitious and aim at achieving not only greater performance but also greater model flexibility. We thus focus on models freely specified without the constraint of analytic solvability in the form of Markov chains with a number of state variables in the range 128-1024 for each risk factor. The task of carrying out a Monte Carlo simulation is thus split into a first stage for obtaining transition probability kernels and a second stage for actually generating scenarios. Although we focus entirely on single factor dynamics in this paper, the methods do extend to multi-factor processes by correlating marginal return distributions through dynamic copulas.

Our research led us to the conclusion that the optimal strategy for orchestrating such an application involves using GPUs for the calculation of kernels and CPUs for the scenario generation. The reasons are fairly technical and are related to the very different characteristics of these microchips in terms of

- memory architecture: physical memory plus three levels of caches for CPUs versus global memory plus a combination of shared and constant memory along with a large number of registers for GPUs,
- clock frequency: 2-3 GHz for CPUs versus about 1GHz for GPUs
- number of cores: 4-16 for CPUs versus 120-240 for GPUs,
- threading model: native POSIX or managed .NET threads with locks, mutex and semaphores for CPUs versus fast light-weight threads without locking mechanisms for GPUs)
- instruction flow management: MIMD multiple-instruction-multiple-data paradigm for CPUs supplemented by SSE2 (streaming SIMD extensions) instructions versus SIMT single-instruction-multiple-threads paradigm for GPUs.

Because of the vast architectural differences between CPUs and GPUs, it is no surprise that relative floating point performance depends very much on the task at hand. GPUs are overwhelmingly more efficient at performing full-matrix linear algebra tasks such as matrix-matrix multiplication and tensorial extensions such as concurrent matrix or matrix-vector multiplication, while CPUs shine at tasks such as scenario generation which greatly benefit of the rich cache hierarchy to speed up random memory access.

In this paper, we describe in detail the algorithms we have built and which we consider optimal. Realizing that nothing can be more informative than actually providing source code, we accompany the release of this article with that of an open source library named OPLib (Albanese 2009) containing the key building blocks of the Monte Carlo algorithms discussed in this paper, implemented and optimized separately on both CPU and GPU architectures.

The Mathematical framework to best understand these methods is described in this paper in its most direct form. As with all applied mathematics, abstract constructs are motivated and draw legitimacy from the underlying computing technology. In the case of Mathematical Finance and pricing theory we are faced with several distinct traditions rooted in the history of probability theory. The tradition of stochastic analysis is well suited to handle situations where transition probability kernels are out of the reach of direct numerical methods and need to be either evaluated in closed form or expressed through asymptotic expansions. Many of these calculations can be carried out using either stochastic calculus or PDE methods, the former being arguably more elegant because of the path-wise representations, the latter more powerful algorithmically. Both methodologies are based on continuum mathematics and infinitesimal calculus. To frame probability in such a way that continuous probability distributions can be treated directly as opposed to be represented as finitary limits, one uses the classical Kolmogorov axioms for probability and in particular the axiom of countable additivity (Kolmogorov 1933). These cannot be understood constructively in full generality but that is not necessary if the aim is to obtain analytical formulas for transition probability kernels or functions thereof.

Recent innovations in computing technology give one the ability of directly manipulating transition probability kernels, i.e. to compute them, compose them and differentiate them. As we explain in this article, this ability descends from the availability of processors that are highly proficient at matrix multiplications and similar tasks. We thus increasingly do not need to rely on analytic solvability. In probability, finitism was one of the main drivers behind the foundational work by Bruno de Finetti, see for instance (de Finetti 1931) and (Plato 1994)for a historical review. The other drivers were subjectivism and logic. De Finetti grounded Probability Theory upon temporal modal logic, an ideal setting for Finance theory. Predicate logic as a discipline was founded by Aristotle. Aristotelian logic is finitist and reference to infinite objects is only accepted in the form of "potential infinite", i.e. the result of a limiting process involving a sequence of finitary propositions. Thus, within Aristotelian logic any proposition is either true or false and the principle of the excluded middle holds. Modal logic, pioneered by Averroes and medieval scholars, introduces the notion of possible and impossible to supplement those of true and false. Temporal logic adds to this a time dimension. Finitist temporal modal logic is the natural framework upon which legal documents and in particular financial payoffs for derivative contracts are written. Modern pricing system implementations rely on lexical analyzers and payoff languages to interpret contract specifications in terms of temporal modal logic, (S. P. Jones and Seward 2000). It is thus a natural environment upon which to ground the theory of probability with Finance in mind.

De Finetti's breakthrough was to discover that the principle of no arbitrage can be laid at the foundation of probability theory. This basic principle states that if any asset allocation can possibly give rise to a gain with respect to a given benchmark within a certain time period, then it should also possibly give rise to a loss. Notice here the emphasis on possibility, not probability. The no arbitrage principle itself has ancient roots and also goes back to medieval times. Probability arises only as a secondary concept through de Finetti's Fundamental Theorem stating that there is no arbitrage if and only if asset price processes can be represented as discounted expectations of future payoffs with respect to some probability measure. The Theorem derives from the Fundamental Theorem of Linear Inequalities also known as Farkas Lemma (Farkas 1902). In Section 2, we derive this result in a very general case of non-Markov processes over a finite state space, a setting that provides a general representation for finitist temporal modal logic.

As we discuss in Section 4, Markov processes are of special interest as they can be represented by transition probability matrices and thus, numerically, they can be treated by means of full matrix algebra. The basic algorithm we use is fast exponentiation, which allows one to compute transition probability kernels over any time horizon while starting from an elementary transition probability kernel over a short time interval. The fast exponentiation algorithm is at least 2000 years old, see (Knuth 1969) for the history. It has been used in the last decade for applications to cryptography, but there the matrices involved are smaller than the ones we are dealing with. Hardware advances allow us to use this method confortably for kernel calculations even in situations where the number of state variables is in the range 128-1024. In order to be able to form elementary transition probability kernels, one needs to choose an elementary time interval that satisfies the so-called Courant-Friederichs-Lewi condition (or in brief the Courant condition), first derived in (R. Courant and Lewy 1928). The Courant condition we require is equivalent to the stability condition for explicit Euler methods. It has sometimes been referred to as a "curse" as it restricts time intervals to be fairly short, typically measured in a fraction of a day in Finance applications. Semi-implicit methods combined with sparse-matrix methods were developed to allow for longer time steps and were shown to be unconditionally marginally stable, as opposed to being conditionally strongly stable as explicit Euler methods are. However, as we explain in this paper, if modern hardware is used to multiply full matrices directly and we make use of fast-exponentiation, kernels can be evaluated while respecting the Courant bound in a number of iterations that grows only logarithmically with the length of the time step. This is important as by respecting the Courant bound we achieve far greater smoothness in our kernel calculations that we would achieve otherwise by means of unconditionally marginally stable methods. This is reflected in strong convergence estimates to the continuum limit in (Albanese 2007*a*) and (Albanese 2007*b*) for solutions of stochastic differential equations and stochastic integrals in the case of diffusion processes with possibly rough coefficients. In practice, we notice that kernel calculations are stable and robust even in single precision arithmetics at the condition of respecting the Courant bound. This is technically important as single precision floating point arithmetics is highly optimized on our favourite engines for matrix algebra, i.e. GPUs.

The applications on which we focus in this paper are related to the problem of evaluating sensitivities in Monte Carlo price calculations. Since we have full control of transition probability kernels, our task is greatly simplified. To explain the main idea, suppose that one wishes to price a derivative portfolio by generating one million scenarios and then taking an equally weighted average of discounted payoffs. If next one wishes to slightly modify the model and reprice the same portfolio, one does not need to regenerate scenarios and revalue them. It suffices to re-use the previous scenario set at the condition that scenario values are averaged with respect to weights equal to the ratio between the probability for that particular scenario under the new model specification divided by the probability with respect to the original model specification. This technique works remarkably well as we document in this paper by considering several first and second order sensitivities. We consider both sensitivities with respect to model parameters and with respect to initial conditions. The same methodology extends also to the case of importance sampling, although we do not discuss that situation. The method is known in the literature as the likelihood-ratio method and was recently discussed by Chen and Glasserman in a finance context in the paper (Chen and Glasserman 2007). There the method is presented as an alternative to Malliavin calculus, although in fact the two are related. Our treatment differs from the one in Glasserman et al. in that we use long-step Monte Carlo as opposed to a short step version and we make use of fast exponentiation to compute transition probability kernels as opposed to relying on analytic closed form expressions. Results of numerical experiments conclude the paper.

2 The First Fundamental Theorem of Finance

Consider a family of time points $t_i = t_0 + i\delta t$ where t_0 is today's date, δt is a constant step and $i = 0, 1, 2, ...$ is an integer. Consider also a discrete state space $\Lambda = \{0, ...d - 1\}$ where $d \geq 1$. Let $\mathcal{P}(\Lambda)$ denote the set of all paths $\gamma = (\gamma_i)_{i=0,1,...}$ with $\gamma_i \in \Lambda$. γ_i is the state variable of path γ at time t_i.

Definition 1 *If $j \geq 0$ is a non-negative integer, a function $\phi(\gamma, t_i)$ with $\gamma \in \mathcal{P}(\Lambda), i = 0, 1, ...$ is called step-j non-anticipatory if*

$$\phi(\gamma, t_i) = \phi(\gamma', t_i) \tag{1}$$

for all i and whenever $\gamma_k = \gamma'_k$ for all $k \leq i + j$.

Definition 2 *A pathspace $\mathcal{P}(\Lambda, \kappa)$ is characterized by a sequence of incidence matrices taking only values 0 and 1 and given by step-1 non-anticipatory functions $\kappa(\gamma, t_i) \in \{0, 1\}$ such that if $\kappa(\gamma, t_i) = 0$ for some $i \geq 0$ then $\kappa(\gamma, t_k) = 0$ for all $k \geq i$. A path γ belongs to the set $\mathcal{P}(\Lambda, k)$ if $\kappa(\gamma, t_i) = 1$ for all $i \geq 0$.*

Definition 3 *A real valued process adapted to the pathspace $\mathcal{P}(\Lambda, \kappa)$ is given by a real valued step-0 non-anticipatory function $A(\gamma, t_i)$.*

Pricing is carried out relative to a valuation benchmark, also called numeraire.

Definition 4 *A numeraire is a positive valued adapted process $g(\gamma, t_i) > 0$.*

An example of a numeraire is given by the price of a commodity with negligible carry costs and negligible convenience yield such as, for instance, gold. A second example of a numeraire is defined through a positive valued process $r(\gamma, t_i)$ interpreted as a short rate, i.e. the money market account process given by

$$B(\gamma, t_i) = (1 + \delta t r(\gamma_0, t_0)) \quad ... \quad (1 + \delta t r(\gamma_{i-1}, t_{i-1})). \tag{2}$$

Definition 5 *Let $\mathcal{P}(\Lambda, k)$ be a pathspace characterized by the incidence matrices $\kappa(\gamma, t_i)$ and let $g(\gamma, t_i)$ be a numeraire process. Let $A^s(\gamma, t_i)$ be a family of non-anticipatory path functionals indexed by $s = 1, ...M$ with $M > 0$ on the time interval $t_i \in [t_0, t_j]$ where $j > 0$. The family of processes $A^s(\gamma, t_i)$ is called g-coherent if for any $t_i \in [t_0, t_j]$ and any set of coefficients $\zeta^s, s = 1, ...M$, the following property holds: if γ is a possible path for which $\kappa(\gamma, t_i) = 1$ and*

$$\frac{1}{g^s(\gamma, t_{i+1})} \sum_s \zeta^s A^s(\gamma, t_{i+1}) - \frac{1}{g^s(\gamma, t_i)} \sum_s \zeta^s A^s(\gamma, t_i) > 0 \tag{3}$$

then there is a second possible path γ' such that $\gamma'_k = \gamma_k$ for all $k \leq i$ and $\kappa(\gamma', t_i) = 1$ and

$$\frac{1}{g^s(\gamma\prime, t_{i+1})} \sum_s \zeta^s A^s(\gamma\prime, t_{i+1}) - \frac{1}{g^s(\gamma, t_i)} \sum_s \zeta^s A^s(\gamma, t_i) < 0. \tag{4}$$

Definition 6 *An elementary transition probability kernel on $\mathcal{P}(\Lambda, k)$ is a family of 1-step non-anticipatory path functionals $q(\gamma, t_i)$ defined for $i = 0, 1, .., \gamma \in \mathcal{P}(\Lambda, k)$ satisfying the following properties:*

(i) $q(\gamma, t_i) \geq 0$,

(ii) $\sum_{\gamma':\gamma'_k=\gamma_k, k\leq i} q(\gamma', t_i) = 1$,

(iii) $q(\gamma, t_i) > 0$ *if and only if* $\kappa(\gamma, t_i) = 1$*, while otherwise* $q(\gamma, t_i) = 0$.

Definition 7 *Let us fix a time $t_j > t_0$ and let $A(\gamma, t_i), i = 0, ...j$ be a process. The expectation of this process at time t_i prior to t_j with respect to the kernel $q(\gamma, t_i)$ is a process denoted by $E^q_{t_i}[A(\gamma, t_j) \mid \{\gamma_k\}_{k\leq i}]$ and defined as follows:*

$$E^q_{t_i}[A(\gamma, t_j)|\{\gamma_k\}_{k\leq i}] = \sum_{\gamma':\gamma'_k=\gamma_k \forall k\leq i} q(\gamma', t_i) \;....\; q(\gamma', t_{j-1}) A(\gamma', t_j). \tag{5}$$

This formula can be recast in terms of conditional probabilities of a path given by the functional

$$p(\gamma, t_i, t_j) \equiv q(\gamma, t_i) \;....\; q(\gamma, t_{j-1}). \tag{6}$$

In terms of this functional, we can express expectations in (5) as follows:

$$E^q_{t_i}[A(\gamma, t_j)|\{\gamma_k\}_{k\leq i}] = \sum_{\gamma':\gamma'_k=\gamma_k \forall k\leq i} p(\gamma', t_i, t_j) A(\gamma', t_j). \tag{7}$$

The interpretation of $p(\gamma, t_i, t_j)$ is that this is the probability for a path to be equal to γ in the time interval $[t_{i+1}, t_j]$ conditioned to knowing that it coincides with γ on the preceding time interval $[t_0, t_i]$.

Definition 8 *Let $g(\gamma, t_i)$ be a numeraire asset, the adapted process $A(\gamma, t_i)$ is called g-discounted martingale with respect to the elementary kernel $q(\gamma, t_i)$ if*

$$A(\gamma, t_i) = E^q_{t_i}\left[\frac{g(\gamma, t_i)}{g(\gamma, t_j)} A(\gamma, t_j) \middle| \{\gamma_k\}_{k\leq i}\right]. \tag{8}$$

where E^q denotes the expectation with respect to the elementary transition probability kernels $q(\gamma, t_i)$.

As an example, consider the case where the numeraire is given by the money market account $B(\gamma, t_i)$ in (2). Fix a time $t_j > t_0$ and let $A(\gamma, t_j)$ be a process. The discounted expectation of this process at time t_i prior to t_j with respect to the money market account $B(\gamma, t_i)$ can be expressed as follows:

$$E^q_{t_i}\left[\frac{B(\gamma,t_i)}{B(\gamma,t_j)}A(\gamma,t_j)\;\middle|\;\{\gamma_k\}_{k\le i}\right] = \sum_{\gamma':\gamma'_k=\gamma_k\ \forall k\le i} q(\gamma',t_i)\quad ....\quad q(\gamma',t_{j-1})\ \frac{B(\gamma',t_i)}{B(\gamma',t_j)}A(\gamma',t_j). \tag{9}$$

Elementary discounted transition probability kernels defined as

$$q^D(\gamma,t_i) = \frac{1}{1+\delta tr\,(\gamma,t_i)}q(\gamma,t_i) \tag{10}$$

can be used to implicitly account for the numeraire asset in the path expansion for discounted expectations by recasting it as follows:

$$E^q_{t_i}\left[\frac{B(\gamma,t_i)}{B(\gamma,t_j)}A(\gamma,t_j)\middle|\{\gamma_k\}_{k\le i}\right] = \sum_{\gamma':\gamma'_k=\gamma_k\ \forall k\le i} q^D(\gamma',t_i)\quad ....\quad q^D(\gamma',t_{j-1})\ A(\gamma',t_j). \tag{11}$$

Definition 9 *The family of adapted processes $A^s(\gamma,t_i), s=1,..M$ is called a family of g-discounted equivalent martingales if there exists an elementary kernel $q(\gamma,t_i)$ on the path-space $\mathcal{P}(\Lambda,k)$ with respect to which the processes $A^s(\gamma,t_i)$, $s=1,..M$, are all g-discounted martingales.*

Theorem 10 ***(First Fundamental Theorem of Finance)*** *Let $g(\gamma,t_i)$ be a numeraire process and let $A^s(\gamma,t_i), s=1,...M$ be family of processes adapted to $\mathcal{P}(\Lambda,k)$ and defined on the time interval $t_i\in[t_0,t_j]$. Then $A^s, s=1,...M$ is a family of equivalent g-discounted martingales if and only if they are a g-coherent family.*

Proof. This theorem was first stated and proved by Bruno de Finetti in (de Finetti 1931). The proof depends on the following Farkas Lemma (see for instance (Farkas 1902)):

Lemma 11 *(Farkas) Let A be a $n\times m$ matrix and let c be a real non-zero $n-$vector. Then either the primal system:*

$$Ax\ge 0, \qquad and \qquad c^Tx<0 \tag{12}$$

has a solution or the dual system

$$A^Ty=c, \qquad and \qquad y\ge 0 \tag{13}$$

has a solution but never both.

To prove the Fundamental Theorem, assume that $A^s(\gamma,t_i)$ is a g-coherent family of processes. Let $i\ge 0$, $t_i\in[t_0,t_j]$. We need to show that for all $i\ge 0$, there exists a transition probability kernel $q(\gamma,t_i)$ such that

$$q(\gamma,t_i)>0 \text{ if and only if } \ k(\gamma,t_i)=1, \tag{14}$$

with respect to which the processes $A^s(\gamma, t_i)$ are discounted martingales, i.e. are such that

$$\sum_{\gamma' : \gamma'_k = \gamma_k \ \forall k \le i} q(\gamma', t_i) A^s(\gamma', t_{i+1}) \frac{g(\gamma', t_i)}{g(\gamma', t_{i+1})} = A^s(\gamma, t_i) \tag{15}$$

for all $s = 1, ...M$, all $i = 0, ..j$ and all $\gamma \in \mathcal{P}(\Lambda, k)$. This is sufficient as, by iterating this equation one arrives to the discounted martingale condition over arbitrarily long time intervals. It is convenient to recast this last equation (15) as follows:

$$\sum_{\gamma' : \gamma'_k = \gamma_k \ \forall k \le i} q(\gamma', t_i) \left(A^s(\gamma', t_{i+1}) \frac{g(\gamma', t_i)}{g(\gamma', t_{i+1})} - A^s(\gamma, t_i) \right) = 0. \tag{16}$$

Let γ^abe a family of paths where $a = 1, \dots, n$ with the following properties:

(i) they coincide for $t \le t_i$, i.e. $\gamma^a_k = \gamma^b_k$ for all $a, b = 1, \dots, n$ and all $k \le i$;

(ii) they differ at time t_{i+1}, i.e. $\gamma^a_{i+1} \ne \gamma^b_{i+1}$ if $a \ne b$.

(iii) For all $a = 1, \dots, n$ we have that $k(\gamma, t_i) = 1$;

(iv) If y satisfies $k(\gamma, t_i) = 1$ then there is a path γ^a in the family such that $y = \gamma^a_{i+1}$.

Let

$$w^{as} = A^s(\gamma^a, t_{i+1}) \frac{g(\gamma^a, t_i)}{g(\gamma^a, t_{i+1})} - A^s(\gamma^a, t_i). \tag{17}$$

Let $\mathcal{V}$ be a n-dimensional vector space with basis vectors $\mathbf{v}^1, ..., \mathbf{v}^n$. Let $\mathbf{w}^s$ be the vector in $\mathcal{V}$ of components

$$\mathbf{w}^s = \sum_{a=1}^{n} w^{as} \mathbf{v}^a. \tag{18}$$

It suffices to show that there is a vector $\mathbf{q} = (q^a)$ such that $q^a \ge 0$, $\forall a = 1, \dots, n$, $\sum_{a=1}^n q^a = 1$ and

$$\mathbf{q} \cdot \mathbf{w}^s \equiv \sum_{a=1}^{n} q^a w^{as} = 0 \qquad \forall s = 1, \dots, M. \tag{19}$$

The coherence hypothesis can be recast in this language as the statement according to which a vector (ζ^s) satisfies

$$\sum_{s=1}^{M} \zeta^s w^{as} \ge 0, \quad \forall a = 1, ..., n \tag{20}$$

only if it is the zero vector, i.e. $\zeta^s = 0$.

The system

$$\begin{cases} \sum_{a=1}^n q^a = 1 \\ \sum_{a=1}^n q^a w^{as} = 0 \\ q^a \ge 0, \end{cases} \tag{21}$$

can be recast as a primal system of linear inequalities in the standard form $Aq = c$, $q \geq 0$, where

$$\mathbf{A} = \begin{pmatrix} 1 \; 1 \; .... \; 1 \\ \\ W^T \end{pmatrix} \text{ and } c = \begin{pmatrix} 1 \\ 0 \\ ... \\ 0 \end{pmatrix} \tag{22}$$

The corresponding dual system is

$$\xi_{-1} + \sum_{s=1}^{M} w^{as} \xi^s \geq 0, \quad \xi_{-1} < 0, \tag{23}$$

i.e. $\sum_{s=1}^{M} w^{as} \xi^s > 0$. Assuming coherence, the dual system does not have a solution. Hence the primal equation does and this establishes one direction of the theorem. Vice versa, thanks again to Farkas Lemma, if the dual has a solution, i.e. there is no g-coherence, then the processes are not g-discounted martingales. ■

3 Measure Changes and the Second Fundamental Theorem of Finance

The Second Fundamental Theorem compares two models corresponding to different numeraire assets but giving rise to the same prices.

Definition 12 *Two processes are said mutually absolutely continuous if the corresponding two sets of admissible paths $\mathcal{P}(\Lambda, \kappa)$ are equal.*

Notice that if two processes are mutually absolutely continuous then the conditional path probabilities $p(\gamma, t_i, t_j)$ and $p'(\gamma, t_i, t_j)$ for any given path γ and time points $t_i < t_j$ are either simultaneously positive or simultaneously zero. This motivates introducing the following notion of derivative:

Definition 13 *If two processes are mutually absolutely continuous and correspond to the path probability densities $p(\gamma, t_i, t_j)$ and $p'(\gamma, t_i, t_j)$, then the Radon-Nykodim derivative of one with respect to the other is defined as the path functional*

$$w(\gamma, t_i, t_j) = \frac{p(\gamma, t_i, t_j)}{p'(\gamma, t_i, t_j)}. \tag{24}$$

whenever the path γ is admissible and as zero otherwise.

Theorem 14 ***(Second Fundamental Theorem of Finance)*** *Let $A(\gamma, t_i)$ be an asset price process on the time interval $t_i \in [t_0, t_j]$. Let $g(\gamma, t_i)$ be a numeraire asset price processes and let $q(\gamma, t_i)$ be a transition probability kernel associated to it. If $g'(\gamma, t_i)$ is a second numeraire asset price process, then a transition probability kernel consistent with it is given by the equation*

$$q'(\gamma, t_i) = \frac{g'(\gamma, t_{i+1})}{g'(\gamma, t_i)} \frac{g(\gamma, t_i)}{g(\gamma, t_{i+1})} q(\gamma, t_i). \tag{25}$$

for all $\gamma \in \mathcal{P}(\Lambda, k)$ and defined as zero otherwise.

The kernel q' describes a process which is mutually absolutely continuous with respect to the process defined by q. The Radon-Nykodim derivative between these two processes is given by

$$w(\gamma, t_i, t_j) = \frac{g(\gamma, t_i)}{g(\gamma, t_j)} \frac{g'(\gamma, t_j)}{g'(\gamma, t_i)}. \tag{26}$$

Proof. Since $g(\gamma, t_i)$ is a numeraire asset price process, i.e. $g(\gamma, t_i) > 0$ is a strictly positive asset price process, we have that

$$g(\gamma, t_i) = \sum_{\gamma' : \gamma'_k = \gamma_k \forall k \leq i} q(\gamma', t_i) \ ... \ q(\gamma', t_{j-1}) g(\gamma', t_j). \tag{27}$$

This equation implies that

$$\sum_{\gamma' : \gamma'_k = \gamma_k \forall k \leq i} q(\gamma', t_i) \ ... \ q(\gamma', t_{j-1}) \frac{g(\gamma', t_j)}{g(\gamma, t_i)} = 1. \tag{28}$$

Hence, the step-1 non-anticipatory functional

$$q'(\gamma, t_i) = \frac{g'(\gamma, t_{i+1})}{g'(\gamma, t_i)} \frac{g(\gamma, t_i)}{g(\gamma, t_{i+1})} q(\gamma, t_i) \tag{29}$$

is a transition probability kernel defining a process equivalent to the one of kernel $q(\gamma, t_i)$. The inverse relation

$$q(\gamma, t_i) = \frac{g(\gamma, t_{i+1})}{g(\gamma, t_i)} \frac{g'(\gamma, t_i)}{g'(\gamma, t_{i+1})} q'(\gamma, t_i) \tag{30}$$

allows one to express the discounted expectation of the future value of an asset by Definition 5 as follows:

$$\begin{aligned} A(\gamma, t_i) &= E^q_{t_i}\left[\frac{g(\gamma, t_i)}{g(\gamma, t_j)} A(\gamma, t_j) \middle| \{\gamma_k\}_{k \leq i} \right] \\ &= \sum_{\gamma' : \gamma'_k = \gamma_k \forall k \leq i} q(\gamma', t_i) \ ... \ q(\gamma', t_{j-1}) \frac{g(\gamma', t_i)}{g(\gamma', t_j)} A(\gamma', t_j) \\ &= \sum_{\gamma' : \gamma'_k = \gamma_k \forall k \leq i} q'(\gamma', t_i) \ ... \ q'(\gamma', t_{j-1}) \frac{g'(\gamma', t_i)}{g'(\gamma', t_j)} A(\gamma', t_j) \\ &= E^{q'}_{t_i}\left[\frac{g'(\gamma, t_i)}{g'(\gamma, t_j)} A(\gamma, t_j) \middle| \{\gamma_k\}_{k \leq i} \right] \end{aligned} \tag{31}$$

■

A useful variant of the Second Fundamental Theorem which is of key importance for the intent of article comes about when one considers two mutually absolutely continuous processes under the same specification for the numeraire process.

Theorem 15 *Consider two mutually absolutely continuous processes with elementary kernels $q(\gamma, t_i)$ and $q'(\gamma, t_i)$ and Radon-Nykodim derivative $w(\gamma, t_i, t_j)$.*

If $f(\gamma, t_i, t_j)$ is a path functional depending on the values attained by the path γ in the time interval $[t_i, t_j]$, then

$$E^q_{t_i}\left[f(\cdot, t_i, t_j) \mid \{\gamma_k\}_{k\le i}\right] = E^{q'}_{t_i}\left[f(\cdot, t_i, t_j)W(\cdot, t_i, t_j) \mid \{\gamma_k\}_{k\le i}\right]. \tag{32}$$

Proof. By the Definition 7, the expectation of $f(\cdot)$ at time zero with respect to the kernel $q(\gamma, t_i)$ is defined as follows:

$$\begin{aligned}
& E^q_{t_i}\left[f(\cdot, t_i, t_j) \mid \{\gamma_k\}_{k\le i}\right] \\
= \;& \sum_\gamma q(\gamma, t_i) \,....\, q(\gamma, t_{j-1}) f(\gamma, t_i, t_j). \\
= \;& \sum_\gamma q'(\gamma, t_i) \,....\, q'(\gamma, t_{j-1}) \frac{q(\gamma, t_i) \,....\, q(\gamma, t_{j-1})}{q'(\gamma, t_i) \,....\, q'(\gamma, t_{j-1})} f(\gamma, t_i, t_j) \\
= \;& E^{q'}_{t_i}\left[f(\cdot, t_i, t_j)w(\cdot, t_i, t_j) \mid \{\gamma_k\}_{k\le i}\right]
\end{aligned} \tag{33}$$

■

4 Markov Processes

Markov processes are a particularly important special class of processes characterized by the fact that transition probability kernels are independent of the past values attained by the path γ. More precisely, elementary transition probability kernels for a Markov process have the special form

$$q(\gamma, t_i) = u_{\delta t}(\gamma_i, \gamma_{i+1}; t_i) \tag{34}$$

where $u_{\delta t}(y_1, y_2; t_i)$ is a function of $y_1, y_2 \in \Lambda$ and time t_i. The Markov generator or Markovian is given by the matrix $\mathcal{L}(y_1, y_2; t_i)$ such that

$$u_{\delta t}(y_1, y_2; t_i) = \delta_{y_1, y_2} + \delta t \mathcal{L}(y_1, y_2; t_i) \tag{35}$$

for all $y_1, y_2 \in \Lambda$ and all t_i, $i = 0, 1, \ldots$. Here

$$\delta_{y_1,y_2} = \begin{cases} 1 & \text{if } y_1 = y_2 \\ 0 & \text{if } y_1 \neq y_2 \end{cases} \tag{36}$$

is the so called Kronecker Delta. The constraints in Definition 6 imply the following two conditions on the matrix $\mathcal{L}(y_1, y_2; t_i)$:

(i)

$$\mathcal{L}(y_1, y_2; t_i) \ge 0 \quad \text{for all } y_1 \neq y_2 \text{ and all } t_i; \tag{37}$$

(ii)

$$\sum_{y_2} \mathcal{L}(y_1, y_2; t_i) = 0 \text{ for all } y_1 \text{ and all } t_i. \tag{38}$$

Due to condition (i) and (ii), the diagonal matrix elements

$$\mathcal{L}(y_1, y_1; t_i) = -\sum_{y_2 \neq y_1} \mathcal{L}(y_1, y_2; t_i) \leq 0 \tag{39}$$

are non-positive. In order to ensure positivity of the diagonal elements of the elementary kernel, we thus need to postulate the following condition:

(iii)

$$\delta t \leq \frac{1}{\max_y |\mathcal{L}(y, y; t_i)|}. \tag{40}$$

This condition is called the Courant condition.

Definition 16 *A matrix $\mathcal{L}(y_1, y_2; t_i)$ satisfying the properties (i) (ii) is called Markov matrix or Markovian.*

In practice, one builds elementary transition probability kernels starting from a Markovian, given which one finds the elementary time interval $\delta t > 0$ (typically one day or a fraction of a day) satisfying the Courant condition.

As a matter of terminology, we distinguish between operators and kernels. A kernel is a matrix $A(x, y)$ with indices x, y taking up a finite set of values. In the previous example $x, y = 0, ...d - 1$. A vector is instead represented by an array $v(x)$. The matrix $A(x, y)$ can also be put in relation with an operator A that transforms vectors linearly, so that

$$(Av)(x) = \sum_y A(x, y)v(y). \tag{41}$$

Vice versa, to each operator that transforms vectors linearly there corresponds a matrix. In fact, if one considers the vector $\delta_y(x) = \delta(x - y)$, one finds

$$(A\delta_y)(x) = A(x, y). \tag{42}$$

Given an operator A, the matrix $A(x, y)$ is called the *kernel of A*.

A *transition probability kernel* over finite time intervals is given by a two-parameter family of matrices $u(y_1, t_1; y_2, t_2)$ dependent on the time coordinates $t_1 \leq t_2$ and representing the transition probabilities from state y_1 at time t_1 to state y_2 at time t_2. For fixed t_1 and t_2, these are the transition probability kernels. The operator corresponding to a transition probability kernel is called *propagator*.

Given the family of elementary transition probability kernels $u_{\delta t}(y_1, y_2; t_i)$ for a Markov process, one can compute more general transition probability kernels $u(y_1, t_i; y_2, t_j)$ over arbitrary time intervals $[t_i, t_j]$ with $i < j$. In the particular case of a time step twice the size of δt from the equation (6), we find

$$u(\gamma_i, t_i; \gamma_{i+2}, t_{i+2}) = \sum_{\gamma_{i+1}} u_{\delta t}(\gamma_i, \gamma_{i+1}; t_i)\, u_{\delta t}(\gamma_{i+1}, \gamma_{i+2}; t_{i+1}) \tag{43}$$

Remarkably, this law is the same as the standard rule for multiplying matrices rows by columns. This is perhaps the single most noteworthy property of Markov processes which allows one to reduce problems in probability theory to linear algebra.

In matrix language, the equation above can be recast as follows:

$$u\left(t_i;t_{i+2}\right)=u_{\delta t}\left(t_i\right)u_{\delta t}\left(t_{i+1}\right). \tag{44}$$

For two generic time indices $i<j$, we have that

$$u\left(t_i;t_j\right)=u_{\delta t}\left(t_i\right)\cdots u_{\delta t}\left(t_{j-1}\right). \tag{45}$$

The path-wise representation for the latter formula is

$$u\left(\gamma_i,t_i;\gamma_j,t_j\right)=\sum_{\gamma:\gamma_i\to\gamma_j}u_{\delta t}\left(\gamma_i,\gamma_{i+1};t_i\right)\cdots u_{\delta t}\left(\gamma_{j-1},\gamma_j;t_{j-1}\right) \tag{46}$$

where the sum is over all paths $\gamma=\{\gamma_i,\ldots,\gamma_j\}$, γ_k is a state variable and $k=i,\ldots j$.

The incidence matrix κ is characterized by the set of all admissible set $\mathcal{P}(\Lambda,\kappa)$. In turn, admissible paths are characterized by the condition

$$u\left(\gamma_i,t_i;\gamma_{i+1},t_{i+1}\right)>0. \tag{47}$$

being satisfied for all $i\geq 0$.

5 Lattice Models

In a lattice model, one typically needs to design a process by giving time dependent Markovians and then finding propagators. To facilitate the numerical analysis, it is very useful to assume that Markovians are piecewise constant. To generate Monte Carlo scenarios one is then faced with the problem of obtaining propagators between pairs of time points which are relevant for the payoff at hand to evaluate.

The relevant times for the payoff and the times at which Markovian parameters change in general do not coincide. This leads us to considering the following three sequences of time points:

- Let the fixing date be zero by convention and let $T_i, i=0,\ldots N_i-1$, be an increasing sequence of time points in the time interval $(0,T]$ such that the generator $\mathcal{L}\left(y_1,y_2;t\right)$ is piecewise constant and equal to the matrix $\mathcal{L}_i\left(y_1,y_2\right)$ on each time interval $[T_{i-1},T_i)$ for $i=0\ldots N_i-1$, where $T_{-1}=0$.
- Let $T'_k, k=0,\ldots N_k-1$ be the times which enter in the payoff definition and at which one needs to evaluate scenarios.
- Let $\bar{T}_q$, $q=0,\ldots N_q-1$, be the union of the time points T_i and T'_k arranged in increasing order.

Propagators need to be obtained over each subinterval $[\bar{T}_{q-1},\bar{T}_q)$, for $q=0,\ldots N_q-1$, where by convention we set $\bar{T}_{-1}=0$. To compute the propagator $u\left(y_{q-1},\bar{T}_{q-1};y_q,\bar{T}_q\right)$ over each time period $[\bar{T}_{q-1},\bar{T}_q)$ we exploits time-homogeneity by making use of the following algorithm called *fast exponentiation*. The algorithm applies in case one wishes to computer a power of the form 2^n where n is an integer and consists of the following iteration:

$$\begin{aligned} u_{2\delta t} &= u_{\delta t} \cdot \; u_{\delta t} \\ u_{4\delta t} &= u_{2\delta t} \cdot u_{2\delta t} \\ &\vdots \\ u_{2^n\delta t} &= u_{2^{n-1}\delta t} \cdot u_{2^{n-1}\delta t} = u_{\Delta T} \end{aligned} \tag{48}$$

Let $(\delta t)_q$ be an elementary period of the form $(\delta t)_q = 2^{-n_q}(\Delta T)_q$ for some integer n_q, where $(\Delta T)_q = (\bar{T}_q - \bar{T}_{q-1})$. The integer n is chosen so large that

$$(\delta t)_q \leq \frac{1}{\max\limits_y \left|\mathcal{L}_{i(q)}\left(y,y\right)\right|} \quad \text{for all } q = 0, ...N_q - 1. \tag{49}$$

By means of fast exponentiation, one can evaluate each of the transition probability kernels

$$\begin{aligned} u\left(y_{q-1}, \bar{T}_{q-1}; y_q, \bar{T}_q\right) &= \left(1 + (\delta t)_q \mathcal{L}_{i(q)}\right)^{\frac{(\Delta T)_q}{(\delta t)_q}} \left(y_{q-1}, y_q\right) \\ &= \left(1 + (\delta t)_q \mathcal{L}_{i(q)}\right)^{2^n} \left(y_{q-1}, y_q\right). \end{aligned} \tag{50}$$

Finally, by multiplying such matrices we arrive at the transition probability kernels

$$U_k(y_{k-1}, y_k) \equiv u\left(y_{k-1}, T'_{k-1}; y_k, T'_k\right). \tag{51}$$

Notice that

$$\lim_{\delta t \downarrow 0} \left(1 + (\delta t)_q \mathcal{L}_{i(q)}\right)^{\frac{(\Delta T)_q}{\delta t}} = e^{(\Delta T)_q \mathcal{L}_{i(q)}}. \tag{52}$$

This is a consequence of Neper's formula that reads as follows for a generic complex number $z \in \mathbb{C}$:

$$e^z = \lim_{n\to\infty} \left(1 + \frac{z}{n}\right)^n \quad . \tag{53}$$

The limit in Neper's formula defines an entire analytic function admitting the Taylor expansion

$$e^z = \sum_{j=0}^{\infty} \frac{z^j}{j!}. \tag{54}$$

The exponential of a matrix $(\Delta T)_q \mathcal{L}_{i(q)}$ is similarly given also by a Taylor expansion,i.e.

$$e^{(\Delta T)_q \mathcal{L}_{i(q)}} = \sum_{j=0}^{\infty} \frac{\left((\Delta T)_q \mathcal{L}_{i(q)}\right)^j}{j!}, \tag{55}$$

as follows from the fact that powers of a given matrix are mutually commutative, i.e.

$$\mathcal{L}^m_{i(q)} \mathcal{L}^n_{i(q)} = \mathcal{L}^n_{i(q)} \mathcal{L}^m_{i(q)}. \tag{56}$$

By the same token, one derives Neper's formula above. Notice that the commutativity argument hinges upon the fact that the generator $\mathcal{L}_{i(q)}$ is constant as a function of time over the interval $[\bar{T}_{q-1}, \bar{T}_q)$.

6 Likelihood Ratio Method

Consider two models described by the pair of Markovians $\mathcal{L}_i$ and $\mathcal{L}'_i$, respectively, which are constant over the same collection of time intervals $[T_{i-1}, T_i)$. The likelihood ratio method allows one to evaluate an expectation with respect to one model by using scenarios generated according to the transition probabilities specific to the other one.

Consider a payoff function $f(\gamma)$ which depends on the values attained by the path γ at the time points $T'_k, k = 0, 1, ...N_k - 1$. To carry out a Monte Carlo simulation we thus need to evaluate the transition probability kernels of the sequence of time intervals $[T'_{k-1}, T'_k)$. Consider the set of long-step paths Γ consisting of sequences of state variables $\Gamma = \{\Gamma_0, ...\Gamma_{N_k-1}\}$, where Γ_k is the state variable at time T'_k. With slight abuse of notation, we write $f(\gamma) = f(\Gamma)$. The probability for the path Γ to occur is given by

$$P(\Gamma) = U_0\left(\Gamma_{-1}, \Gamma_0\right) U_1\left(\Gamma_0, \Gamma_1\right) \cdots U_{N_k-1}\left(\Gamma_{N_k-2}, \Gamma_{N_k-1}\right) \tag{57}$$

where Γ_{-1} is the value of the path Γ at time $T'_{-1} = 0$.

The expectation of the payoff function $f(\gamma)$ by Definition 7 is

$$\begin{aligned} E_0^u\left[f(\cdot)\right] &= \sum_\gamma u_{\delta t}(\gamma_0, \gamma_1; t_0) \cdots u_{\delta t}(\gamma_{j-1}, \gamma_j, t_{j-1})\, f(\gamma) \\ &= \sum_\Gamma U_0\left(\Gamma_{-1}; \Gamma_0\right) U_1\left(\Gamma_0, \Gamma_1\right) \cdots U_{N_k-1}\left(\Gamma_{N_k-2}, \Gamma_{N_k-1}\right) f(\Gamma) \\ &= \sum_\Gamma P(\Gamma) f(\Gamma) \end{aligned} \tag{58}$$

Lemma 17 *If u and u' are the elementary transition probability kernels for two Markov processes which happen to be mutually absolutely continuous, then the corresponding path-probability functions $P(\Gamma)$ and $P'(\Gamma)$ are either simultaneously positive or simultaneously zero, i.e. the corresponding discrete time processes of kernels U and U' are mutually absolutely continuous.*

Proof. Let's notice that

$$P(\Gamma) = \sum_{\gamma : \Gamma_k = \gamma_{T'_k} \forall k \geq -1} p(\gamma). \tag{59}$$

Hence if $P(\Gamma) > 0$ for some long-step path Γ, then there is a path $\gamma \in \mathcal{P}(\Lambda, \kappa)$ such that $\Gamma_k = \gamma_{T'_k}$ for all $k = -1, ...N_{k-1}$, for which $p(\gamma) > 0$. Since u and u' are mutually absolutely continuous, we also have that $p'(\gamma) > 0$. But then also $P'(\Gamma) > 0$. ■

This lemma does not have a converse. If the long-step processes described by the kernels U and U' are mutually absolutely continuous, it may well be that the short step processes are not. In the following section, we discuss situations of practical relevance for which this precisely the case.

Definition 18 *If U and U' are the long-step transition probability kernels for two mutually absolutely continuous Markov processes at the time points $T'_k, k = 0, 1, ...N_k - 1$, then one defines the long-step Radon-Nykodym derivative as follows:*

$$W(\Gamma) = \frac{P(\Gamma)}{P'(\Gamma)} = \frac{U_0\left(\Gamma_{-1}, \Gamma_0\right) U_1\left(\Gamma_0, \Gamma_1\right) \cdots U_{N_k-1}\left(\Gamma_{N_k-2}, \Gamma_{N_k-1}\right)}{U'_0\left(\Gamma_{-1}, \Gamma_0\right) U'_1\left(\Gamma_0, \Gamma_1\right) \cdots U'_{N_k-1}\left(\Gamma_{N_k-2}, \Gamma_{N_k-1}\right)} \tag{60}$$

Theorem 19 *If U and U' are the long-step transition probability kernels for two mutually absolutely continuous Markov processes at the time points $T'_k, k = 0, 1, ...N_k - 1$, then*

$$E_0^u[f(\cdot) \mid \bar{y}] = \quad E_0^{u'}[f(\cdot)W(\cdot) \mid \bar{y}] \tag{61}$$

where $W(\Gamma)$ denotes the long-step Radon-Nykodym derivative between the two processes and $\bar{y}$ is the initial state variable.

This result finds two types of applications. The first is to the calculation to sensitivities with respect to model parameters where u' is a small perturbation of the elementary kernel u and we are interested in determining the rate of change of the discounted expectation. The second application is to importance sampling, where we aim at evaluating the discounted expectation of a payoff sensitive to rare events and to increase sampling we opt to generate scenarios according to a process different from the base process in order to produce a greater number of relevant events. This of course necessitates the introduction of weights to compensate for the process miss-specification.

A variant of this situation occurs when we are interesting in changing not the process but the initial condition.

Theorem 20 *Consider two state variables y_1, y_2 and a long-step kernel U. Suppose that*

$$U_0\left(y_1, y_3\right) > 0 \tag{62}$$

if and only if y_3 is such that

$$U_0\left(y_2, y_3\right) > 0. \tag{63}$$

Define the weight

$$Q(y_1, y_2; \Gamma) = \frac{U_0\left(y_2, \Gamma_0\right)}{U_0\left(y_1, \Gamma_0\right)}. \tag{64}$$

Then

$$E_0^u[f(\cdot) \mid y_2] = \quad E_0^u[f(\cdot)Q(y_1, y_2; \cdot) \mid y_1]. \tag{65}$$

7 Monte Carlo Pricing

A derivative instrument is characterized by a sequence of cash flows contingent to a realized path γ. Cash flows are given by a step-0 non anticipatory functional $\phi(\gamma, t)$. Typically, cash flows depend only on the values achieved by the underlying process on a subset $T'_k, k = 0, ...N_{k-1}$ of dates, i.e. we can interpret the path functional $\phi(\gamma, t)$ as a functional on long-step paths $\Phi(\Gamma, t)$, whereby $\Phi(\Gamma, T'_k)$ is the cash flow occurring at time T'_k.

Prices are given by discounted expectations. To account for the discount term, it is convenient to embed it directly into discounted elementary transition kernels u^D according to equation (10). Long step discounted kernels U^D are evaluated by fast-exponentiation of discounted transition probability kernels. A discounted expectation of the payoff functional $\Phi(\Gamma, t)$ is evaluated as

$$\sum_k \sum_\Gamma U^D(\Gamma_{-1}, \Gamma_0)...U^D(\Gamma_{k-1}, \Gamma_k)\Phi(\Gamma, T'_k). \tag{66}$$

A discounted payoff functional is defined as follows

$$\Phi^D(\Gamma, T'_k) = \frac{U^D(\Gamma_{-1}, \Gamma_0)...U^D(\Gamma_{k-1}, \Gamma_k)}{U(\Gamma_{-1}, \Gamma_0)...U(\Gamma_{k-1}, \Gamma_k)} \Phi(\Gamma, T'_k). \tag{67}$$

In terms of this discounted functional, the discounted expectation in equation (66) can simply be expressed as

$$E^U \left[\Phi^D(\Gamma, T'_k) \right] = \sum_k \sum_\Gamma U(\Gamma_{-1}, \Gamma_0)...U(\Gamma_{k-1}, \Gamma_k) \Phi^D(\Gamma, T'_k). \tag{68}$$

under the un-discounted measure U. Since the kernel U is a proper transition probability kernel, such an expression can be valued using Monte Carlo methods.

In the important special case of models whereby we use the money market account as the numeraire asset and whereby interest rates are deterministic, the ratio $\Phi^D(\Gamma, T'_k)/\Phi(\Gamma, T'_k)$ is independent of the path Γ and can be easily evaluated as the inverse of the money market account value

$$B(t) = e^{r_i(t-T_i)} \prod_{i:T_i<t} e^{r_i(T_i-T_{i-1})} \tag{69}$$

evaluated at $t = T'_k$, i.e.

$$\Phi^D(\Gamma, T'_k) = B(T'_k)^{-1} \Phi(\Gamma, T'_k). \tag{70}$$

8 Generation of Monte Carlo Scenarios on CPUs and GPUs

The actual implementation of a Monte Carlo pricing algorithm is based on drawing uniformly distributed pseudo-random deviates in the interval $[0, 1]$ and inferring scenario paths based on them.

There are several pseudo-random generations algorithms in the literature and broadly used software libraries. In the case study discussed in this paper we use a family of Matsumoto's Mersenne Twister algorithms of period 2^{2203}, one for each execution thread, see (Matsumoto 1998). We have two implementations, one for CPUs and another for GPUs. On the CPU, we make use of pseudo-random numbers generated by means of the SIMD version of the algorithm described in (Matsumoto 2007-2009) as implemented in the Intel Math Kernel libraries. The subroutine `viRngUniformBits` provides a stream of random bits evaluated 128 bits at a time by means of SSE2 instructions. The algorithm is particularly effective on 64 bit chips. We read 32 bit pseudo-random integers out of the sequence and convert them into uniformly distributed, double precision deviates by dividing by 2^{32}. On the GPU, we make use of the 32-bit version of the same algorithm implemented by Podlozhnyuk in the nVidia CUDA SDK. In this case, random bit sequences are produced 32 bits at a time and the converted into single precision floating point numbers by dividing by 2^{32}.

In each time period $[T'_{k-1}, T'_k)$, we define the cumulative transition probability kernel $C_k(y_1, y_2)$ as follows:

$$C_k(y_1, y_2) = \sum_{z=0}^{y_2} U(y_1, z) \tag{71}$$

The starting point of each path in the first time period $[T'_{-1}, T'_0)$ is a fixed lattice site y_0 corresponding to the spot state variable. To generate sample paths of the form $\Gamma_k, k = 0, 1 \ldots, N_k - 1$ for the state variables, we generate N_k deviates ξ_k uniformly distributed in $[0, 1]$. If

$$\xi_k \leq C_k (\Gamma_{k-1}, 0) \tag{72}$$

then we set

$$\Gamma_k = 0. \tag{73}$$

If, instead,

$$\xi_k > C_k (\Gamma_{k-1}, d - 1) \tag{74}$$

then we set

$$\Gamma_k = d - 1. \tag{75}$$

Otherwise, Γ_k is determined as the site such that

$$C_k (\Gamma_{k-1}, \Gamma_k) < \xi_k \leq C_k (\Gamma_{k-1}, \Gamma_k + 1). \tag{76}$$

At the $k-$th time interval, to find the state variable Γ_k that solves this constraint, one can use a binary search.

In our experience, the optimal implementation on GPUs and CPUs differ quite radically because of the different memory architectures and threading models. On CPUs it is very useful to speed up the search by using a hash table of pre-computed upper and lower bounds. A hash table is based on a discretization of the interval $[0, 1]$ in d sub-intervals of equal size. Let $z = 0, ...d - 1$ be an index for these sub-intervals. For each z, one finds an array of lower bounds $lb(z)$ and an array of upper bounds $ub(z)$ with values in $\{0, 1, ...d - 1\}$, so that

$$C_k (\Gamma_{k-1}, lb(z)d\ ) \leq \frac{z}{d}\ \leq\ \xi\ <\ \frac{z+1}{d} \leq C_k (\Gamma_{k-1}, ub(z)d\ ) \tag{77}$$

for every $\xi \in \left[\frac{z}{d}, \frac{z+1}{d}\right]$. Since the lattice size is practically restricted to be far less then 2^{16}, the upper and lower bounds can be stored as short 16-bit integers and jointly stored in a 32-bit integer as follows:

$$h(z) = lb(z) + 2^{16} ub(z) \tag{78}$$

Once we draw a pseudo-random deviate and find that $ub(z) - lb(z) > 1$, we use a binary search between $ub(z)$ and $lb(z)$ and set $\Gamma_k = ub(z)$. Otherwise, if $ub(z) - lb(z) <= 1$, no iteration is needed and we still set $\Gamma_k = ub(z)$. An exception is represented by the case when $lb(z) = 0$, in which case we need to control whether the condition in (72) is realized; if so, we set $\Gamma_k = 0$.

On GPUs, in our experience the use of hash tables along the lines we describe does not improve performance. The reason is that hash table need to be stored in global memory and be retrieved through random un-coalesced access. This operation is notoriously time expensive as it needs to be performed sequentially by each individual thread in a thread-block and costs about 180 clock cycles. On GPUs it is thus worthwhile to execute a binary search taking as initial lower and upper pivots the state variables 0 and $d - 1$. Furthermore, it is worth to design the search in such a way that the maximum number of iterations is executed so that the `for` loop can be unrolled, thus avoiding an expensive control for the possibility of early branching.

On CPUs access to hash tables and cumulative transition probability kernels can be further sped up by properly organizing the scenario simulation. At a higher level we partition the job in batches. At a lower level we need to execute too loops: a loop over scenarios and a loop over calendar time as indexed by the variable k. The order matters. On CPUs, it is useful to select as the innermost loop the one over scenarios in a given batch and then iterate over the time steps. This way, over each time interval one accesses the same transition probability kernel and the operating system has a change to page the corresponding memory into level-3 cache. On current CPUs, one finds about 2 MB of cache per core while a 512^2 transition probability kernel and hash table take 1 MB each. Thus they fit snugly into level-3 cache. On GPUs there is no cache and as a consequence this argument does not apply. On the contrary, it is worthwhile choosing as the innermost loop the one over the time coordinate as in this case, at the $k-$th time period one can store the value Γ_{k-1} in registers and have it immediately available, thus avoiding two expensive global memory transactions.

The performance ratios we observe between GPUs and CPUs is about a factor three, both considering 2008 hardware (Tesla 860 and quad-core Xeon 8460) and 2009 hardware (Tesla 1060 and 16-core Xeon 8600 Nehamel). The code is published as an open source distribution available at www.albanese.co.uk.

Microchip	Mersenne Twister	Model with $d = 128$	Model with $d = 512$
Xeon 5460	602 ME/s	170 ME/s	169 ME/s
Nehamel	849 ME/s	593 ME/s	529 ME/s
Tesla 860C	1528 ME/s	121 ME/s	95 ME/s
Tesla 1060C	2327 ME/s	200 ME/s	196 ME/s

Table 1: Performance of CPUs and GPUs at scenario generation in million evaluations per second. The Xeon 5460 processor used is by Intel, has 4 cores and a frequency of 3.16MHz. The Nehamel processor is also by Intel, has 16 cores at 2.4 GHz. The Tesla 860C and the Tesla 1060C are chips by nVidia with 128 and 240 cores, respectively, running at 1.3 GHz. Performances are independent of model specification.

As explained, algorithms for kernel valuation rely on fast exponentiation. Here, the situation is reversed with respect to the scenario generation case and GPUs offer a greater range of possibilities for optimization. We find that one can achieve highly efficient performance by implementing fourth level BLAS extension subroutines operating on tensors. In particular, fast exponentiation benefits by performing matrix multiply operations concurrently. This is achieved by fourth level BLAS extensions also introduced in OPLib. In particular, OPLib contains the subroutines `SGEMM4` for general matrix-matrix multiplication for a given array of matrix pairs and the subroutine `SSQMM` for evaluating the second power of an array of square matrices. Also important for applications to pricing theory are the routines `SSGEMV4` that accomplishes multiplications between a given array of matrices and a number of vectors for each input matrix. CPU side we were so far unable to take advantage of fourth level extensions of the BLAS. Performance for kernel valuations are given by the following table:

Microchip	SGEMM	SGEMM4	SGEMV	SGEMV4	d = 128	d = 512
Xeon 5460	84.7 GF/s	84.7 GF/s	3.1 GF/s	3.1 GF/s	9.3 GF/s	14.7 GF/s
Nehamel	63.7 GF/s	63.7 GF/s	3 GF/s	3 GF/s	6.8 GF/s	11.7 GF/s
Tesla 860C	186 GF/s	200 GF/s	3 GF/sec	151 GF/s	105 GF/s	186 GF/s
Tesla 1060C	341 GF/s	369 GF/s	2.6 GF/s	288 GF/s	256 GF/s	256 GF/s

Table 2: Performance of CPUs and GPUs at kernel valuation.

9 Sensitivities

Pricing sensitivities with respect to small changing of initial conditions or model parameters, also known as Greeks, are vital tools in risk management. Greeks give information on hedge ratios needed to immunize portfolios with respect to market risk.

Below, we make use of the following notations for forward and backward finite difference:

$$\frac{\nabla^+\Phi}{\nabla^+X}(y) = \frac{\Phi(y+1)-\Phi(y)}{X(y+1)-X(y)} \tag{79}$$

and

$$\frac{\nabla^-\Phi}{\nabla^-X}(y) = \frac{\Phi(y-1)-\Phi(y)}{X(y-1)-X(y)} \tag{80}$$

and the symmetric second derivative

$$\frac{\Delta\Phi}{\Delta X}(y) = \frac{2}{\nabla^+X-\nabla^-X}\left(\frac{\nabla^+\Phi}{\nabla^+X}(y) - \frac{\nabla^-\Phi}{\nabla^-X}(y)\right). \tag{81}$$

9.0.1 Delta

Delta is the price sensitivity with respect to the underlying instrument spot price. If $\Phi^D(\Gamma, t)$ is the discounted cash flow functional and assuming for notational simplicity that there is a single cash-flow at time $T = T'_{N_k-1}$, the Delta is given by

$$\begin{aligned} Delta &= \frac{\nabla^+E_0^U\left[\Phi^D(\Gamma,T)\right]}{\nabla^+X}(\bar{y}) \\ &= \frac{E_0^U\left[\Phi^D(\Gamma,T)|\bar{y}+1\right] - E_0^U\left[\Phi^D(\Gamma,T)|\bar{y}\right]}{X(\bar{y}+1)-X(\bar{y})} \\ &= \frac{E_0^U\left[\Phi^D(\Gamma,T)Q^+(\Gamma)|\bar{y}\right] - E_0^U\left[\Phi^D(\Gamma,T)|\bar{y}\right]}{X(\bar{y}+1)-X(\bar{y})} \\ &= \frac{E_0^U\left[\Phi^D(\Gamma,T)\left(Q^+(\Gamma)-1\right)|\bar{y}\right]}{X(\bar{y}+1)-X(\bar{y})} \end{aligned} \tag{82}$$

where

$$Q^+(\Gamma) = \frac{U_0(\bar{y}+1,\Gamma_0)}{U_0(\bar{y},\Gamma_0)}. \tag{83}$$

9.0.2 Gamma

Gamma is the second derivative of the value function with respect to the underlying price and measures also the rate of change in the Delta and is defined as follows:

$$\begin{aligned} Gamma &= \frac{\Delta E_0^U\left[\Phi^D(\Gamma,T)\right]}{\Delta X}(\bar{y}) \\ &= \frac{2}{\nabla^+ X - \nabla^- X}\left(\frac{\nabla^+ E_0^U\left[\Phi^D(\Gamma,T)\right]}{\nabla^+ X}(\bar{y}) - \frac{\nabla^- E_0^U\left[\Phi^D(\Gamma,T)\right]}{\nabla^- X}(\bar{y})\right) \\ &= \frac{2}{X(\bar{y}+1) - X(\bar{y}-1)}\left(\frac{E_0^U\left[\Phi^D(\Gamma,T)\left(Q^+(\Gamma)-1\right)|\bar{y}\right]}{X(\bar{y}+1)-X(\bar{y})} - \frac{E_0^U\left[\Phi^D(\Gamma,T)\left(Q^-(\Gamma)-1\right)|\bar{y}\right]}{X(\bar{y}-1)-X(\bar{y})}\right) \end{aligned} \tag{84}$$

where $Q^+(\Gamma)$ is given by (83) and

$$Q^-(\Gamma) = \frac{U_0(\bar{y}-1,\Gamma_0)}{U_0(\bar{y},\Gamma_0)} \tag{85}$$

9.0.3 First Order Sensitivities with Respect to Model Parameters

To evaluate numerically sensitivities with respect to a model parameter λ, we make use of the Radon-Nykodim derivatives of the base process giving rise to the long step kernel $U = U(\lambda)$ with respect to another model specification with a slightly augmented value for the parameter λ and corresponding to the long-step kernel $U^+ = U(\lambda + \delta\lambda)$. Let

$$P(\bar{y},\lambda) = E_0^{U(\lambda)}\left[\Phi^D(\Gamma,T)\right] \tag{86}$$

be the pricing function with arguments the spot state variable $\bar{y}$ and the generic parameter λ, all other parameters being omitted. The first λ sensitivity is evaluated as follows:

$$\begin{aligned} \frac{P(\bar{y},\lambda+\delta\lambda)-P(\bar{y},\lambda)}{\delta\lambda} &= \frac{E_0^{U^+}\left[\Phi^D(\Gamma,T)|\bar{y}\right] - E_0^U\left[\Phi^D(\Gamma,T)|\bar{y}\right]}{\delta\lambda} \\ &= \frac{E_0^U\left[\Phi^D(\Gamma,T)W^+(\Gamma|\bar{y})|\bar{y}\right] - E_0^U\left[\Phi^D(\Gamma,T)|\bar{y}\right]}{\delta\lambda} \\ &= \frac{E_0^U\left[\Phi^D(\Gamma,T)\left(W^+(\Gamma|\bar{y})-1\right)|\bar{y}\right]}{\delta\lambda} \end{aligned} \tag{87}$$

where

$$W^+(\Gamma|\bar{y}) = \frac{U_0^+(\bar{y},\Gamma_0)}{U_0(\bar{y},\Gamma_0)}\frac{U_1^+(\Gamma_0,\Gamma_1)}{U_1(\Gamma_0,\Gamma_1)}\cdots\frac{U_{N_k-1}^+(\Gamma_{N_k-2},\Gamma_{N_k-1})}{U_{N_k-1}(\Gamma_{N_k-2},\Gamma_{N_k-1})} \tag{88}$$

9.0.4 Second Order Sensitivities with Respect to Model Parameters

The second order variation with respect to the generic model parameter λ is evaluated similarly to the first order sensitivity by considering a long-step kernel $U^+ = U(\lambda + \delta\lambda)$

with slightly augmented value for λ and a long-step kernel $U^- = (\lambda - \delta\lambda)$ with a slightly decreased value.

$$\begin{aligned}
& \frac{P(\bar{y}, \lambda + \delta\lambda) + P(\bar{y}, \lambda - \delta\lambda) - 2P(\bar{y}, \lambda)}{(\delta\lambda)^2} \\
= \ & \frac{1}{(\delta\lambda)^2} \left(E_0^{U^+} \left[\Phi^D(\Gamma, T) | \bar{y} \right] + E_0^{U^-} \left[\Phi^D(\Gamma, T) | \bar{y} \right] - 2E_0^U \left[\Phi^D(\Gamma, T) | \bar{y} \right] \right) \\
= \ & \frac{1}{(\delta\lambda)^2} E_0^U \Big[\Phi^D(\Gamma, T) \big(W^+ (\Gamma | \bar{y}) + W^- (\Gamma | \bar{y}) - 2 \big) | \bar{y} \Big] \qquad (89)
\end{aligned}$$

where $W^+(\Gamma|\bar{y})$ is given by (88) and

$$W^-(\Gamma|\bar{y}) = \frac{U_0^- (\bar{y}, \Gamma_0)}{U_0 (\bar{y}, \Gamma_0)} \frac{U_1^- (\Gamma_0, \Gamma_1)}{U_1 (\Gamma_0, \Gamma_1)} \cdots \frac{U_{N_k-1}^- (\Gamma_{N_k-2}, \Gamma_{N_k-1})}{U_{N_k-1} (\Gamma_{N_k-2}, \Gamma_{N_k-1})} \qquad (90)$$

9.0.5 Mixed Sensitivities

Mixed sensitivities with respect to state variables and model parameters, also known as cross-gammas are defined as follows:

$$\begin{aligned}
& \frac{1}{\nabla^+ X} \left(\frac{P(\bar{y}+1, \lambda + \delta\lambda) - P(\bar{y}+1, \lambda)}{\delta\lambda} - \frac{P(\bar{y}, \lambda + \delta\lambda) - P(\bar{y}, \lambda)}{\delta\lambda} \right) \\
= \ & \frac{1}{\delta\lambda \nabla^+ X} \Big(E_0^U \left[\Phi^D(\Gamma, T) \left(W^+ (\Gamma | \bar{y}+1) - 1 \right) | \bar{y}+1 \right] - E_0^U \left[\Phi^D(\Gamma, T) \left(W^+ (\Gamma | \bar{y}) - 1 \right) | \bar{y} \right] \Big) \\
= \ & \frac{1}{\delta\lambda \nabla^+ X} \Big(E_0^U \left[\Phi^D(\Gamma, T) \left(W^+ (\Gamma | \bar{y}+1) - 1 \right) Q^+(\Gamma) | \bar{y} \right] - E_0^U \left[\Phi^D(\Gamma, T) \left(W^+ (\Gamma | \bar{y}) - 1 \right) | \bar{y} \right] \Big) \\
= \ & \frac{1}{\delta\lambda \nabla^+ X} E_0^U \left[\Phi^D(\Gamma, T) \left(W^+ (\Gamma | \bar{y}+1) Q^+(\Gamma) - Q^+(\Gamma) - W^+(\Gamma|\bar{y}) + 1 \right) | \bar{y} \right] \qquad (91)
\end{aligned}$$

where $Q^+(\Gamma)$ is given by (83) and $W^+(\Gamma|\bar{y})$ is given by (88). Notice that

$$\begin{aligned}
& W^+(\Gamma|\bar{y}+1)Q^+(\Gamma) \\
& \quad = \frac{U_0^+ (\bar{y}+1, \Gamma_0)}{U_0 (\bar{y}+1, \Gamma_0)} \frac{U_1^+ (\Gamma_0, \Gamma_1)}{U_1 (\Gamma_0, \Gamma_1)} \cdots \frac{U_{N_k-1}^+ (\Gamma_{N_k-2}, \Gamma_{N_k-1})}{U_{N_k-1} (\Gamma_{N_k-2}, \Gamma_{N_k-1})} \frac{U_0 (\bar{y}+1, \Gamma_0)}{U_0 (\bar{y}, \Gamma_0)} \\
& \quad = \frac{U_0^+ (\bar{y}+1, \Gamma_0)}{U_0 (\bar{y}, \Gamma_0)} \frac{U_1^+ (\Gamma_0, \Gamma_1)}{U_1 (\Gamma_0, \Gamma_1)} \cdots \frac{U_{N_k-1}^+ (\Gamma_{N_k-2}, \Gamma_{N_k-1})}{U_{N_k-1} (\Gamma_{N_k-2}, \Gamma_{N_k-1})} \qquad (92)
\end{aligned}$$

9.0.6 Rho

Rho is the derivative of the option value with respect to the risk free rate and measures the sensitivity to the interest rate level. In case interest rates are deterministic, i.e. independent on the state variables of the underlying process, we use the money-market account as a numeraire asset and a change in interest rates is reflected in a change in the value of the numeraire asset. A constant shift in rates would lead from the money-market account in (69) to the perturbed process

$$B'(t) = e^{(r_i + \delta r)(t - T_i)} \prod_{i: T_i < t} e^{(r_i + \delta r)(T_i - T_{i-1})}. \qquad (93)$$

Hence, the different discounting reflects as follows on the value of the discounted payoff functional:

$$\Phi^{D'}(\Gamma, T) = R(T)\Phi^{D}(\Gamma, T). \tag{94}$$

where

$$R(t) = e^{-\delta r(t-T_i)} \prod_{i:T_i<t} e^{-\delta r(T_i - T_{i-1})}. \tag{95}$$

Hence

$$\begin{aligned} Rho &= \frac{P(\bar{y}, r+\delta r) - P(\bar{y}, r)}{\delta r} \\ &= \frac{E_0^{U^+}\left[\Phi^{D'}(\Gamma, T)|\bar{y}\right] - E_0^{U}\left[\Phi^{D}(\Gamma, T)|\bar{y}\right]}{\delta r} \\ &= \frac{E_0^{U}\left[\Phi^{D'}(\Gamma, T)W^+(\Gamma)|\bar{y}\right] - E_0^{U}\left[\Phi^{D}(\Gamma, T)|\bar{y}\right]}{\delta r} \\ &= \frac{E_0^{U}\left[\Phi^{D}(\Gamma, T)\left(R(T)W^+(\Gamma) - 1\right)|\bar{y}\right]}{\delta r} \end{aligned} \tag{96}$$

10 Model Specifications

In this section we review three model specifications based on Brownian motion, a local volatility process and a stochastic volatility process.

Discrete diffusion processes are defined on a lattice of $n_x > 1$ sites given by the set of integers $\Lambda = \{0, 1, 2, \ldots, n_x - 1\}$. The Markov generator has the form

$$\mathcal{L}(x_1, x_2; t) = \mu(x_1, t)\nabla^{\pm}(x_1, x_2) + \frac{\sigma^2(x_1, t)}{2}\Delta(x_1, x_2). \tag{97}$$

Here $\nabla^{\pm}(x_1, x_2)$ is the kernel for the discrete gradient and Laplace operators given by

$$\nabla^{\pm}(x_1, x_2) = \delta_{x_1 \pm 1, x_2} - \delta_{x_1, x_2}. \tag{98}$$

The plus sign is chosen in case $\mu > 0$ while we select the negative sign in case $\mu < 0$, so that either way $\mu\nabla^{\pm}$ is a correctly defined Markovian. The operator $\Delta(x_1, x_2)$ is given by

$$\Delta(x_1, x_2) = \delta_{x_1+1, x_2} - 2\delta_{x_1, x_2} + \delta_{x_1-1, x_2} \tag{99}$$

Financial risk factors are associated to the lattice process by means of a non-linear function $S(x, t)$. The parameter $\mu(x, t)$ is called *drift* and the parameter $\sigma(x, t)$ is called *volatility.* For reasons of numerical efficiency, it is important to define model parameters and the risk factor function $\phi(x, t)$ in such a way that they are piecewise constant. Periods of time over which parameters are constant are called *epochs.* Over each epoch, it is convenient to evaluate model parameters indirectly by assigning the first two moments of the risk factor short time increments, i.e.

$$\begin{aligned} \frac{m_1(x_1, t)}{\delta t} &\equiv \sum_{x_2} u_{\delta t}(x_1, x_2)(\phi(x_2, t) - \phi(x_1, t)) \\ &= \sum_{x_2} \mathcal{L}(x_1, x_2)(\phi(x_2, t) - \phi(x_1, t)) \end{aligned} \tag{100}$$

and

$$\begin{aligned}\frac{m_2(x_1,t)}{\delta t} &\equiv \sum_{x_2} u_{\delta t}(x_1,x_2)(\phi(x_2,t)-\phi(x_1,t))^2 \\ &= \sum_{x_2} \mathcal{L}(x_1,x_2)(\phi(x_2,t)-\phi(x_1,t))^2. \end{aligned} \tag{101}$$

Possible examples of diffusion processes of practical use in financial modeling include

- Brownian motion for which moments are not state dependent, i.e.

$$m_1(x,t)=\mu(t), \qquad m_2(x,t)=\sigma(t)^2; \tag{102}$$

- geometric Brownian motion with

$$m_1(x,t)=\mu(t)\phi(x,t), \qquad m_2(x,t)=\sigma(t)^2\phi(x,t)^2, \tag{103}$$

- mean reverting diffusions with

$$m_1(x,t)=\kappa(t)(\theta(t)-\phi(x,t)), \qquad m_2(x,t)=\sigma(t)^2\phi(x,t)^2, \tag{104}$$

- local volatility processes with

$$m_1(x,t)=\mu(t)\phi(x,t), \qquad m_2(x,t)=\sigma(t)^2\left(\frac{\phi(x,t)}{\phi(x_0,t)}\right)^{2\beta(S(x))-2}\phi(x,t)^2. \tag{105}$$

A two factor or regime switching model is characterized by a state variable given by a pair $y=(x,r)$ where $x=0,1,...n_x-1$ and $r=0,1,..n_r-1$. The variable y itself can be written in the form $y=x+n_x*r$ and takes values $y=0,...d-1$ where $d=n_x\cdot n_r$. Let us introduce the functions

$$r(y)=\left[\frac{y}{n_x}\right] \tag{106}$$

and

$$x(y)=y-n_x r(y). \tag{107}$$

It is useful to cast the Markovian in the form

$$\mathcal{L}(y_1,y_2)\equiv\mathcal{L}_x(x(y_1),x(y_2);r(y_1))\delta_{r(y_1),r(y_2)}+\mathcal{A}(y_1,y_2). \tag{108}$$

Here $\mathcal{L}_x(x(y_1),x(y_2);r(y_1))$ is a restricted Markovian in the state variable x only specific to the regime as a function of the regime state variable $r(y_1)$. The matrix $\mathcal{A}(y_1,y_2)$ is a remainder term. Notice that the first term in this equation describes only the transitions between x state variables, not transition between regime state variables. The general intuition behind this separation of terms is that the x variables make transitions with high frequency on a finely discretized lattice while the regime variables change less often.

To correlate the x and the r variables one could in principle think of considering cross moments of the form

$$\frac{m_{rx}(x_1,r_1t)}{\delta t}\equiv\sum_{x_2,r_2}\mathcal{L}(x_1+n_x\cdot r_1,x_2+n_x\cdot r_2)(\phi(x_2,t)-\phi(x_1,t))(\psi(r_2,t)-\psi(r_1,t)). \tag{109}$$

However, due to the typically major gap in time scales between fast modes driven by the variable x and slow modes driven by the variable r, it turns out that to enforce such a cross-moment condition one would require a fine discretization of both the variables x and r. To allow for a coarser discretization of the regime variable r, we instead proceed otherwise to model correlations and derive them out of a definition of a jump component in the process. Namely, conditioned to a transition in regime variable x occurring, we allow the x variable to jump either up or down according to an exponential distribution.

To be more specific, the construction of a regime switching generator proceeds as follows. Firstly we consider the regime specific dynamics given by a family of generators of the form

$$\mathcal{L}_x(x_1, x_2; r_1) = \mu_x(x_1, r_1, t)\nabla^{\pm}(x_1, x_2) + \frac{\sigma_x^2(x_1, r_1, t)}{2}\Delta(x_1, x_2) \tag{110}$$

These generators can be identified by means of first and second moments as described above for the case of one factor diffusion processes.

Secondly, one considers a dynamics for the regime variable given by a similar Markovian of the form

$$\mathcal{L}_r(r_1, r_2; x_1) = \mu_r(x_1, r_1, t)\nabla^{\pm}(x_1, x_2) + \frac{\sigma_r^2(x_1, r_1, t)}{2}\Delta(x_1, x_2). \tag{111}$$

For each transition $r_1 \to r_2$ with $r_1 \neq r_2$, we introduces jumps in the x coordinate in such a way to preserve the transition probability rate given by this Markovian $\mathcal{L}_r(r_1, r_2; x_1)$. More precisely, we define the residual matrix $\mathcal{A}(y_1, y_2)$ so that

$$\sum_{x_2} \mathcal{A}(x_1 + n_x \cdot r_1, x_2 + n_x \cdot r_2) = \mathcal{L}_r(r_1, r_2; x_1) \tag{112}$$

for all $r_1 \neq r_2$. Assuming that jumps are distributed exponentially, we set

$$\mathcal{A}(x_1 + n_x \cdot r_1, x_2 + n_x \cdot r_2) = c(x_1, r_1, r_2)e^{-(\phi(x_2)-\phi(x_1))/a_{\pm}} \tag{113}$$

where $a_{\pm}$ are two constants. In this equation, we select a_+ in case $x_2 >= x_1$ and select a_- in case $x < x_2$. The constant $c(x_1, r_1, r_2)$ is positive and is evaluated in such a way to satisfy equation (112).

A final remark concerns the drift. As a consequence of the two-step construction above, the drift of the resulting process deviates from the input drift used for the initial x process. As a consequence, it is often not even worthwhile to specify this drift to any value different from zero in the first stage. After introducing jumps, the drift can always be restored to the desired value with a simple construction. Namely, we evaluate the first moment

$$\begin{aligned} \frac{m_1(y_1, t)}{\delta t} &\equiv \sum_{y_2} u_{\delta t}(y_1, y_2)(\phi(x(y_2), t) - \phi(x(y_1), t)) \\ &= \sum_{x_2} \mathcal{L}'(y_1, y_2)(\phi(x(y_2), t) - \phi(x(y_1), t)) \end{aligned} \tag{114}$$

where $\mathcal{L}'(y_1, y_2)$ is the Markovian obtained after the first two steps and we then set

$$\mathcal{L}'(y_1, y_2) = \mathcal{L}'(y_1, y_2) + \delta\mu(y_1)\nabla^{\pm}, \tag{115}$$

where $\delta\mu(y_1)$ is chosen so that the first moment for the resulting Markovian is at the desired value and the sign in the gradient operator $\nabla^{\pm}$ is chosen so that $\delta\mu(y_1)\nabla^{\pm}$ is itself a correctly defined Markovian.

11 Numerical Experiments

In this section, we discuss some numerical experiments illustrating our algorithm for finding price sensitivities with Monte Carlo simulations by means of the likelihood ratio method. We consider a local volatility model of volatility function $\sigma_0 S^\beta$ with interest rate $r = 0$, volatility $\sigma_0 = 25\%$. The initial stock price is $S_0 = 100$ and parameter $\beta = 0.25$. We price an option of maturity $T = 10$ years and a butterfly type payoff of the form

$$(S_T - 95)^+ - 2(S_T - 100)^+ + (S_T - 105)^+ \tag{116}$$

In addition, there is a knock-out clause whenever the stock hits barriers at $L = \$75$ and $U = \$125$ at epoch dates occurring with yearly frequency starting from the value date. We evaluate numerically the following sensitivities:

- Delta and Gamma sensitivities as described in Section 9;
- The Vega and the Vomma, defined as the first and second sensitivities with respect to a small change in $\sigma_0 = 25\%$ to $\sigma_0 + 1\%$;
- the Vanna, defined as the mixed sensitivity with respect to a small change in spot underlying price and a small change in σ_0;
- the Rho, defined as the sensitivity with respect to a small change in the interest rate from 0 to 0.001.
- the BetaPV01, defined as the price sensitivity with respect to a small change in the β parameter from $\beta = 25\%$ to $\beta + 1\%$.

In the graphs below, on the X axis we plot the number of batches for the simulation and in each batch, we generate one million scenarios. We compare sensitivities obtained by simply rerunning the simulation with the same initial seeds and bumped parameters with the likelihood ratio method reviewed in this paper. The main advantage of the likelihood ratio method is speed, as it requires recomputing the kernels but not generating new scenarios or evaluating payoffs anew. This is a particularly important advantage in cases where one is evaluating large portfolios and a computational bottleneck is given by the payoff valuation step. A secondary advantage is an effect of variance reduction that we notice empirically.

12 Conclusion

We review a general Monte Carlo simulation methodology that covers a large class of not necessarily solvable processes whose dynamics is expressed in the form of a Markov chain. The theory is illustrated ab initio and algorithms are reviewed in detail. The interested reader is invited to download the open source library OPLib (Albanese 2009) for coded examples and benchmarks. We conclude that Monte Carlo algorithms are best orchestrated on current hardware platforms by using GPU coprocessors for kernel valuations and CPUs for scenario generation.

References

Albanese, C. (2007*a*). Kernel Convergence Estimates for Diffusions with Continuous Coefficients.

Albanese, C. (2007*b*). Stochastic Integrals and Abelian Processes.

Albanese, C. (2009). Oplib 1.0, a monte Carlo Pricing Library Based on Operator Methods.

Chen, N. and P. Glasserman (2007). Malliavin Greeks without Malliavin Calculus. **117**, 1689–1723.

de Finetti, B. (1931). Sul Significato Soggettivo della Probabilita'.. **17**, 298–329.

Farkas, J. (1902). Ueber die Theorie der Einfachen Ungleichungen. **124**, 1–27.

Knuth, D.E. (1969). Semi-numerical Algorithms: The Art of Computer Programming.

Kolmogorov, A. (1933). *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer.

Matsumoto, M. (1998). Mersenne Twister: a 623-dimensionally equidistributed uniform psudo-random number generator. **8**, 3.

Matsumoto, M. (2007-2009). Simd-oriented fast mersenne twister.

Plato, J. Von (1994). *Creating Modern Probability*. Dover Publications, New York.

R. Courant, K. Friedrichs and H. Lewy (1928). ber die Partiellen Differenzengleichungen der Mathematischen Physik. **100**, 32–74.

S. P. Jones, J. M. Eber and J. Seward (2000). Composing Contracts: An Adventure in Financial Engineering.

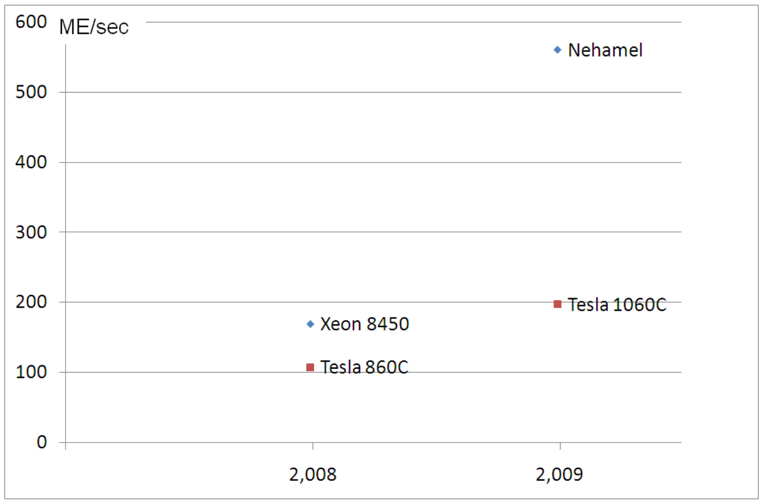

Figure 1: Comparative performance at Monte Carlo scenario generation.

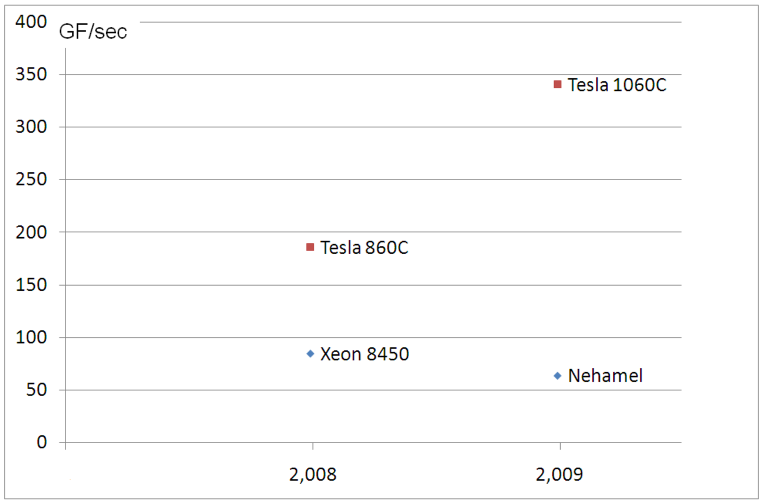

Figure 2: Comparative performance at third level BLAS.

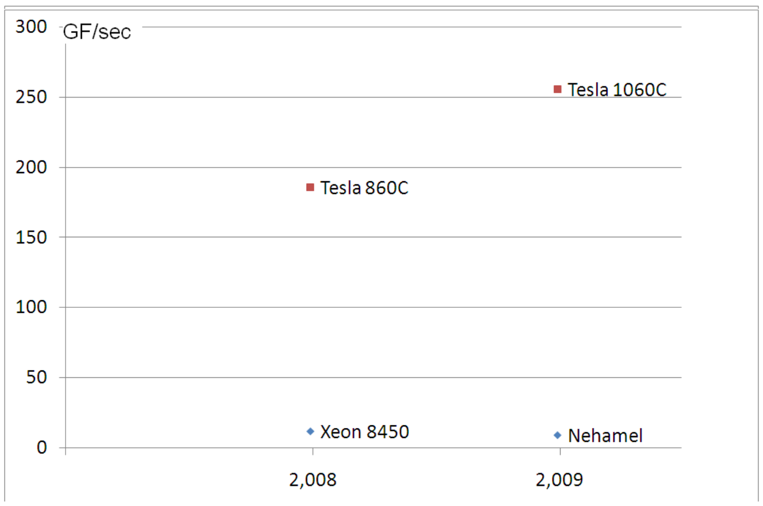

Figure 3: Comparative performance at multiple kernel evaluation. This is handled concurrently on GPUs and serially on CPUs.

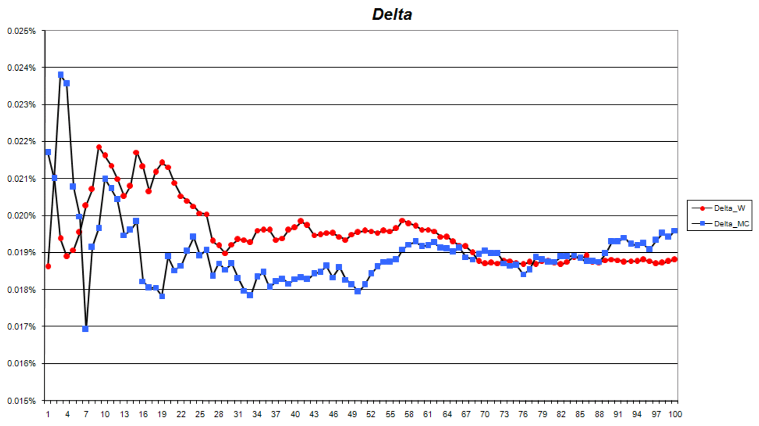

Figure 4: Convergence of the Delta for a barrier butterfly spread as a function of the number of simulation batches.

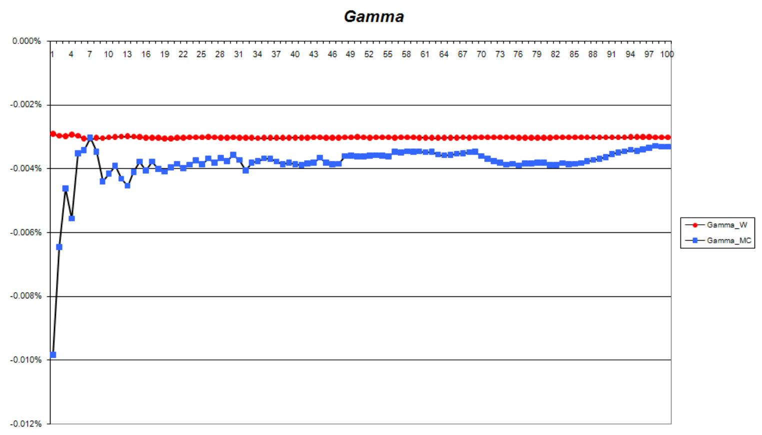

Figure 5: Convergence of the Gamma for a barrier butterfly spread as a function of the number of simulation batches.

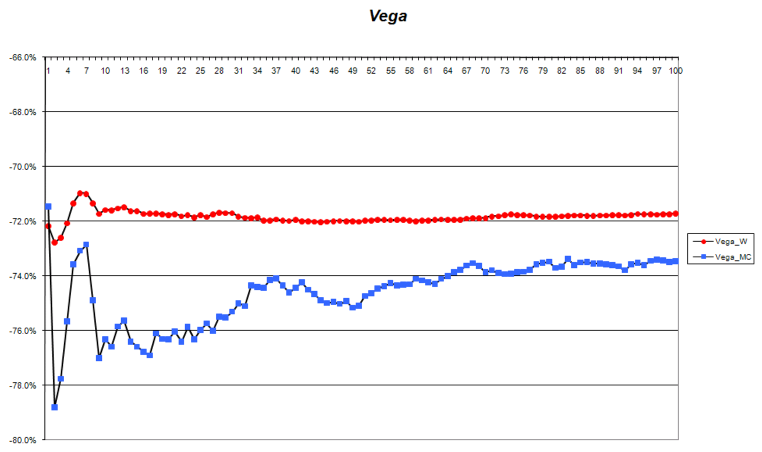

Figure 6: Convergence of the Vega for a barrier butterfly spread as a function of the number of simulation batches.

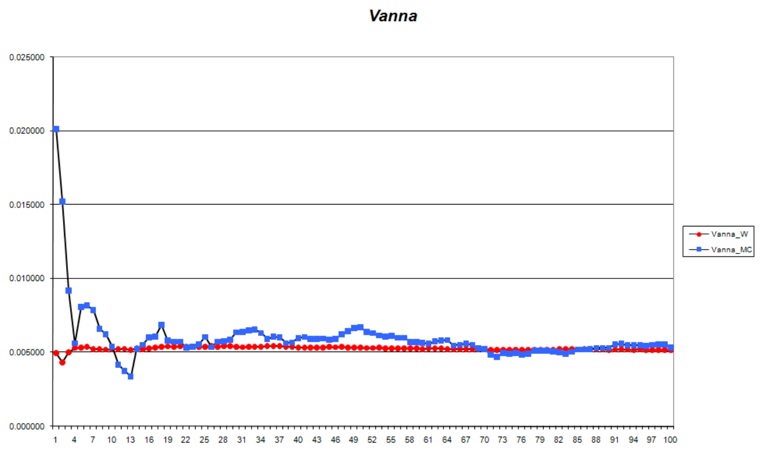

Figure 7: Convergence of the Vanna for a barrier butterfly spread as a function of the number of simulation batches.

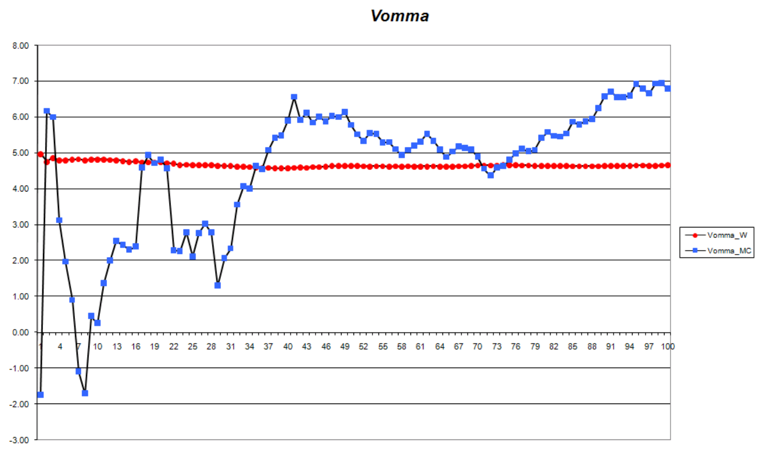

Figure 8: Convergence of the Vomma for a barrier butterfly spread as a function of the number of simulation batches.

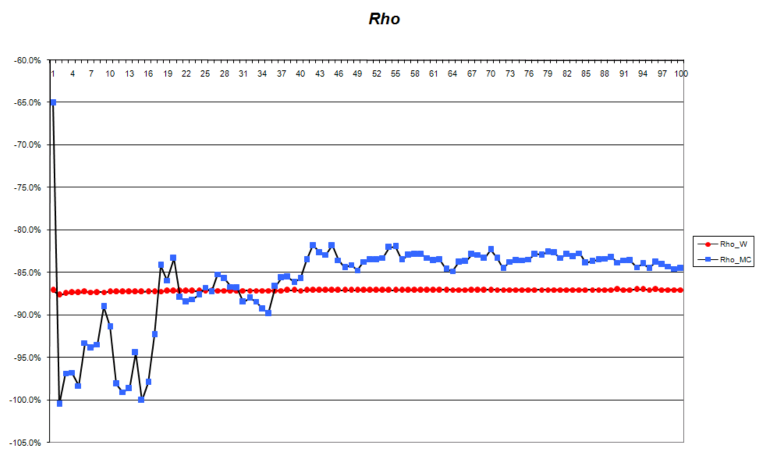

Figure 9: Convergence of the Rho for a barrier butterfly spread as a function of the number of simulation batches.

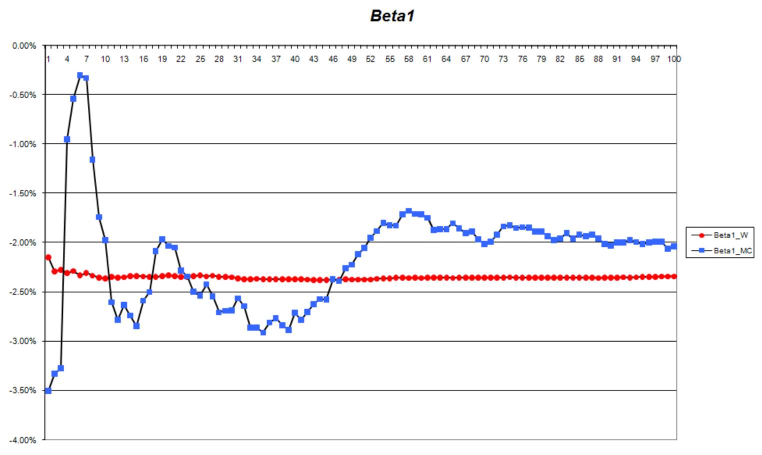

Figure 10: Convergence of the BetaPV01 for a barrier butterfly spread as a function of the number of simulation batches.