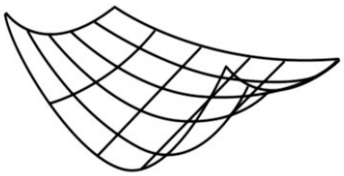

QUANTITATIVE FINANCE RESEARCH

UTS
UNIVERSITY OF TECHNOLOGY SYDNEY

QUANTITATIVE FINANCE RESEARCH

Research Paper 382 May 2017

Fast Quantization of Stochastic Volatility Models

Ralph Rudd, Thomas A. McWalter, Jörg Kienitz and Eckhard Platen

ISSN 1441-8010 www.qfrc.uts.edu.au

Fast Quantization of Stochastic Volatility Models

Ralph Rudd[1], Thomas A. McWalter*[1,2], Jörg Kienitz[1,3] and Eckhard Platen[1,4]

[1]Department of Actuarial Science and the African Collaboration for Quantitative Finance and Risk Research, University of Cape Town
[2]Department of Finance and Investment Management, University of Johannesburg
[3]Fachbereich Mathematik und Naturwissenschaften, Bergische Universität Wuppertal
[4]Finance Discipline Group and School of Mathematical and Physical Sciences, University of Technology Sydney

April 21, 2017

Abstract

Recursive Marginal Quantization (RMQ) allows fast approximation of solutions to stochastic differential equations in one-dimension. When applied to two factor models, RMQ is inefficient due to the fact that the optimization problem is usually performed using stochastic methods, e.g., Lloyd's algorithm or Competitive Learning Vector Quantization. In this paper, a new algorithm is proposed that allows RMQ to be applied to two-factor stochastic volatility models, which retains the efficiency of gradient-descent techniques. By margining over potential realizations of the volatility process, a significant decrease in computational effort is achieved when compared to current quantization methods. Additionally, techniques for modelling the correct zero-boundary behaviour are used to allow the new algorithm to be applied to cases where the previous methods would fail. The proposed technique is illustrated for European options on the Heston and Stein-Stein models, while a more thorough application is considered in the case of the popular SABR model, where various exotic options are also priced.

1 Introduction

Quantization is a lossy compression technique that has been applied to many challenging problems in mathematical finance, including pricing options with path dependence and early exercise [Pagès and Wilbertz, 2009; Sagna, 2011; Bormetti et al., 2016], stochastic control problems [Pagès et al., 2004] and non-linear filtering [Pagès and Pham, 2005].

Pagès and Sagna [2015] introduced a technique known as Recursive Marginal Quantization (RMQ), which approximates the marginal distribution of a stochastic differential equation by recursively quantizing the Euler approximation of the process. This was extended to higher-order schemes by McWalter et al. [2017]. RMQ can be applied to any one-dimensional SDE, even when the transition density is unknown, and has been used to efficiently calibrate a local volatility model by Callegaro et al. [2014, 2015a].

*Correspondence: tom@analytical.co.za

Applying the standard RMQ technique to a two-factor SDE generally requires the use of stochastic numerical methods, such as the randomized Lloyd's method or stochastic gradient descent methods such as Competitive Learning Vector Quantization (see Pagès [2014] for an overview of these methods). The computational cost of these techniques is prohibitive.

To overcome this numerical inefficiency, Callegaro et al. [2015b] used conditioning to derive a modified RMQ algorithm that can be applied to stochastic volatility models while retaining the use of the underlying Newton-Raphson technique. This was achieved by performing a one-dimensional RMQ on the volatility process and then conditioning on the realizations of the resultant quantized process. We derive a new RMQ algorithm for the stochastic volatility setting by showing that the correlation between the two processes may be neglected when minimizing the distortion. We call this innovation the Joint Recursive Marginal Quantization (JRMQ) algorithm. It results in an increase in accuracy and a large increase in efficiency. Furthermore, it allows for the modelling of the correct zero-boundary behaviour of the underlying processes. We now provide an overview of the paper.

In Section 2 an overview of the RMQ algorithm in the one-dimensional case is provided. Section 3 derives the JRMQ algorithm for the stochastic volatility setting with the main result of the paper contained in Proposition 3.1. Section 4 discusses how to efficiently compute the joint probabilities required by the new algorithm. In Section 5, a concise matrix formulation is provided to ease implementation. Section 6 prices European options under the Stein-Stein, Heston and SABR stochastic volatility models. In Section 7, a single grid generated by the JRMQ algorithm for the SABR model is used to price Bermudan and barrier options, and volatility corridor swaps. Section 8 concludes.

2 Quantization of Single-factor Models

Let X be a continuous random variable, taking values in $\mathbb{R}$, and defined on the probability space $(\Omega, \mathscr{F}, \mathbb{P})$. We seek an approximation of this random variable, denoted $\widehat{X}$, taking values in a set of finite cardinality, Γ^x, with the minimum expected squared Euclidean difference from the original. Constructing this approximation is known as vector quantization, with $\widehat{X}$ called the *quantized* version of X and the set $\Gamma^x = \{x^1, \ldots, x^N\}$ known as the *quantizer* with cardinality N. The elements of Γ^x are called *codewords*.

The primary utility of quantization is the efficient approximation of expectations of functionals of the random variable X using

$$\mathbb{E}\left[H(X)\right] = \int_{\mathbb{R}} H(x)\, d\mathbb{P}(X \leq x) \approx \sum_{i=1}^{N} H(x^i)\mathbb{P}\big(\widehat{X} = x^i\big),$$

where $\widehat{X}$ denotes the quantized version of X. We now briefly describe the mathematics of vector quantization. Consider the nearest-neighbour projection operator, $\pi_{\Gamma^x} : \mathbb{R} \mapsto \Gamma^x$, given by

$$\pi_{\Gamma^x}(X) := \big\{x^i \in \Gamma^x \,\big|\, \|X - x^i\| \leq \|X - x^j\| \text{ for all } j = 1, \ldots, N; \text{ where equality holds only for } i < j\big\}.$$

The quantized version of X is defined in terms of this projection operator as $\widehat{X} := \pi_{\Gamma^x}(X)$. The *region* $R^i(\Gamma^x)$, for $1 \leq i \leq N$, is defined as

$$R^i(\Gamma^x) := \big\{z \in \mathbb{R} \,\big|\, \pi_{\Gamma^x}(z) = x^i\big\},$$

and is the subset of $\mathbb{R}$ mapped to codeword x^i through the projection operator.

The expected squared Euclidean error, known as the *distortion*, is given by

$$\begin{aligned} D(\Gamma^x) &= \mathbb{E}\left[\|X - \widehat{X}\|^2\right] \\ &= \int_{\mathbb{R}} \|x - \pi_{\Gamma^x}(x)\|^2 \, d\mathbb{P}(X \leq x) \\ &= \sum_{i=1}^{N} \int_{R_i(\Gamma^x)} \|x - x^i\|^2 \, d\mathbb{P}(X \leq x), \end{aligned}$$

and is the function that must be minimized in order to obtain the optimal quantizer. We retain the symbol x to refer to the continuous domain of the distribution of the random variable X, whereas x^i refers to the discrete codewords of the resulting quantizer, Γ^x, for $1 \leq i \leq N$.

When the gradient and Hessian of the distortion can be computed in closed-form, a simple Newton-Raphson algorithm may be used to minimise the distortion,

$${}^{(n+1)}\mathbf{\Gamma}^x = {}^{(n)}\mathbf{\Gamma}^x - \left[\nabla^2 D\left({}^{(n)}\mathbf{\Gamma}^x\right)\right]^{-1} \nabla D\left({}^{(n)}\mathbf{\Gamma}^x\right).$$

Here, $0 \leq n < n_{\max}$ is the iteration index of the algorithm and $[{}^{(n)}\mathbf{\Gamma}^x]_i = x^i$, for $1 \leq i \leq N$, is a column vector containing the codewords. The gradient vector and Hessian matrix of the distortion are $\nabla D\left({}^{(n)}\mathbf{\Gamma}^x\right)$ and $\nabla^2 D\left({}^{(n)}\mathbf{\Gamma}^x\right)$, respectively. Note that the distortion function is applied element-wise to the column vector ${}^{(n)}\mathbf{\Gamma}^x$, and ${}^{(0)}\mathbf{\Gamma}^x$ is an initial guess for the quantizer. McWalter et al. [2017] provide explicit expressions for the gradient vector and Hessian matrix in the one-dimensional case, and an efficient matrix formulation for implementation.

To extend the applicability of vector quantization for use with SDEs, Pagès and Sagna [2015] proposed recursive marginal quantization of the Euler scheme for an SDE. In order to fix the notation used in the remainder of the paper we briefly specify this problem.

Consider the one-dimensional continuous-time stochastic differential equation

$$dX_t = a^x(X_t)\, dt + b^x(X_t)\, dW_t^x, \qquad X_0 = x_0,$$

defined on $(\Omega, \mathscr{F}, (\mathscr{F}_t)_{t\in[0,T]}, \mathbb{P})$, a filtered probability space satisfying the usual conditions. The discrete-time Euler approximation $\widetilde{X}$ of X, on an evenly spaced time grid, is given by

$$\begin{aligned} \widetilde{X}_{k+1} &= \widetilde{X}_k + a^x(\widetilde{X}_k)\Delta t + b^x(\widetilde{X}_k)\sqrt{\Delta t} Z_{k+1}^x \\ &=: \mathcal{U}^x(\widetilde{X}_k, Z_{k+1}^x), \end{aligned} \tag{1}$$

for $0 \leq k < K$, where $\Delta t = T/K$, and $Z_{k+1}^x \sim \mathscr{N}(0,1)$ are independent standard Gaussian random variables.

The optimal quantizer for the continuous-time process X, at each fixed time $t_{k+1} = (k+1)\Delta t$, should be computed using the distortion

$$\mathbb{E}\left[\|X_{t_{k+1}} - \pi_{\Gamma^x}(X_{t_{k+1}})\|^2\right].$$

This is, however, not possible in the general case, since the distribution of $X_{t_{k+1}}$ is unknown. We instead consider the distortion computed in terms of the Euler approximation $\widetilde{X}_{k+1}$.

Let Γ_k^x be the quantizer of $\widetilde{X}$, at time-step k with $0 \leq k \leq K$. To remain consistent with the specification of the problem above, the quantizer at the initial time, t_0, is given by $\Gamma_0^x = \{x_0\}$. We fix the cardinality of the quantizers at all other time steps to be N^x — this may, however, be relaxed (see, for example, the discussion on optimal dispatching in Pagès and Sagna [2015]). Since the Euler update is normally distributed, the quantizer at the first time step is just the vector quantization of a normal distribution. The distortion of the quantizer for each successive step is then given by

$$\begin{aligned}
\widetilde{D}(\Gamma_{k+1}^x) &= \mathbb{E}\left[\left\|\widetilde{X}_{k+1} - \pi_{\Gamma^x}(\widetilde{X}_{k+1})\right\|^2\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\left\|\widetilde{X}_{k+1} - \widehat{X}_{k+1}\right\|^2 \,\middle|\, \widetilde{X}_k\right]\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\left\|\mathcal{U}^x(\widetilde{X}_k, Z_{k+1}^x) - \widehat{X}_{k+1}\right\|^2 \,\middle|\, \widetilde{X}_k\right]\right] \\
&= \int_{\mathbb{R}} \mathbb{E}\left[\left\|\mathcal{U}^x(x, Z_{k+1}^x) - \widehat{X}_{k+1}\right\|^2\right] d\mathbb{P}(\widetilde{X}_k \leq x).
\end{aligned}$$

To proceed, we approximate the above distortion using the distribution of $\widehat{X}_k$ rather than $\widetilde{X}_k$, in which case

$$\widetilde{D}(\Gamma_{k+1}^x) \approx D(\Gamma_{k+1}^x) := \sum_{i=1}^{N^x} \mathbb{E}\left[\left\|\mathcal{U}^x(x_k^i, Z_{k+1}) - \widehat{X}_{k+1}\right\|^2\right] \mathbb{P}(\widehat{X}_k = x_k^i),$$

where the approximate distortion is defined without any accents.

This is the one dimensional vector quantization problem, where the distribution being quantized is a marginal distribution consisting of the probability-weighted sum of Euler updates, each having originated from the codewords in the quantizer at the previous time step. Recursively applying this procedure to the updates of the Euler process is known as recursive marginal quantization (RMQ). Since the vector quantization problem specified in this manner is one-dimensional, the efficient Newton-Raphson procedure can be used to minimize the resulting distortion, which yields the quantizer at each time-step. McWalter et al. [2017] derive explicit and efficient expressions for the gradient vector and Hessian matrix required for the Newton-Raphson procedure and show that recursive marginal quantization of higher-order schemes is possible — specifically the Milstein and simplified weak-order 2.0 schemes. In the present work, we only consider the Euler scheme.

Note that the Euler update (1) can be written in an affine form as

$$\mathcal{U}^x(\widetilde{X}_k, Z_{k+1}^x) = m_k^i Z_{k+1}^x + c_k^i, \tag{2}$$

with

$$m_k^i := b^x(x_k^i)\sqrt{\Delta t} \qquad \text{and} \qquad c_k^i := x_k^i + a^x(x_k^i)\Delta t. \tag{3}$$

Thus, for a given quantizer, Γ_{k+1}^x, the standardized region boundaries associated with each codeword are given by

$$r_{k+1}^{i,j\pm} = \frac{\frac{1}{2}(x_{k+1}^{j\pm1} + x_{k+1}^j) - c_k^i}{m_k^i}, \tag{4}$$

for $1 \leq i, j \leq N^x$. This refers to the upper and lower region boundary of codeword x_{k+1}^j when viewed from codeword x_k^i. Equations (2) to (4) are central to the standard RMQ algorithm, see McWalter et al. [2017], and are presented here for use later in the paper.

3 Quantization of Stochastic Volatility Models

In this section, we consider the recursive marginal quantization of a generic stochastic volatility model described by the coupled SDEs

$$dX_t = a^x(X_t)\,dt + b^x(X_t)\,dW_t^x, \qquad X_0 = x_0, \tag{5}$$

$$dY_t = a^y(Y_t)\,dt + b^y(X_t, Y_t)\,(\rho\,dW_t^x + \sqrt{1-\rho^2}\,dW_t^\perp), \qquad Y_0 = y_0 \tag{6}$$

defined on $(\Omega, \mathscr{F}, (\mathscr{F}_t)_{t\in[0,T]}, \mathbb{P})$, where W_t^x and $W_t^\perp$ are independent standard Brownian motions. In this system, the Cholesky decomposition, specified in terms of the correlation parameter $\rho \in [-1, 1]$, is chosen explicitly in order to facilitate derivations. Here, X_t, referred to as the *independent* process, drives the specification of the stochastic volatility factor in the *dependent* process Y_t.

The Euler scheme for the above system is given by

$$\widetilde{X}_{k+1} = \mathcal{U}^x(\widetilde{X}_k, Z_{k+1}^x), \qquad \widetilde{X}_0 = x_0 \tag{7}$$

$$\widetilde{Y}_{k+1} = \widetilde{Y}_k + a^y(\widetilde{Y}_k) + b^y(\widetilde{X}_k, \widetilde{Y}_k)\sqrt{\Delta t}(\rho Z_{k+1}^x + \sqrt{1-\rho^2} Z_{k+1}^\perp), \qquad \widetilde{Y}_0 = y_0 \tag{8}$$

$$=: \mathcal{U}^y(\widetilde{X}_k, \widetilde{Y}_k, Z_{k+1}^x, Z_{k+1}^\perp), \tag{9}$$

for $0 \le k < K$, where $\mathcal{U}^x(\widetilde{X}_k, Z_{k+1}^x)$ is defined by (1) and $Z_{k+1}^x, Z_{k+1}^\perp \sim \mathscr{N}(0,1)$ are independent standard Gaussian random variables. The main result of this paper is to show that quantizing the Euler update $\widetilde{Y}_{k+1} = \mathcal{U}^y(\widetilde{X}_k, \widetilde{Y}_k, Z_{k+1}^x, Z_{k+1}^\perp)$ is equivalent to quantizing the update given by

$$\overline{\mathcal{U}}^y(\widetilde{X}_k, \widetilde{Y}_k, Z) = \widetilde{Y}_k + a^y(\widetilde{Y}_k)\Delta t + b^y(\widetilde{X}_k, \widetilde{Y}_k)\sqrt{\Delta t} Z, \tag{10}$$

where $Z \sim \mathscr{N}(0,1)$ is any standard Gaussian random variable. Having established this result, we proceed to quantize the system and derive a one-dimensional vector quantization algorithm based on a Newton-Raphson iteration.

Proposition 3.1. *Given the Euler scheme defined by (7) and (8), the distortion of the quantizer Γ_{k+1}^y may be expressed as*

$$\widetilde{D}(\Gamma_{k+1}^y) = \mathbb{E}\left[\left\|\overline{\mathcal{U}}^y(\widetilde{X}_k, \widetilde{Y}_k, Z) - \widehat{Y}_{k+1}\right\|^2\right],$$

where the margined update function is defined by (10) with $Z \sim \mathscr{N}(0,1)$.

Proof. The distortion of the quantizer Γ_{k+1}^y for $\widetilde{Y}_{k+1}$ is given in terms of the update (9) as

$$\begin{aligned}
\widetilde{D}(\Gamma_{k+1}^y) :=& \mathbb{E}\left[\left\|\widetilde{Y}_{k+1} - \widehat{Y}_{k+1}\right\|^2\right] \\
=& \mathbb{E}\left[\mathbb{E}\left[\left\|\widetilde{Y}_{k+1} - \widehat{Y}_{k+1}\right\|^2 \,\middle|\, \widetilde{X}_k, \widetilde{Y}_k\right]\right] \\
=& \int_{\mathbb{R}^2} \mathbb{E}\left[\left\|\mathcal{U}^y(x, y, Z_{k+1}^x, Z_{k+1}^\perp) - \widehat{Y}_{k+1}\right\|^2\right] d\mathbb{P}(\widetilde{X}_k \le x, \widetilde{Y}_k \le y) \\
=& \int_{\mathbb{R}^2} \mathbb{E}\left[f\big(\mathcal{U}^y(x, y, Z_{k+1}^x, Z_{k+1}^\perp)\big)\right] d\mathbb{P}(\widetilde{X}_k \le x, \widetilde{Y}_k \le y),
\end{aligned}$$

where $f(w) := \left(w - \pi_{\Gamma_{k+1}^y}(w)\right)^2$. The inner expectation may be written explicitly as

$$\mathbb{E}\left[f\big(\mathcal{U}^y(x, y, Z_{k+1}^x, Z_{k+1}^\perp)\big)\right] = \frac{1}{2\pi}\int_{\mathbb{R}^2} f\big(\mathcal{U}^y(x, y, u, v)\big) \exp\left(\frac{-u^2}{2}\right) \exp\left(\frac{-v^2}{2}\right) dv\, du.$$

Now, let

$$z = \rho u + \sqrt{1-\rho^2}v,$$

which means that

$$v = \frac{z-\rho u}{\sqrt{1-\rho^2}} \qquad \text{and} \qquad dv = \frac{1}{\sqrt{1-\rho^2}}\,dz.$$

Then,

$$\begin{aligned}
&\mathbb{E}\left[f\big(\mathcal{U}^y(x,y,Z^x_{k+1},Z^\perp_{k+1})\big)\right] \\
&\quad= \frac{1}{2\pi\sqrt{1-\rho^2}}\int_{\mathbb{R}^2} f\big(\overline{\mathcal{U}}^y(x,y,z)\big)\exp\left(\frac{-u^2}{2}\right)\exp\left(\frac{-(z-\rho u)^2}{2(1-\rho^2)}\right)dz\,du \\
&\quad= \frac{1}{2\pi\sqrt{1-\rho^2}}\int_{\mathbb{R}^2} f\big(\overline{\mathcal{U}}^y(x,y,z)\big)\exp\left(\frac{-z^2}{2}\right)\exp\left(\frac{-(u-\rho z)^2}{2(1-\rho^2)}\right)dz\,du \\
&\quad= \frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}} f\big(\overline{\mathcal{U}}^y(x,y,z)\big)\exp\left(\frac{-z^2}{2}\right)\underbrace{\frac{1}{\sqrt{2\pi(1-\rho^2)}}\int_{\mathbb{R}}\exp\left(\frac{-(u-\rho z)^2}{2(1-\rho^2)}\right)du}_{=1}\,dz \\
&\quad= \frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}} f\big(\overline{\mathcal{U}}^y(x,y,z)\big)\exp\left(\frac{-z^2}{2}\right)dz,
\end{aligned}$$

where we have used Fubini's theorem in the penultimate step. Thus, we obtain

$$\mathbb{E}\left[f\big(\mathcal{U}^y(x,y,Z^x_{k+1},Z^\perp_{k+1})\big)\right] = \mathbb{E}\left[f\big(\overline{\mathcal{U}}^y(x,y,Z)\big)\right].$$

Putting everything together, we have

$$\begin{aligned}
\widetilde{D}\big(\Gamma^y_{k+1}\big) &= \int_{\mathbb{R}^2}\mathbb{E}\left[f\big(\overline{\mathcal{U}}^y(x,y,Z)\big)\right]\,d\mathbb{P}(\widetilde{X}_k\le x,\widetilde{Y}_k\le y) \\
&= \int_{\mathbb{R}^2}\mathbb{E}\left[\left\|\overline{\mathcal{U}}^y(x,y,Z)-\widehat{Y}_{k+1}\right\|^2\right]\,d\mathbb{P}(\widetilde{X}_k\le x,\widetilde{Y}_k\le y) \\
&= \mathbb{E}\left[\left\|\overline{\mathcal{U}}^y(\widetilde{X}_k,\widetilde{Y}_k,Z)-\widehat{Y}_{k+1}\right\|^2\right],
\end{aligned}$$

as required. □

Remark 3.2. *The above proposition demonstrates that the quantization of $\widetilde{Y}_{k+1}$ depends only on its distribution, and, from the perspective of the distortion function, the correlation between $\widetilde{Y}_{k+1}$ and $\widetilde{X}_{k+1}$ is irrelevant. Another way of saying this is that*

$$f\big(\mathcal{U}^y(x,y,Z^x_{k+1},Z^\perp_{k+1})\big)\overset{d}{=}f\big(\overline{\mathcal{U}}^y(x,y,Z)\big),$$

where $Z\sim\mathcal{N}(0,1)$, and, since we only need to consider weighted sums of expectations of these values when computing the distortion, the correlation between Z^x_{k+1} and $Z^\perp_{k+1}$ need not be considered. As we shall see later, it is necessary to take correlation into account when computing the joint probabilities of $\widetilde{Y}_{k+1}$ and $\widetilde{X}_{k+1}$.

As we did in the previous section, we now quantize the expression for the distortion. The quantization of the Euler scheme for the independent process, $\widetilde{X}$, proceeds directly using the standard RMQ algorithm from Section 2, and can be performed for all time steps

without reference to $\widetilde{Y}$. Suppose, at time step k, the quantizer for the dependent process Γ_k^y has been computed along with the corresponding joint probabilities $\mathbb{P}(\widehat{X}_k = x_k^i, \widehat{Y}_k = y_k^u)$, for $1 \leq i \leq N^x$ and $1 \leq u \leq N^y$, then the distortion for the quantizer of $\widetilde{Y}_{k+1}$ may be approximated by

$$\begin{aligned}\widetilde{D}(\Gamma_{k+1}^y) &= \int_{\mathbb{R}^2} \mathbb{E}\left[\left\|\overline{\mathcal{U}}^y(x,y,Z) - \widehat{Y}_{k+1}\right\|^2\right] d\mathbb{P}(\widetilde{X}_k \leq x, \widetilde{Y}_k \leq y) \\ &\approx \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \mathbb{E}\left[(\overline{\mathcal{U}}^y(x_k^i, y_k^u, Z) - \widehat{Y}_{k+1})^2\right] \mathbb{P}(\widehat{X}_k = x_k^i, \widehat{Y}_k = y_k^u) \qquad (11) \\ &=: D(\Gamma_{k+1}^y).\end{aligned}$$

The main result from Pagès and Sagna [2015] shows that the approximation in (11) results in a convergent procedure. We again assume that the cardinality of Γ_k^y is fixed at N^y for all $0 < k \leq K$ and that $\Gamma_0^y = \{y_0\}$. As before, the quantizer Γ_1^y may be computed using standard vector quantization of the normal distribution.

For the remainder of this section we assume that, conditional on knowing the quantizers Γ_k^x and Γ_k^y, their associated joint probabilities are known — in Section 4 we shall provide two different approaches for computing them. Under this assumption and having rewritten the distortion (11) in terms of the margined update function, the minimization problem that generates the quantizer at time-step $k+1$ may be specified using the Newton-Raphson iteration

$${}^{(n+1)}\mathbf{\Gamma}_{k+1}^y = {}^{(n)}\mathbf{\Gamma}_{k+1}^y - \left[\nabla^2 D\left({}^{(n)}\mathbf{\Gamma}_{k+1}^y\right)\right]^{-1} \nabla D\left({}^{(n)}\mathbf{\Gamma}_{k+1}^y\right), \qquad (12)$$

where $\mathbf{\Gamma}_{k+1}^y$ is a column vector of the codewords in Γ_{k+1}^y and $0 \leq n < n_{\max}$ is the iteration index. Closely following McWalter et al. [2017], closed-form expressions for the gradient of the distortion, $\nabla D\left(\mathbf{\Gamma}_{k+1}^y\right)$, and the tridiagonal Hessian matrix, $\nabla^2 D\left(\mathbf{\Gamma}_{k+1}^y\right)$, may now be derived.

To summarise notation, we write the update of the dependent process as

$$\overline{\mathcal{U}}^y(x_k^i, y_k^u, Z) =: U_{k+1}^{i,u} = \overline{m}_k^{i,u} Z + \overline{c}_k^u,$$

where

$$\overline{m}_k^{i,u} := b^y(x_k^i, y_k^u)\sqrt{\Delta t} \qquad \text{and} \qquad \overline{c}_k^u := y_k^u + a^y(y_k^u)\Delta t.$$

Note that the i and j indices, for $1 \leq i, j \leq N^x$, always refer to the codewords of the quantizers for the $\widetilde{X}$-process, whereas the u and v indices, for $1 \leq u, v \leq N^y$, always refer to the codewords of the quantizers for the $\widetilde{Y}$-process.

The gradient of the distortion is given by

$$\begin{aligned}\frac{\partial D(\Gamma_{k+1}^y)}{\partial y_{k+1}^v} &= 2\sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \mathbb{E}\left[\mathbb{I}_{\left\{U_{k+1}^{i,u} \in R_{k+1}^v\right\}}(y_{k+1}^v - U_{k+1}^{i,u})\right] \mathbb{P}(\widehat{X}_k = x_k^i, \widehat{Y}_k = y_k^u) \\ &= 2\sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \int_{U_{k+1}^{i,u} \in R_{k+1}^v} (y_{k+1}^v - U_{k+1}^{i,u})\, d\mathbb{P}(Z < z)\mathbb{P}(\widehat{X}_k = x_k^i, \widehat{Y}_k = y_k^u), \qquad (13)\end{aligned}$$

where R_{k+1}^v is the region associated with codeword y_{k+1}^v. To rewrite the integration bounds in terms of the Gaussian random variable, consider that $U_{k+1}^{i,u} \in R_{k+1}^v$ implies that $U_{k+1}^{i,u}$ lies between the region boundaries of the codeword y_{k+1}^v. This means

$$r_{k+1}^{v-} < U_{k+1}^{i,u} \leq r_{k+1}^{v+} \qquad \text{and} \qquad r_{k+1}^{v\pm} := \tfrac{1}{2}(y_{k+1}^{v\pm 1} + y_{k+1}^v),$$

and $r_{k+1}^{1-} = -\infty$ and $r_{k+1}^{N^y+} = \infty$ by definition. Thus,

$$U_{k+1}^{i,u} \in R_{k+1}^{v} = \begin{cases} r_{k+1}^{i,u,v-} < Z \leq r_{k+1}^{i,u,v+} & \text{for } \overline{m}_k^{i,u} \geq 0 \\ r_{k+1}^{i,u,v-} > Z \geq r_{k+1}^{i,u,v+} & \text{for } \overline{m}_k^{i,u} < 0, \end{cases}$$

where

$$r_{k+1}^{i,u,v\pm} := \frac{r_{k+1}^{v\pm} - \overline{c}_k^u}{\overline{m}_k^{i,u}}, \tag{14}$$

is defined to be the standardized region boundary. Similar to the region boundaries of the independent process, see (4), it refers to the region boundaries of the codeword y_{k+1}^v, when viewed from the codewords x_k^i and y_k^u of the previous time step.

Let f_Z and F_Z by the PDF and CDF of a standard normal random variable Z, respectively, and define M_Z as the first lower partial expectation of Z,

$$M_Z(z) := \mathbb{E}\left[Z\mathbb{I}_{\{Z<z\}}\right].$$

Then, by direct evaluation of the integral in (13), each element of the gradient of the distortion at time-step $k+1$ is given by

$$\begin{aligned} \frac{\partial D(\Gamma_{k+1}^y)}{\partial y_{k+1}^v} = 2\sum_{i=1}^{N^x}\sum_{u=1}^{N^y}\Big[&(y_{k+1}^v - \overline{c}_k^u)\operatorname{sgn}(\overline{m}_k^{i,u})(F_Z(r_{k+1}^{i,u,v+}) - F_Z(r_{k+1}^{i,u,v-})) \\ &-|\overline{m}_k^{i,u}|(M_Z(r_{k+1}^{i,u,v+}) - M_Z(r_{k+1}^{i,u,v-}))\Big]\,\mathbb{P}(\widehat{X}_k = x_k^i, \widehat{Y}_k = y_k^u). \end{aligned} \tag{15}$$

The N^y-elements of the main diagonal of the tridiagonal Hessian matrix, $\nabla^2 D\left(\mathbf{\Gamma}_{k+1}^y\right)$, are given by

$$\begin{aligned} \frac{\partial^2 D(\Gamma_{k+1}^y)}{\partial (y_{k+1}^v)^2} = \sum_{i=1}^{N^x}\sum_{u=1}^{N^y}\Bigg[&2\operatorname{sgn}(\overline{m}_k^{i,u})(F_Z(r_{k+1}^{i,u,v+}) - F_Z(r_{k+1}^{i,u,v-})) \\ &+ \frac{1}{2|\overline{m}_k^{i,u}|} f_Z(r_{k+1}^{i,u,v+})(y_{k+1}^v - y_{k+1}^{v+1}) \\ &+ \frac{1}{2|\overline{m}_k^{i,u}|} f_Z(r_{k+1}^{i,u,v-})(y_{k+1}^{v-1} - y_{k+1}^v)\Bigg]\mathbb{P}(\widehat{X}_k = x_k^i, \widehat{Y}_k = y_k^u), \end{aligned} \tag{16}$$

with the $(N^y - 1)$-elements of the super-diagonal and sub-diagonal given by

$$\frac{\partial^2 D(\Gamma_{k+1}^y)}{\partial y_{k+1}^v \partial y_{k+1}^{v+1}} = \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \frac{1}{2|\overline{m}_k^{i,u}|} f_Z(r_{k+1}^{i,u,v+})(y_{k+1}^v - y_{k+1}^{v+1})\mathbb{P}(\widehat{X}_k = x_k^i, \widehat{Y}_k = y_k^u) \tag{17}$$

and

$$\frac{\partial^2 D(\Gamma_{k+1}^y)}{\partial y_{k+1}^v \partial y_{k+1}^{v-1}} = \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \frac{1}{2|\overline{m}_k^{i,u}|} f_Z(r_{k+1}^{i,u,v-})(y_{k+1}^{v-1} - y_{k+1}^v)\mathbb{P}(\widehat{X}_k = x_k^i, \widehat{Y}_k = y_k^u), \tag{18}$$

respectively.

The formulae above are similar to those derived for the standard RMQ case, with an additional summation over the codewords of the independent process. Thus, we again have a

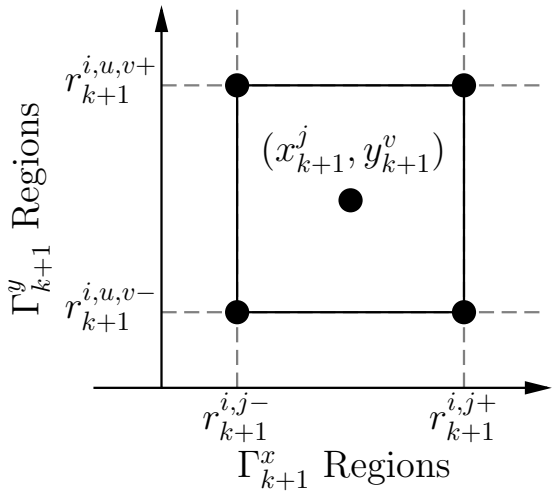

Figure 1: A standardized region for the bivariate Gaussian distribution with indices i and u fixed.

one-dimensional vector quantization problem, but this time the marginal distribution to be quantized consists of a sum of Euler updates that are weighted using joint probabilities. For this reason, we shall refer to this variant of the RMQ algorithm as the *joint* RMQ algorithm (JRMQ). This will allow us to distinguish it in the text from the standard RMQ algorithm described in Section 2.

When the above formulation is compared with the approach proposed by Callegaro et al. [2015b] (see Appendix D of their paper), it is observed that our equations have one less summation, since we do not need to condition on the independent process at time-step $k+1$. This means that the expressions for the gradient and Hessian presented here are an order of magnitude more efficient to implement.

4 Computing the Joint Probabilities

Up to this point, we have assumed that the joint probabilities required in (15) to (18) are available. In this section, we shall show how to compute these probabilities exactly and using a computationally efficient approximation. To facilitate efficient implementation, we also provide a matrix formulation of the system in Section 5.1.

From (7) and (8) it is evident that, conditional on the realizations of $\widetilde{X}_k$ and $\widetilde{Y}_k$, the joint probability distribution of $\widetilde{X}_{k+1}$ and $\widetilde{Y}_{k+1}$ is bivariate Gaussian. We define

$$Z^y_{k+1} := \rho Z^x_{k+1} + \sqrt{1-\rho^2} Z^{\perp}_{k+1},$$

such that $\widetilde{Y}_{k+1} = \overline{\mathcal{U}}(\widetilde{X}_k, \widetilde{Y}_k, Z^y_{k+1})$.

Consider the joint probability of $\widetilde{X}_{k+1}$ and $\widetilde{Y}_{k+1}$ in the form

$$\begin{aligned}
&F_{\widetilde{X}_{k+1},\widetilde{Y}_{k+1}}(x,y) \\
&\quad= \int_{\mathbb{R}^2} \mathbb{P}(\mathcal{U}^x(r, Z^x_{k+1}) \le x, \overline{\mathcal{U}}^y(r,s,Z^y_{k+1}) \le y)\, d\mathbb{P}(\widetilde{X}_k \le r, \widetilde{Y}_k \le s) \\
&\quad\approx \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \mathbb{P}(\mathcal{U}^x(x^i_k, Z^x_{k+1}) \le x, \overline{\mathcal{U}}^y(x^i_k, y^u_k, Z^y_{k+1}) \le y)\mathbb{P}(\widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k) \qquad (19) \\
&\quad= \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \mathbb{P}\left(Z^x_{k+1} \le \frac{x - c^i_k}{m^i_k}, Z^y_{k+1} \le \frac{y - \overline{c}^u_k}{\overline{m}^{i,u}_k}\right) \mathbb{P}(\widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k).
\end{aligned}$$

The approximation in (19) is formed by replacing the continuous, and unknown, distributions of $\widetilde{X}_k$ and $\widetilde{Y}_k$ with the discrete, and known quantized distributions of $\widehat{X}_k$ and $\widehat{Y}_k$, as in the standard RMQ case. The necessary joint probability is then given by

$$\begin{aligned}
&\mathbb{P}(\widetilde{X}_{k+1} = x^j_{k+1}, \widetilde{Y}_{k+1} = y^v_{k+1}) \\
&\qquad= \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \left[\int_{r^{i,u,v-}_k}^{r^{i,u,v+}_k} \int_{r^{i,j-}_k}^{r^{i,j+}_k} \phi_2(x,y,\rho)\, dx\, dy\right] \mathbb{P}(\widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k),
\end{aligned} \qquad (20)$$

where $\phi_2(x,y,\rho)$ is the bivariate Gaussian density function for two standard Gaussian random variables correlated by ρ. The double integral above refers to the probability of a rectangle delimited by the standardized regions of Γ^x_{k+1} and Γ^y_{k+1}, see Figure 1. Therefore, for each $1 \le j \le N^x$ and $1 \le v \le N^y$,

$$\begin{aligned}
\mathbb{P}(\widetilde{X}_{k+1} = x^j_{k+1}, \widetilde{Y}_{k+1} = y^v_{k+1}) = \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \Big[&\Phi_2\big(r^{i,j+}_k, r^{i,u,v+}_k, \rho\big) - \Phi_2\big(r^{i,j-}_k, r^{i,u,v+}_k, \rho\big) \\
&-\Phi_2\big(r^{i,j+}_k, r^{i,u,v-}_k, \rho\big) + \Phi_2\big(r^{i,j-}_k, r^{i,u,v-}_k, \rho\big)\Big] \\
&\times \mathbb{P}(\widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k),
\end{aligned} \qquad (21)$$

where $\Phi_2(x,y,\rho)$ is the standard bivariate Gaussian cumulative distribution function with correlation ρ evaluated at x and y.

Given the quantizers at time k, the joint probability in (21) is exact. However, it requires the evaluation of the bivariate Gaussian distribution function. Although most programming languages have an efficient implementation of this function, it is significantly more expensive to compute than the univariate distribution. The joint probability can be approximated using only calls to the univariate Gaussian CDF by using quadrature to approximate the inner integral of (20).

While other approaches are possible, a simple quadrature rule is used by replacing $\widetilde{X}_{k+1}$

with its quantized version, $\widehat{X}_{k+1}$, which is constant over the interval. Then (20) becomes

$$\begin{aligned}
&\mathbb{P}(\widetilde{X}_{k+1} = x^j_{k+1}, \widetilde{Y}_{k+1} = y^v_{k+1}) \\
&\quad \approx \sum_{i=1}^{N^x}\sum_{u=1}^{N^y}\left[\int_{r^{i,u,v-}_k}^{r^{i,u,v+}_k} \phi_2\big(y,\rho\big|\tfrac{x^j_{k+1}-c^i_k}{m^i_k}\big)dy\right]\mathbb{P}(\widehat{X}_{k+1}=x^j_{k+1})\mathbb{P}(\widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k) \\
&\quad = \sum_{i=1}^{N^x}\sum_{u=1}^{N^y}\left[F_Z\left(\frac{r^{i,u,v+}_k - \rho\frac{x^j_{k+1}-c^i_k}{m^i_k}}{\sqrt{1-\rho^2}}\right) - F_Z\left(\frac{r^{i,u,v-}_k - \rho\frac{x^j_{k+1}-c^i_k}{m^i_k}}{\sqrt{1-\rho^2}}\right)\right] \\
&\qquad\qquad \times \mathbb{P}(\widehat{X}_{k+1}=x^j_{k+1})\mathbb{P}(\widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k),
\end{aligned} \tag{22}$$

where $\phi_2(y,\rho|x)$ is the conditional bivariate Gaussian density. It is worthwhile to note that this approximation to the joint probability, although derived differently, is identical to that of Callegaro et al. [2015b].

The computational efficiency of this approximation is demonstrated in the Sections 6 and 7.

5 Implementing the Algorithm

In this section a concise matrix formulation for the JRMQ algorithm is presented, similar to that provided in McWalter et al. [2017] for the standard RMQ case.

5.1 Matrix Formulation

Throughout this section, the index $1 \le i \le N^x$ refers to time-step k and $1 \le j \le N^x$ refers to time-step $k+1$, and both are associated with the $\widetilde{X}$-process. For the $\widetilde{Y}$-process, the index $1 \le u \le N^y$ refers to time-step k and the index $1 \le v \le N^y$ refers to time-step $k+1$.

To initialize the JRMQ algorithm, the standard RMQ algorithm is applied to the $\widetilde{X}$-process and yields the quantizers $\mathbf{\Gamma}^x_k$ and associated probabilities $\mathbf{p}^x_k$ at each time-step $0 \le k \le K$. The following three variables are initialized

$$[\mathbf{\Gamma}^y_0]_1 = y_0, \qquad [\mathbf{p}^x_0]_1 = 1, \qquad [\mathbf{J}_0]_{1,1} = 1,$$

being the time-zero quantizer, associated probability and margined probability, respectively, of the $\widetilde{Y}$-process. The standard one-dimensional vector quantization algorithm (on the normal distribution) is used to produce $\mathbf{\Gamma}^y_1$ and $\mathbf{p}^y_1$, being the quantizer and associated probability vector of the $\widetilde{Y}$-process at the first time step. The corresponding joint probabilities at time-step one may then be computed using either (27) or (30) with $k = 0$ and $N^x = N^y = 1$.

We now describe the implementation of the recursive step form time-step k to $k+1$. Consider the time-step k quantizers

$$[\mathbf{\Gamma}^x_k]_i = x^i_k \qquad \text{and} \qquad [\mathbf{\Gamma}^y_k]_u = y^u_k,$$

of the independent and dependent processes, respectively, and the associated joint probability matrix $\mathbf{J}_k$, of size $N^x \times N^y$,

$$[\mathbf{J}_k]_{i,u} = \mathbb{P}(\widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k),$$

all of which are assumed known (already computed). The rows of $\mathbf{J}_k$ are denoted by $\mathbf{J}_k^{(i)}$.

The time-step $k+1$ quantizer for the dependent process and associated probabilities are computed as follows: Aside from an initial guess for $\mathbf{\Gamma}_{k+1}^y$, which is taken to be $\mathbf{\Gamma}_k^y$, we initialize the N^y-element column vector

$$[\mathbf{c}_k]_u = \overline{c}_k^u$$

and the set of N^y-element column vectors, indexed by i,

$$[\mathbf{m}_k]_u^{(i)} = \overline{m}_k^{i,u},$$

in terms of the time-step k quantities listed above. For each iteration of the Newton-Raphson algorithm, three sets of matrices, indexed by i, are computed. The first two sets have matrices of size $N^y \times N^y$, given by

$$\begin{aligned}[\mathbf{P}_{k+1}]_{u,v}^{(i)} &= \mathbb{P}(\widehat{Y}_{k+1} = y_{k+1}^v | \widehat{X}_k = x_k^i, \widehat{Y}_k = y_k^u), \\ &= F_Z(r_{k+1}^{i,u,v+}) - F_Z(r_{k+1}^{i,u,v-})\end{aligned}$$

and

$$[\mathbf{M}_{k+1}]_{u,v}^{(i)} = M_Z(r_{k+1}^{i,u,v+}) - M_Z(r_{k+1}^{i,u,v-}),$$

while the third has matrices of size $N^y \times (N^y - 1)$, given by

$$[\mathbf{f}_{k+1}]_{u,v}^{(i)} = f_Z(r_{k+1}^{i,u,v+}).$$

The above matrices allow the gradient and the Hessian of the distortion for $\mathbf{\Gamma}_{k+1}^y$ to be written in simplified form. The N^y-element gradient vector is

$$\nabla D(\mathbf{\Gamma}_{k+1}^y)^\top = \sum_{i=1}^{N^x} 2\mathbf{J}_k^{(i)}(((\mathbf{\Gamma}_{k+1}^y \mathbf{1}_{N^y})^\top - \mathbf{c}_k \mathbf{1}_{N^y}) \circ \mathbf{P}_{k+1}^{(i)} - (|\mathbf{m}_k^{(i)}| \mathbf{1}_{N^y}) \circ \mathbf{M}_{k+1}^{(i)}), \tag{23}$$

where $\circ$ is the Hadamard (or element-wise) product and $\mathbf{1}_z$ is defined to be a length-z row vector of ones. By specifying the column vector

$$[\Delta\mathbf{\Gamma}_{k+1}^y]_v = y_{k+1}^{v+1} - y_{k+1}^v, \tag{24}$$

with $1 \le v \le (N^y - 1)$, the $(N^y - 1)$-element off-diagonal of the tridiagonal Hessian matrix is given by

$$\mathbf{h}_{\mathsf{off}} = \sum_{i=1}^{N^x} -\frac{1}{2}\mathbf{J}_k^{(i)}((|\mathbf{m}_k^{(i)}|^{\circ -1} \mathbf{1}_{N^y - 1}) \circ \mathbf{f}_{k+1}^{(i)} \circ (\Delta\mathbf{\Gamma}_{k+1}^y \mathbf{1}_{N^y})^\top) \tag{25}$$

and the N^y-element main diagonal by

$$\mathbf{h}_{\mathsf{main}} = \sum_{i=1}^{N^x} 2\mathbf{J}_k^{(i)} \mathbf{P}_{k+1}^{(i)} + [\mathbf{h}_{\mathsf{off}} | 0] + [0 | \mathbf{h}_{\mathsf{off}}]. \tag{26}$$

Here, $\circ - 1$ refers to the element-wise inverse.

Equations (23) to (26) provide a matrix representation of equations (15) to (18) and correspond to those in the matrix implementation of the single-factor RMQ case. This allows straightforward implementation of the Newton-Raphson algorithm described by (12), ultimately yielding $\mathbf{\Gamma}^y_{k+1}$. It remains to compute the necessary probabilities.

The elements of the joint probability matrix, $\mathbf{J}_{k+1}$, at time-step $k+1$, are computed using the bivariate Gaussian distribution as

$$\begin{aligned}
[\mathbf{J}_{k+1}]_{j,v} &= \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \mathbb{P}(\widehat{X}_{k+1} = x^j_{k+1}, \widehat{Y}_{k+1} = y^v_{k+1} | \widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k)\mathbb{P}(\widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k) \\
&= \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \Big(\Phi_2(r^{i,j+}_k, r^{i,u,v+}_k, \rho) - \Phi_2(r^{i,j-}_k, r^{i,u,v+}_k, \rho) \\
&\qquad -\Phi_2(r^{i,j+}_k, r^{i,u,v-}_k, \rho) + \Phi_2(r^{i,j-}_k, r^{i,u,v-}_k, \rho)\Big)\,[\mathbf{J}_k]_{i,u},
\end{aligned} \tag{27}$$

with the probabilities associated with the new quantizer given by

$$\mathbf{p}^y_{k+1} = \sum_{j=1}^{N^x} \mathbf{J}^{(j)}_{k+1}.f \tag{28}$$

Finally, to compute the transition probability matrix for the time-step $k+1$, it is necessary to recompute the $\mathbf{P}_{k+1}$ matrix using the final regions associated with the new set of codewords at $k+1$. Then

$$\begin{aligned}
[\mathbf{P}^y_{k+1}]_{u,v} &= \frac{\mathbb{P}(\widehat{Y}_k = y^u_k, \widehat{Y}_{k+1} = y^v_{k+1})}{\mathbb{P}(\widehat{Y}_k = y^u_k)} \\
&= \frac{\sum_{i=1}^{N^x} \mathbb{P}(\widehat{Y}_{k+1} = y^v_{k+1} | \widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k)\mathbb{P}(\widehat{X}_k = x^i_k, \widehat{Y}_k = y^u_k)}{\mathbb{P}(\widehat{Y}_k = y^u_k)} \\
&= \frac{\sum_{i=1}^{N^x} [\mathbf{P}_{k+1}]^{(i)}_{u,v}[\mathbf{J}_k]_{i,u}}{[\mathbf{p}^y_k]_u}.
\end{aligned} \tag{29}$$

To compute the joint probabilities using the computationally efficient approximation instead of the bivariate Gaussian distribution, (27) is replaced by

$$[\mathbf{J}_{k+1}]_{j,v} = \sum_{i=1}^{N^x}\sum_{u=1}^{N^y} \left[F_Z\left(\frac{r^{i,u,v+}_k - \rho\frac{x^j_{k+1}-c^i_k}{m^i_k}}{\sqrt{1-\rho^2}}\right) - F_Z\left(\frac{r^{i,u,v-}_k - \rho\frac{x^j_{k+1}-c^i_k}{m^i_k}}{\sqrt{1-\rho^2}}\right)\right][\mathbf{p}^x_{k+1}]_j[\mathbf{J}_k]_{i,u}. \tag{30}$$

The time-step $k+1$ quantizer probabilities and transition probability matrix, (28) and (29), are now computed in terms of (30).

5.2 Zero Boundary Behaviour

As in the standard RMQ algorithm, to correctly model the underlying processes it may be necessary to implement a reflecting or absorbing boundary at zero. Both the dependent and independent process can be modified in this way, but, once the distribution of either process has been adjusted, it is difficult to compute the joint probabilities using the bivariate Gaussian distribution. Thus, the joint probability approximation (22) is used.

The Independent Process

In the stochastic volatility setting, the independent process represents the stochastic volatility or variance of the dependent process. This implies that it must remain strictly positive and thus it is only necessary to consider a reflecting boundary. In Monte Carlo simulation a reflecting boundary is modelled by the fully-truncated Euler scheme, shown to be the least-biased scheme for stochastic volatility models in Lord et al. [2010][1].

Implementing a reflecting boundary in the standard RMQ case is discussed in detail in McWalter et al. [2017] and modifying the independent process in this way leaves the JRMQ algorithm unchanged.

The Dependent Process

As the dependent process usually represents either an asset price or an interest rate, depending on the application, either a reflecting or absorbing boundary at zero may be appropriate. When the process modeled is an asset price, an absorbing boundary allows the possibility of bankruptcy, whereas a reflecting boundary is necessary to correctly model interest rates.

Modifying the algorithm to account for an absorbing boundary at zero is straightforward and incurs no additional computational burden. To ensure the non-negativity of the process, the domain of the marginal distribution implied by the quantizer at time k is smaller than zero if

$$Z < -\frac{\overline{c}_k^u}{\overline{m}_k^{i,u}},$$

which implies that, under the requirement to ensure positive codewords at the next time step, the left-most region boundary must be set to

$$r_{k+1}^{i,u,1-} = -\frac{\overline{c}_k^u}{\overline{m}_k^{i,u}},$$

for $1 \leq i \leq N^x$ and $1 \leq u \leq N^y$. This is equivalent to setting $r_{k+1}^{1-} = 0$ and thus truncates the domain, cf. (14).

Truncating the domain of the implied marginal distribution at each time step will result in quantizers with probabilities that do not sum to one. This is because there is now effectively an additional codeword at zero, at which probability has accumulated. At the completion of the algorithm, the quantizers can be augmented with this additional codeword and its associated probability.

To model a reflecting boundary at zero, first the domain of the implied marginal distribution at each time step must be modified such that only positive codewords can be attained. This is achieved by altering the left-most region boundary as above. Secondly, the distribution, density and lower partial expectation functions that appear in (15) to (18) must be replaced by their reflected counterparts,

$$f_{\overline{Z}_{k+1}^{i,u}}(y) = f_Z(y) + f_Z(2\bar{y}_k^{i,u} - y),$$
$$F_{\overline{Z}_{k+1}^{i,u}}(y) = F_Z(y) - F_Z(2\bar{y}_k^{i,u} - y),$$

[1]Note that a *truncated* Euler scheme models *reflecting* boundary behaviour.

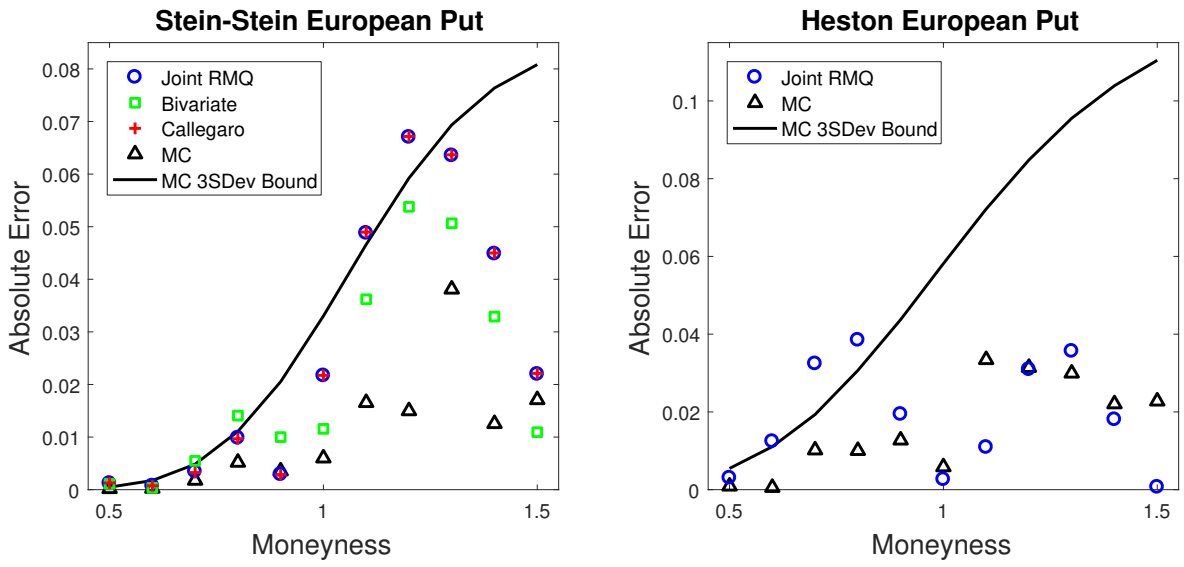

Figure 2: The European put pricing error under the Stein-Stein and Heston models.

and

$$M_{\overline{Z}_{k+1}^{i,u}}(y) = M_Z(y) + M_Z(2\bar{y}_k^{i,u} - y) - 2\bar{y}_k^{i,u} F_Z(2\bar{y}_k^{i,u} - y),$$

for $y \in [\bar{y}_k^{i,u}, \infty)$, where $\bar{y}_k^{i,u} = -\frac{\bar{c}_k^u}{\bar{m}_k^{i,u}}$. Note that these functions have an i and u subscript, indicating that they will be different for each term in the summations of (15) to (18).

6 Pricing European Options

In this section, we consider the pricing of European options under the Stein and Stein [1991], Heston [1993] and SABR [Hagan et al., 2002] models. The Stein-Stein and Heston models are both amenable to semi-analytical pricing using Fourier transform techniques, whereas an analytical approximation exists for both the Black and Bachelier implied volatility under the SABR model. The Fourier pricing technique implemented uses the little trap formulation of the characteristic function from Albrecher et al. [2006] for the Heston model, while the Schöbel and Zhu [1999] characteristic function formulation is used for the Stein-Stein model. The implied volatility approximation for the SABR model is the latest from Hagan et al. [2016].

The Stein-Stein example is used to illustrate the computational efficiency advantage of the new algorithm compared to the RMQ algorithm from Callegaro et al. [2015b], whereas the Heston example serves to highlight the effectiveness of correctly modelling the zero-boundary behaviour of the independent process. For the SABR model, parameter sets were chosen that are difficult to handle with traditional methods, illustrating the flexibility of the JRMQ algorithm.

All simulations were executed using MATLAB 2016b on a computer with a 2.00 GHz Intel i-3 processor and 4 GB of RAM. All Monte Carlo simulations in this section used 500 000 paths with 120 time steps per path.

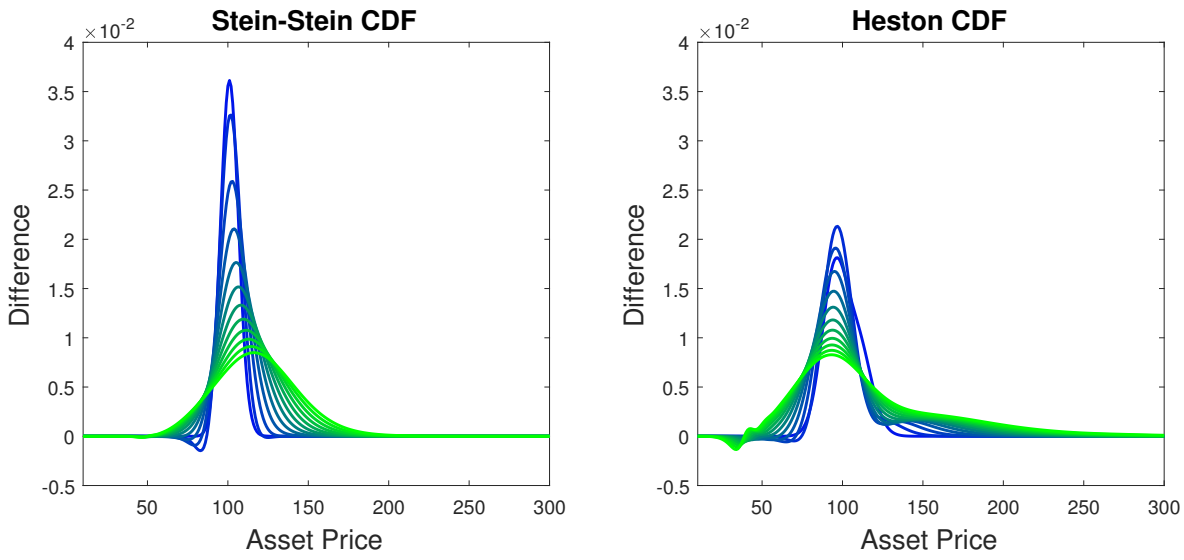

Figure 3: The error in the marginal distributions for the dependent process in the Stein-Stein and Heston models.

6.1 The Stein and Stein Model

The SDEs for the Stein-Stein model may be specified in the notation of (5) and (6) as

$$a^x(X_t) = \kappa(\theta - X_t), \qquad b^x(X_t) = \sigma,$$
$$a^y(Y_t) = rY_t, \qquad b^y(X_t, Y_t) = X_t Y_t,$$

and in the example considered the parameters chosen are $\kappa = 4$, $\theta = 0.2$, $\sigma = 0.1$, $r = 0.0953$, $\rho = -0.5$, $x_0 = 0.2$ and $y_0 = 100$, with the maturity of the option set at one year. These parameters are from Table 1 in Schöbel and Zhu [1999].

The left graph in Figure 2 displays the pricing error of four algorithms. The first is the JRMQ algorithm presented in this paper using the joint probability approximation from (22), the second is the JRMQ algorithm using the bivariate Gaussian distribution, the third is the stochastic volatility RMQ algorithm from Callegaro et al. [2015b] and the fourth is a two-dimensional standard Euler Monte Carlo simulation.

For the RMQ algorithms, $K = 12$ time steps were used with $N^x = 30$ codewords at each step for the independent process and $N^y = 60$ codewords for the dependent process. We consider variable moneyness by changing the strike over the fixed initial asset price.

The JRMQ algorithm took 3.8 seconds to price all strikes when using the probability approximation and 77.2 seconds when using the bivariate Gaussian distribution. The algorithm from Callegaro et al. [2015b] took 26.3 seconds to price all strikes and the Monte Carlo simulation took 6.6 seconds per strike.

The computation time of the JRMQ algorithm for this example was approximately 7 times faster than the algorithm of Callegaro et al. [2015b], when using approximate joint probabilities. Despite this large decrease in computation time, the JRMQ algorithm prices with the same accuracy. Barring three points, both algorithms price to within the three standard deviation bound of the significantly higher resolution Monte Carlo simulation. Using the bivariate Gaussian distribution instead of the approximation significantly reduces the average

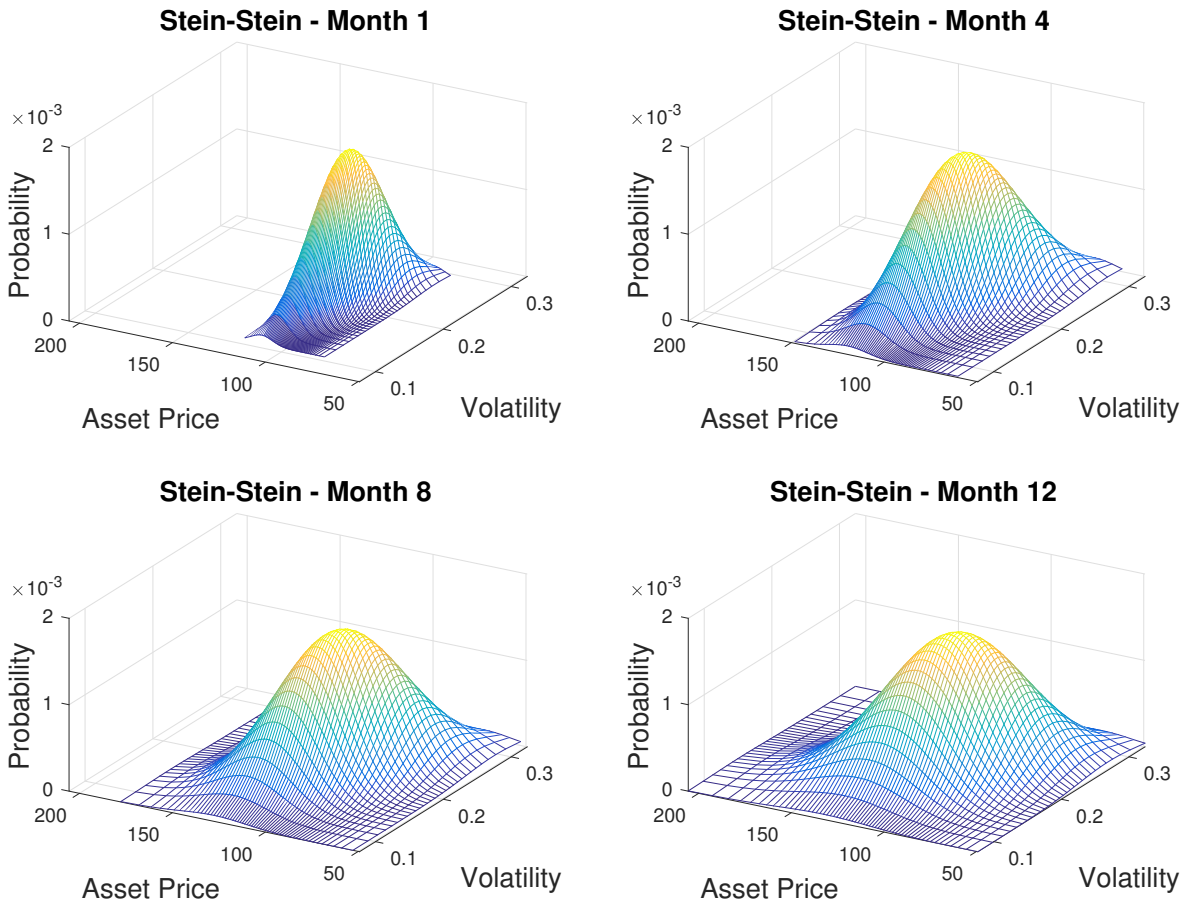

Figure 4: Evolution of the approximate joint probability for the Stein-Stein model.

error over the range of moneyness considered, but this is at the expense of a large increase in computation time. For this reason, the remaining applications use only the approximation.

Since the Stein-Stein model has a closed-form characteristic function, it is possible to compute the marginal distribution for the dependent process. The difference between this marginal distribution and the one computed using the JRMQ algorithm is presented in the left graph in Figure 3. The curve is blue at the initial time and changes color to green as we move toward maturity. The maximum error is under 4% initially and decays to well under 1% as time advances. These errors are in line with those of the one-dimensional Euler RMQ case illustrated in McWalter et al. [2017].

Figure 4 illustrates the evolution of the approximate joint probabilities over time. Note that these are the joint probabilities associated with the quantizers and thus the grid is not uniform; there are 60 points along the asset price axis and 30 points along the volatility axis.

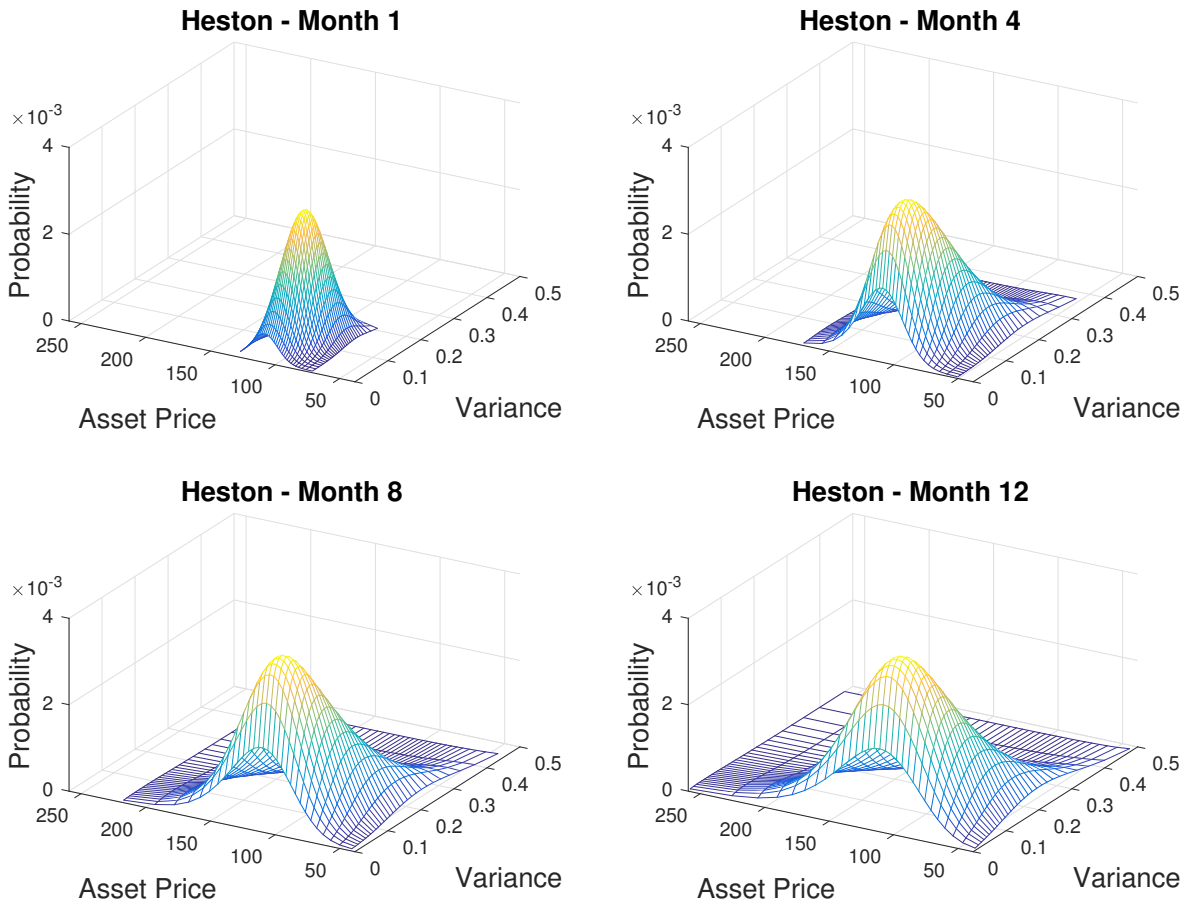

Figure 5: Evolution of the approximate joint probability for the Heston model.

6.2 The Heston Model

The SDEs for the Heston model may be specified in the notation of (5) and (6) as

$$a^x(X_t) = \kappa(\theta - X_t), \qquad b^x(X_t) = \sigma\sqrt{X_t},$$
$$a^y(Y_t) = rY_t, \qquad b^y(X_t, Y_t) = \sqrt{X_t}Y_t,$$

and in the example considered the parameters chosen are $\kappa = 2$, $\theta = 0.09$, $\sigma = 0.4$, $r = 0.05$, $\rho = -0.3$, $x_0 = 0.09$ and $y_0 = 100$, with the maturity of the option set at one year. These parameters are based on the SV-I parameter set from Table 3 of Lord et al. [2010], with σ adjusted from 1 to 0.4 to ensure that the Feller condition is satisfied for the square-root variance process.

The right graph in Figure 2 displays the pricing error for JRMQ compared with a two-dimensional fully truncated log-Euler scheme, suggested as the least-biased Monte Carlo scheme for stochastic volatility models in Lord et al. [2010]. For the JRMQ algorithm, $K = 12$ time steps were used with $N^x = N^y = 30$ codewords at each step for both processes. The JRMQ algorithm took 1.4 seconds to price all strikes, whereas the Monte Carlo simulation took 7.8 seconds for a single strike.

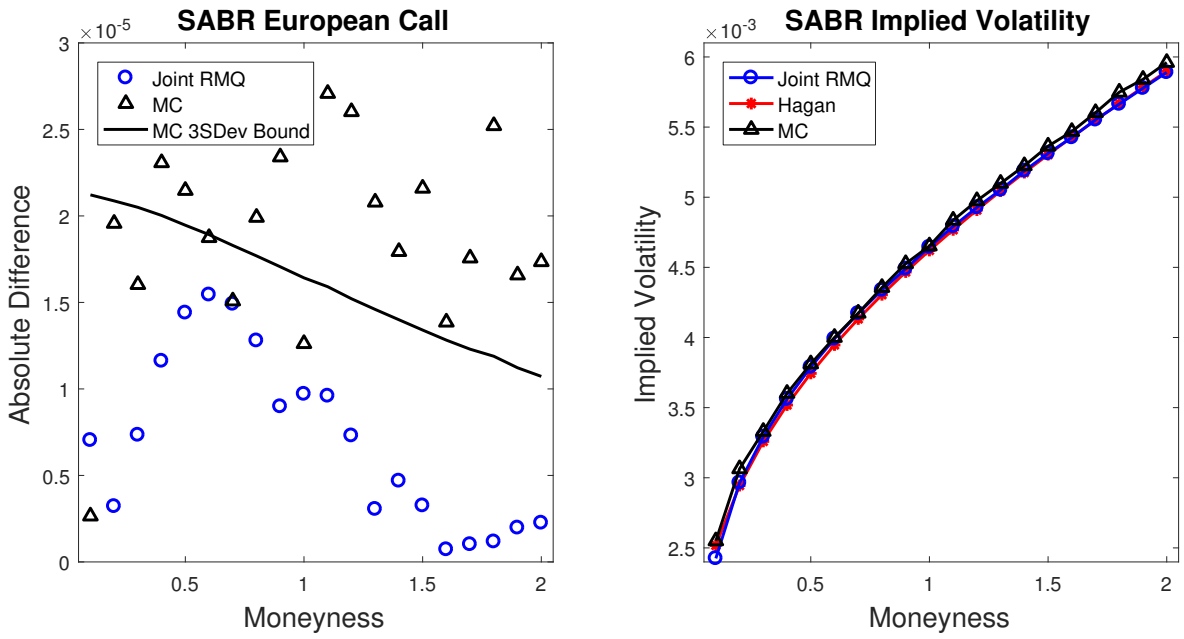

Figure 6: Prices and implied Bachelier volatilities for the standard SABR model, using a parameter set applicable for interest rates.

A reflecting zero-boundary was used when computing the standard RMQ algorithm for the independent variance process. Compared to a high-resolution Monte Carlo simulation, the JRMQ algorithm performs very well despite the coarseness of the grid.

Even though the Feller condition is satisfied, due to the discretization of time, there is a non-zero probability of the Euler approximation for the variance process becoming negative. This is handled in the RMQ algorithm by using a reflecting zero-boundary. Modelling the boundary in this way leads to an increased accuracy in pricing, especially when compared to the Monte Carlo simulation.

The right graph in Figure 3 presents the error in the marginal distribution of the dependent process implied by the RMQ algorithm when compared to the distribution obtained from the characteristic function using the Fourier transform technique. The error is just over 2% initially and decreases to below 1% as time advances. Figure 5 illustrates the evolution of the joint probabilities of the asset price and variance process.

6.3 The SABR Model

The SDEs for the standard SABR model may be specified in the notation of (5) and (6) as

$$a^x(X_t) = 0, \qquad b^x(X_t) = \nu X_t,$$
$$a^y(Y_t) = 0, \qquad b^y(X_t, Y_t) = X_t Y_t^{\beta},$$

with $0 \leq \beta \leq 1$. A partial reason for the popularity of the SABR model is that the implied volatility may be computed using an analytical approximation [Hagan et al., 2002]. Further work has extended the original formula (see, for example, Oblój [2007] and Paulot [2015]),

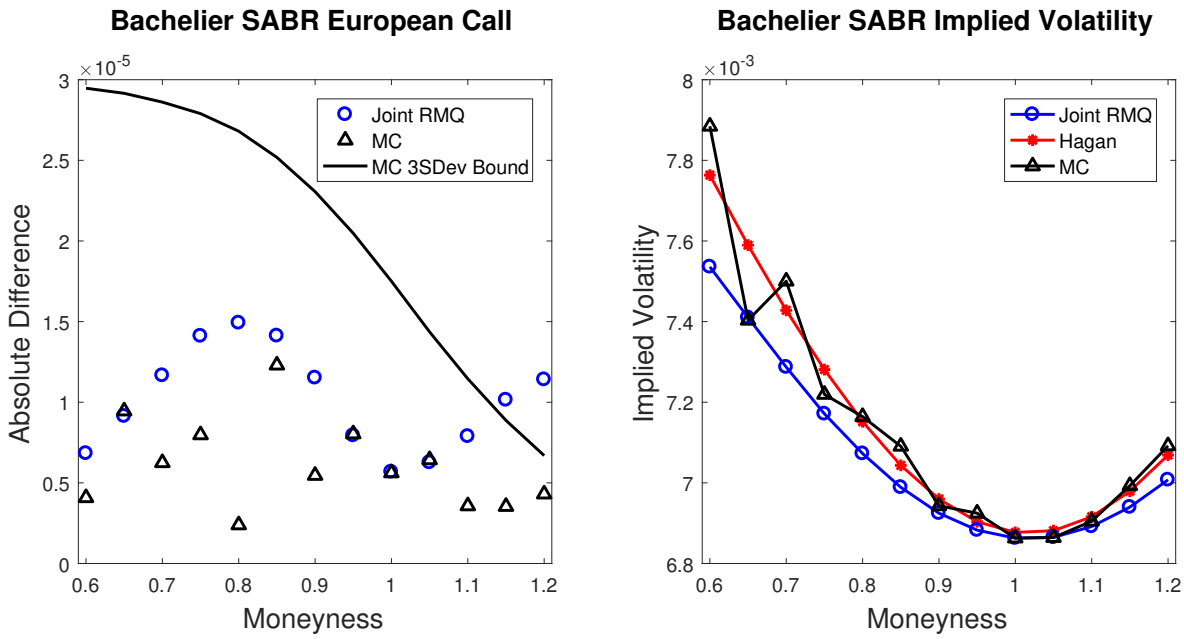

Figure 7: Implied Bachelier volatility and pricing error for the Bachelier SABR model.

with the latest and most accurate approximation given in Hagan et al. [2016], which allows a more general specification of the volatility function.

In this section, we consider European options for two examples of extreme parameter sets that may arise in the context of interest rate modelling.

In Figure 6 the parameters chosen are $\beta = 0.7$, $\nu = 0.3$, $\rho = -0.3$, $x_0 = 20\%$ and $y_0 = 0.5\%$, with the maturity of the option set at one year. This parameter set is Test Case III from Chen et al. [2012], and was specifically chosen to be appropriate to the fixed income market and to illustrate the correct handling of zero-boundary behaviour. The reference price is the implied volatility formula with the boundary correction from Hagan et al. [2016].

For the JRMQ algorithm, $K = 24$ time steps were used with $N^x = N^y = 30$ codewords at each step for both processes. A reflecting zero-boundary was implemented for the dependent process. The Monte Carlo simulation utilized a fully-truncated Euler discretization scheme.

The three standard deviation bound in the left graph in Figure 6 indicates that the Monte Carlo simulation is not converging to the same result as the Hagan et al. [2016] implied volatility, used here as the reference price. In their discussion, Chen et al. [2012] indicate that this is a challenging parameter set for traditional Monte Carlo simulations. Barring a single point, the JRMQ algorithm is more accurate than the Monte Carlo simulation across the range of strikes. It is also significantly faster to compute. The JRMQ algorithm took 5.3 seconds to price all strikes, whereas the Monte Carlo simulation took 13.4 seconds per strike, due to the much larger number of time steps.

In Figure 7, European call option prices and corresponding implied Bachelier volatilities are displayed for the RMQ algorithm, the Hagan implied volatility approximation, and an Euler Monte Carlo simulation. The parameters chosen are $\beta = 0$, $\nu = 0.3691$, $\rho = -0.0286$, $X_0 = 0.68\%$, $Y_0 = 4.35\%$, with the maturity of the option set at one year. This parameter set is Test Case I from Korn and Tang [2013] and it describes a challenging simulation

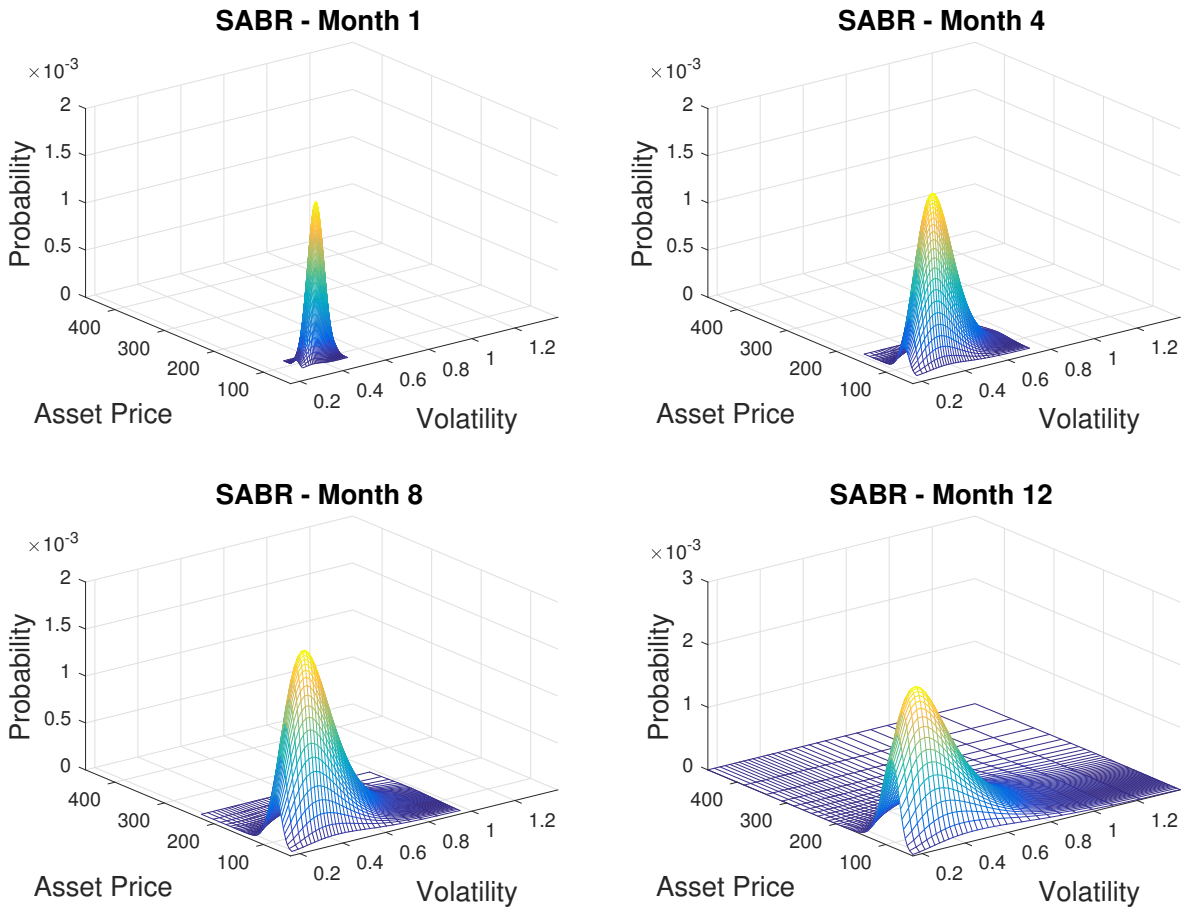

Figure 8: Time-evolution of the approximate joint probability for the SABR model.

environment with a low initial forward rate which is very volatile.

For the JRMQ algorithm, $K = 24$ time steps were used with $N^x = 10$ codewords at each step for the independent process and $N^y = 90$ codewords for the dependent process. The JRMQ algorithm took 5.5 seconds to price all strikes, whereas the Monte Carlo simulation took 5.6 seconds per strike.

Despite the extreme parameter set, all but two of the JRMQ prices fall well within the three standard deviation bound of the much higher resolution Monte Carlo simulation.

7 Pricing Exotic Options

An advantage of the RMQ algorithm, similar to binomial and trinomial tree methods, is the ability to price many options off the same grid that results from a single run. This is demonstrated in this section by using a single pass of the JRMQ algorithm to price European, Bermudan and barrier options, and volatility corridor swaps.

The SABR model parameters for all the examples in this section are $\beta = 0.9$, $\nu = 0.4$, $\rho = -0.3$, $X_0 = 0.4$ and $Y_0 = S_0 \exp(rT)$, where Y now models the T-forward price of an equity asset with $S_0 = 100$, $r = 0.05$ and the maturity T is equal to one year. The JRMQ

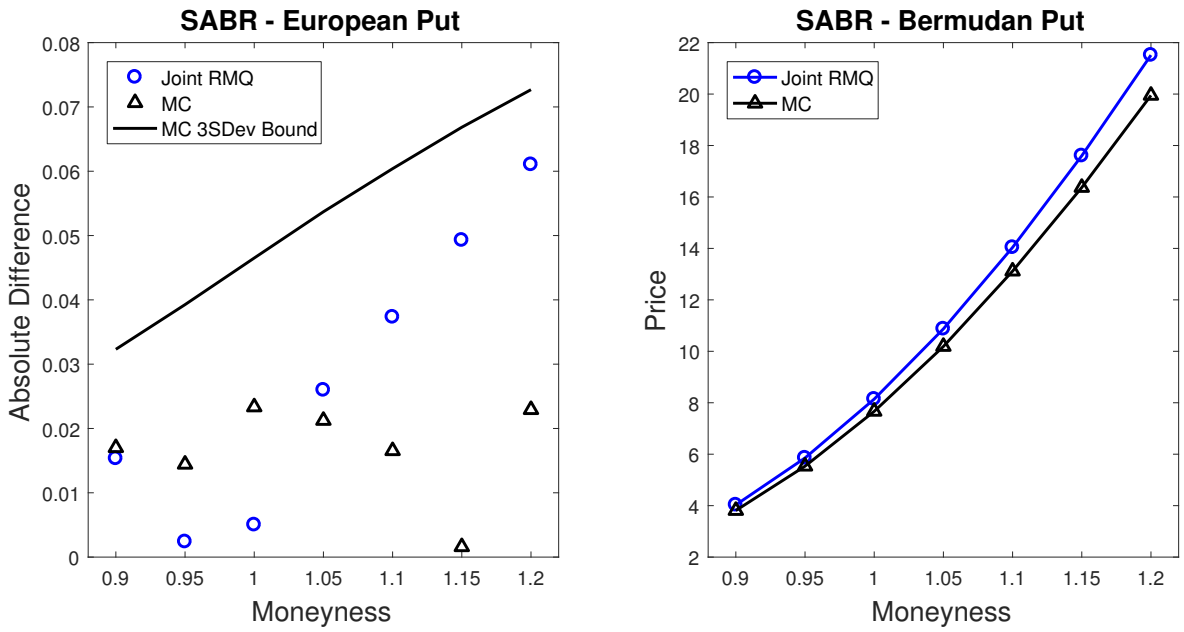

Figure 9: European and Bermudan put option prices for the SABR model.

algorithm used $K = 24$ time steps with $N^x = 30$ codewords for the volatility process and $N^y = 60$ codewords for the forward price process. The Monte Carlo simulations are executed using a fully-truncated Euler scheme with 500 000 paths and 120 time-steps.

To generate the quantization grid, the JRMQ algorithm took 7.8 seconds for these parameters. The computational cost of generating derivative prices using the resulting grid is negligible in comparison. Figure 8 illustrates the time-evolution of the approximate joint probability of the forward price of the asset and the volatility over the course of the year.

The left graph in Figure 9 illustrates the difference in the prices of European put options using JRMQ and the prices using the implied volatility formula of Hagan et al. [2016]. The right graph shows the prices for a Bermudan put with monthly exercise opportunities using JRMQ and a least-squares Monte Carlo simulation. For each strike, computing an option price using Monte Carlo simulation takes approximately 14.5 seconds for the European options and 16.9 seconds for the Bermudan options. The high-level algorithm for pricing Bermudan options using a quantization grid is outlined in McWalter et al. [2017].

The left graph in Figure 10 shows the JRMQ and Monte Carlo prices for a discrete up-and-out put option, with monthly monitoring, where the barrier level is expressed as a multiple of the at-the-money strike. The right graph shows the prices for a series of volatility corridor swaps. The payoff of a volatility corridor swap is given by

$$\frac{1}{T}\int_0^T X_z \mathbb{I}_{\{L<S_z<H\}}\, dz, \tag{31}$$

where $S_t = Y_t \exp(-r(T-t))$ is the asset price in our deterministic interest-rate framework and L and H specify the corridor of the asset price in which the volatility is accumulated. The algorithm for pricing volatility corridor swaps on a quantization grid in the stochastic volatility setting is presented in Callegaro et al. [2015b] and uses a left-endpoint approximation

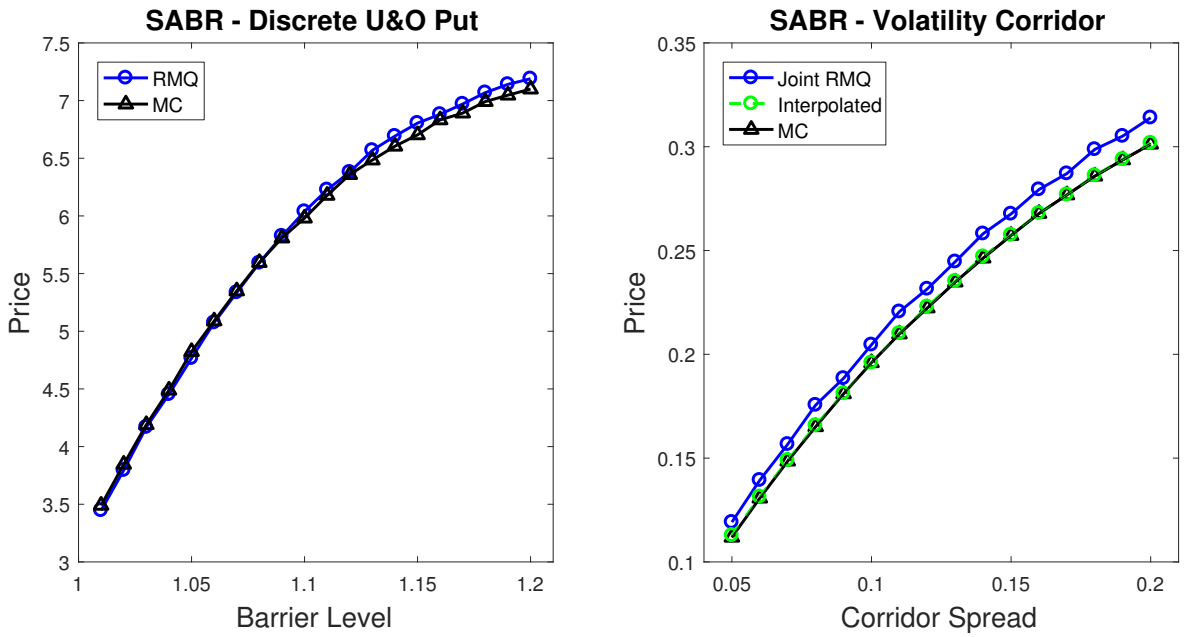

Figure 10: Price comparison for discrete up-and-out put options and volatility corridor swaps in the SABR model.

to the integral in (31)

The corridor spreads on the x-axis represent a percentage bound around the initial asset price value, i.e., the lower bound of the corridor is given by $L = S_0(1-s)$ and the upper bound by $H = S_0(1+s)$, where s is the corridor spread. The vertical gap between the prices generated by the Monte Carlo simulation and the RMQ algorithm is partially due to the increased accuracy of the Monte Carlo simulation when using simple quadrature to approximate (31), as a result of the large number of time steps used. For a single barrier value or a single corridor spread, the Monte Carlo simulation takes approximately 15.2 seconds and 16.3 seconds to price these derivatives.

The accuracy of JRMQ volatility corridor swap prices can be improved without using additional time steps. An increase in the accuracy of the approximation to the integral (31) is achieved by interpolating both the asset price and the volatility over each interval, see Appendix A. The improved accuracy of this interpolated JRMQ price is displayed in the right graph of Figure 10.

8 Conclusion

In this work, we present a Joint Recursive Marginal Quantization algorithm for stochastic volatility models that provides a significant computational advantage over the most recent developments in this area.

The central idea is to margin over, and effectively *undo*, the Cholesky decomposition in the two-dimensional Euler scheme when performing the quantization. We show how the joint probabilities can be computed exactly and using a computationally efficient approximation.

A concise matrix formulation was provided for efficient implementation. The robustness

of the algorithm was demonstrated by pricing options with path dependencies, early exercise boundaries and exotic features. Parameter sets that would be appropriate to interest rate and equity environments were used to demonstrate the correct handling of the boundary behaviour.

JRMQ was shown to be accurate and fast when compared to traditional Monte Carlo methods. This will allow the calibration of large derivative books, as per Callegaro et al. [2015a], to be extended from only considering local volatility models to the more flexible stochastic volatility models, while retaining the efficiency of the original recursive marginal quantization algorithm.

References

H. Albrecher, P. Mayer, W. Schoutens, and J. Tistaert. The little Heston trap. *KU Leuven Section of Statistics Technical Report*, 06(05), 2006.

G. Bormetti, G. Callegaro, G. Livieri, and A. Pallavicini. A backward Monte Carlo approach to exotic option pricing. 2016. Available at SSRN 2686115.

G. Callegaro, L. Fiorin, and M. Grasselli. Pricing and calibration in local volatility models via fast quantization. Available at SSRN 2495829, 2014.

G. Callegaro, L. Fiorin, and M. Grasselli. Quantized calibration in local volatility. *Risk Magazine*, 28:62–67, 2015a.

G. Callegaro, L. Fiorin, and M. Grasselli. Pricing via quantization in stochastic volatility models. Available at SSRN 2669734, 2015b.

B. Chen, C. Oosterlee, and J. van der Weide. Efficient unbiased simulation scheme for SABR stochastic volatility model. *International Journal of Theoretical and Applied Finance*, 15 (2), 2012.

P. S. Hagan, D. Kumar, A. S. Lesniewski, and D. E. Woodward. Managing smile risk. *The Best of Wilmott*, 1:249–296, 2002.

P. S. Hagan, D. Kumar, A. S. Lesniewski, and D. E. Woodward. Universal smiles. *Wilmott*, 2016(84):40–55, 2016.

S. L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6(2):327–343, 1993.

R. Korn and S. Tang. Exact analytical solution for the normal SABR model. *Wilmott*, 2013 (66):64–69, 2013.

R. Lord, R. Koekkoek, and D. V. Dijk. A comparison of biased simulation schemes for stochastic volatility models. *Quantitative Finance*, 10(2):177–194, 2010.

T. A. McWalter, R. Rudd, J. Kienitz, and E. Platen. Recursive marginal quantization of higher-order schemes. Available at SSRN 2894753, 2017.

J. Oblój. Fine-tune your smile: Correction to Hagan et al. *arXiv:0708.0998*, 2007.

G. Pagès. Introduction to optimal vector quantization and its applications for numerics. Technical report, July 2014. URL https://hal.archives-ouvertes.fr/hal-01034196.

G. Pagès and H. Pham. Optimal quantization methods for nonlinear filtering with discrete-time observations. *Bernoulli*, 11(5):893–932, 2005.

G. Pagès and A. Sagna. Recursive marginal quantization of the Euler scheme of a diffusion process. *Applied Mathematical Finance*, 22(5):463–498, 2015.

G. Pagès and B. Wilbertz. Optimal Delaunay and Voronoi quantization methods for pricing American options. In R. Carmona, P. Hu, P. Del Moral, and N. Oudjane, editors, *Numerical Methods in Finance*, pages 171–217. Springer, 2009.

G. Pagès, H. Pham, and J. Printems. An optimal Markovian quantization algorithm for multidimensional stochastic control problems. *Stochastics and Dynamics*, 4(4):501–545, 2004.

L. Paulot. Asymptotic implied volatility at the second order with application to the SABR model. In *Large Deviations and Asymptotic Methods in Finance*, pages 37–69. Springer, 2015.

A. Sagna. Pricing of barrier options by marginal functional quantization. *Monte Carlo Methods and Applications*, 17(4):371–398, 2011.

R. Schöbel and J. Zhu. Stochastic volatility with an Ornstein–Uhlenbeck process: an extension. *European Finance Review*, 3(1):23–46, 1999.

E. M. Stein and J. C. Stein. Stock price distributions with stochastic volatility: an analytic approach. *Review of Financial Studies*, 4(4):727–752, 1991.

Appendix A: Volatility Corridor Swaps

Consider

$$\frac{1}{T}\int_0^T X_z \mathbb{I}_{\{L<S_z<H\}}\,dz = \frac{1}{T}\sum_{k=0}^{K-1}\int_{t_k}^{t_{k+1}} X_z \mathbb{I}_{\{L<S_z<H\}}\,dz$$
$$\approx \frac{1}{T}\sum_{k=0}^{K-1}\int_{t^*(S_{t_k})}^{t^*(S_{t_{k+1}})} \frac{X_{t_{k+1}}-X_{t_k}}{\Delta t}(z-t_k)+X_{t_k}\,dz, \tag{32}$$

where the volatility process X_t has been replaced by a linear interpolation on the interval $t \in [t_k, t_{k+1}]$ with

$$t^*(s) = \begin{cases} t_k & \text{if } L \le s \le H \text{ and } s = S_{t_k}, \\ t_{k+1} & \text{if } L \le s \le H \text{ and } s = S_{t_{k+1}}, \\ \frac{H-S_{t_k}}{S_{t_{k+1}}-S_{t_k}}\Delta t + t_k & \text{if } s > H, \\ \frac{L-S_{t_k}}{S_{t_{k+1}}-S_{t_k}}\Delta t + t_k & \text{if } s < L, \end{cases}$$

providing the intercepts of the line connecting S_{t_k} and $S_{t_{k+1}}$ with the corridor. This interpolation is illustrated in Figure 11 and accounts for the indicator function by constraining the integration to where the asset price is in the corridor. Explicitly computing a single term from (32) gives

$$G(t_k, t_{k+1}, X_{t_k}, S_{t_k}, X_{t_{k+1}}, S_{t_{k+1}}) :=$$
$$\frac{X_{t_{k+1}}-X_{t_k}}{2\Delta t}\left[(t^*(S_{t_{k+1}})-t_k)^2-(t^*(S_{t_k})-t_k)^2\right]+X_{t_k}\left[t^*(S_{t_{k+1}})-t^*(S_{t_k})\right].$$

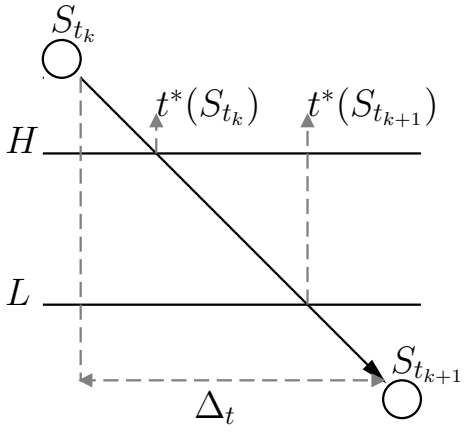

Figure 11: Linear interpolation of the asset price provides the bounds for the integration.

The value of a volatility corridor swap can now be computed as the expectation under the risk-neutral measure approximated using the quantization grids for X and S,

$$\mathbb{E}\left[\frac{1}{T}\int_0^T X_z \mathbb{I}_{\{L<S_z<H\}}\, dz\right] \approx \frac{1}{T}\sum_{k=0}^{K-1}\sum_{i=1}^{N^x}\sum_{j=1}^{N^x}\sum_{u=1}^{N^y}\sum_{v=1}^{N^y} G(t_k, t_{k+1}, x_k^i, s_k^u, x_{k+1}^j, s_{k+1}^v) \\ \times \mathbb{P}(\widehat{X}_k = x_k^i, \widehat{S}_k = s_k^u, \widehat{X}_{k+1} = x_{k+1}^j, \widehat{S}_{k+1} = s_{k+1}^v),$$

with the probability

$$\mathbb{P}(\widehat{X}_k = x_k^i, \widehat{S}_k = s_k^u, \widehat{X}_{k+1} = x_{k+1}^j, \widehat{S}_{k+1} = s_{k+1}^v) = \\ \mathbb{P}(\widehat{S}_{k+1} = s_{k+1}^v | \widehat{X}_k = x_k^i, \widehat{S}_k = s_k^u, \widehat{X}_{k+1} = x_{k+1}^j)\mathbb{P}(\widehat{X}_k = x_k^i, \widehat{S}_k = s_k^u, \widehat{X}_{k+1} = x_{k+1}^j),$$

computed as part of the matrix formulation in Section 5.