Eliciting a Smile: Numerical Methods for Option Pricing with Matrix Affine Jump Diffusions

Peter H. Gruber*

First version: Jan 23, 2015
This version: May 27, 2015

Abstract

I present a framework for the efficient and stable estimation of matrix affine jump diffusion (MAJD) volatility models using transform methods. Due to its matrix nature, there are several computational challenges for evaluating the Laplace transform of MAJD models: (i) it contains a multivalued complex logarithm, causing a potentially large bias in option prices, (ii) it is computationally more costly by two orders of magnitude compared to similar scalar models, and (iii) it contains a numerically unstable matrix inversion. To solve these problems, I formulate the matrix rotation count algorithm for the complex logarithm and I propose a volatility-independent approximation of the integration limits for the Cosine Fourier inversion. The latter forms the basis of a fast estimation scheme via a separation of the state- and parameter-dependent parts of the Laplace transform. The results can also be applied to scalar multivariate affine models, as they are nested in the MAJD class.

* peter.gruber@usi.ch

1 Introduction

The affine jump diffusion (MAJD) process, introduced in Leippold and Trojani (2008), represents a promising class of multivariate stochastic volatility models. It allows for mutually-exciting risk factors, unspanned skewness effects and a realistic modeling of variance risk, all without compromising on tractability. So far, the literature has focused on the theoretical properties of these processes (Muhle-Karb, Pfaffel and Stelzer (2010), Mayerhofer (2014)), on applications in option pricing (da Fonseca, Grasselli and Tebaldi (2008), Gruber, Tebaldi and Trojani (2010)), and on approximations of the resulting volatility smile (Benabid, Bensusan and El Karoui (2009)). Numerical aspects of estimation and evaluation of these models in the context of transform methods have not yet been considered. This paper aims at filling this gap.

In doing so, I build on a vast literature on the numerical aspects of option pricing. The use of Fourier integrals for option pricing has been pioneered by Chen and Scott (1992), Heston (1993) and Bates (1996). Carr and Madan (1999) introduce the Fast Fourier Transform (FFT) method to finance, which allows for the pricing of an entire option chain with a fixed number (typically 2^12) of evaluations of the Laplace transform. Fang and Oosterlee (2008) improve on the work of Carr and Madan by choosing a new set of basis functions and by introducing an efficient truncation of the probability density function. Their Fourier-Cosine method (COS) is typically 20 times faster than the FFT, without compromising precision.

Numerical problems related to the implementation of transform methods, especially with a complex logarithm, were first emphasized by Schöbel and Zhu (1999). Kahl and Jäckel (2005) take this topic up and formulate the rotation count algorithm for scalar stochastic volatility models. Lord and Kahl (2010) build on their insights, but advocate a re-formulation of the Laplace transform to avoid rather than remedy the problem of the complex logarithm. Cont and Hamida (2005) use evolutionary algorithms to estimate a low-dimensional volatility model.

The matrix nature of the MAJD class introduces several new challenges in the fields of computational efficiency, numerical stability, parameter identification and optimization:

First, matrix models are by definition high-dimensional models. In particular, the large number of state components increases the computational complexity of jointly estimating model parameters and the latent state. The smallest MAJD model, which is based on a symmetric 2×2 matrix and serves as an illustration throughout this paper, has three state components and 16 risk-neutral parameters.

Second, the Laplace transform has several matrix components, some of which double the dimensionality of the problem. At its core, it contains a complex matrix

exponential of double dimension, i.e. of a 4×4 matrix in the smallest case. Matrix exponentials have a computational cost which is roughly two orders of magnitude larger than scalar exponentials.

Third, the Laplace transform contains a complex matrix logarithm, which is a multivalued function. Using the principal branch of the complex logarithm, as implemented by default in nearly every programming language, causes potentially large biases in the option price.

Fourth, the Laplace transform contains an inverse matrix, which is numerically unstable if the matrix is close to rank deficiency. This is the case for large arguments of the Laplace transform, i. e. for long horizons and/or if the integration limits are set conservatively (i. e. very widely).

Fifth, little is ex-ante known about the role of individual state components. We know for example that the state in a one-factor volatility model basically corresponds to short term variance, which makes the square of the short term implied volatility a natural candidate as starting value in a convex optimization. The high dimension of MAJD models makes it possible that different state components pick up distinct second-order phenomena.[1] It is therefore very difficult to deduce starting values for the state from economic considerations. The same applies to parameter estimation.

Sixth, although speed is nowadays rarely an issue in estimating an option pricing model, MAJD models are an exception. On one side, the dimensionality of the problem requires an unusually high number of evaluations of the Laplace transform, on the other side, these evaluations are unusually costly. Pricing one option with standard numerical integration takes about 9 seconds. Estimating the benchmark 2×2 model on 7 years of weekly data using nonlinear least squares requires the calculation of some 1 billion option prices and would take 285 years.[2]

This paper proposes several independent speed improvements that total to a factor of 10^6, reducing the estimation time to a few hours and thus rendering the model estimation feasible. The biggest contribution to these improvements comes from the matrix rotation count algorithm, with a factor of about 70. It enables the use of the FFT method, which prices an entire chain of about 20 options in less than 3 seconds. An additional factor of almost 20 is achieved by using the COS method, which requires fewer evaluations of the Laplace transform. A further factor of 50 obtains from a novel evaluation scheme of the likelihood function, that separates state- and parameter-dependent parts of the Laplace transform. This evaluation scheme also improves the

[1] See Gruber et al. (2010) for a detailed analysis of the role of the state components in the benchmark case.

[2] See Table 4 for the specification of the reference data set, and Table 5 for the benchmark model. The reference computer system is a 2.8 GHz Intel Xeon 540 running MATLAB 2014b on one core.

scaling of the computational cost with respect to the size of the data set, especially the length of the time series. It requires a delicate choice of the integration bounds. I present an approximation that limits the error in implied volatility to 10^{-3} implied volatility percentage points (i.e. less than 0.1 implied volatility basis points). A final factor of 20 obtains from parallelization.

Beyond speed improvements, this paper identifies a numerical instability of the Laplace transform for large values of the imaginary part of its argument. This requires a precise choice of the integration bounds and truncation parameters for the FFT and COS series.

I furthermore present two useful reparametrizations of the state and parameter matrices, which facilitate the constrained optimization with respect to admissibility conditions of the stochastic process.

The rest of the paper is organized as follows. Section 2 gives a brief overview of the MAJD process and the estimation strategies under consideration. Section 3 discusses issues arising from the complex logarithm and introduces the matrix rotation count algorithm. Section 4 presents the fast evaluation scheme of the likelihood function based on the separation of state- and parameter-dependent parts of the Laplace transform and an approximation of the integration limits. Section 5 discusses the numerical instability of a matrix inverse in the Laplace transform. Section 6 discusses problems of optimization and proposes alternative matrix representations. Section 7 concludes. The appendix specifies the 2×2 matrix process which is used as an illustration throughout this paper.

2 The MAJD process for option pricing

2.1 The process

Variance states follow the affine diffusion process:[3]

$$dX_t = [\beta Q'Q + MX_t + X_t M']dt + \sqrt{X_t} dB_t Q + Q' dB_t' \sqrt{X_t} \,, \tag{1}$$

where the state matrix X_t is an $n \times n$ symmetric, positive definite matrix, the parameters M and Q are $n \times n$ matrices and B is an $n \times n$ standard Brownian motion under the risk-neutral martingale measure $\mathbb{Q}$. X is the Wishart process introduced by Bru (1991). The process is positive definite (semidefinite) if $\beta > n + 1$ ($\beta > n - 1$), such

[3] Leippold and Trojani allow for jumps in variance and denote the Laplace transform of the respective jump size distribution Θ^X. Jumps in variance are omitted here, as their presence has no impact on the results presented in this paper.

that the diffusive variance $tr(X_t)$ cannot reach (cross) the zero bound.

Remark. The symmetric state matrix X_t has $n(n+1)/2$ distinct components. The n diagonal elements X_{ii} are always positive (non-negative), if the process is positive definite (positive semidefinite). The $n(n-1)/2$ out-of diagonal elements X_{ij} with $i \neq j$ can take positive or negative values within the bounds of the (semi)definiteness property. When M or Q are not diagonal matrices, all state components are dynamically interconnected. With both M and Q diagonal, the process collapses to n independent variance processes of the Heston-type. This model therefore naturally nests multivariate stochastic volatility models with independent factors such as Heston (1993), Bates (2000) or Christoffersen, Heston and Jacobs (2009).

Given the matrix state dynamics (1), the returns process under the risk neutral probability measure $\mathbb{Q}$ is specified as

$$\frac{dS_t}{S_{t-}} = (r - q - \lambda_t \overline{k})dt + tr(\sqrt{X_t} dZ_t) + k dN_t \ , \tag{2}$$

where r and q denote interest rate and dividend yield, $tr(\cdot)$ denotes the trace of a matrix and $Z_t = B_t R + W_t \sqrt{I_n - RR'}$. Matrix W is another $n \times n$ standard Brownian motion, independent of B, and the correlation between variance and return shocks R is an $n \times n$ matrix such that $I_n - RR'$ is positive semi-definite.

Jumps in returns follow a compound Poisson process kdN_t with jump intensity $\lambda_t = \lambda_0 + tr(\Lambda X_t)$, where $\lambda_0 \geq 0$ and Λ is a positive definite $n \times n$ matrix. The distribution of the return jump size k is specified by its Laplace transform Θ^Y.

2.2 Estimation strategies and computational cost

Estimating a latent factor model such as the MAJD class usually encompasses the joint estimation of the parameters and the latent state. Before we can estimate the parameter matrices, we need to rewrite them in form of a vector θ, as all major optimization algorithms work only for vectors, see Section 6.2. We can formulate the optimization problem in a maximum likelihood context as:

$$(\widehat{\theta}, \{\widehat{X}_t\}) = \underset{\theta, \{X_t\}}{\arg\max}\, \mathcal{L}(\theta, \{X_t\}; \mathcal{O}) \tag{3}$$

where $\mathcal{L}(\cdot)$ denotes the likelihood function, $\{X_t\}$ the whole time series of X_t and $\mathcal{O}$ stands for the panel of option data. This is a high-dimensional problem, which does not lend itself well to optimization. For a time series of length T, we need to estimate

$n_\theta + T \cdot n(n+1)/2$ quantities.[4]

In order to break the dimensionality of the joint problem, I opt for a nested estimation:[5]

$$\text{Inner problem:} \qquad \{\widehat{X}_t|\theta\} = \arg\max_{\{X_t\}} \mathcal{L}(\theta, \{X_t\}; \mathcal{O}) \tag{4}$$

$$\text{Outer problem:} \qquad \widehat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta, \{\widehat{X}_t|\theta\}; \mathcal{O}) \tag{5}$$

i.e. for each parameter vector θ I find the optimal time series of the state $\widehat{X}_t$ (inner optimization) and the I optimize over θ, evaluating the likelihood function at $\widehat{X}_t|\theta$. The point estimator for the parameter vector θ can be written as

$$\widehat{\theta} = \arg\max_{\theta} \mathcal{L}\Big(\theta, \arg\max_{\{X_t\}} \mathcal{L}(\theta, \{X_t\}; \mathcal{O}); \mathcal{O}\Big) \ . \tag{6}$$

This reduces the dimension of the optimization problem, as we now have one problem of dimension n_θ and T problems of dimension $n(n+1)/2$. The form of the likelihood function (5) and the number of option prices required for one evaluation depends on the estimation strategy:

Nonlinear least squares (NLLS). This optimization approach follows Bates (2000) and Huang and Wu (2004) and minimizes the (weighted) relative pricing errors. The first (inner) step computes the optimal state conditional on the parameter vector. This is defined independently for each trading day t

$$\widehat{X}_t|\theta = \arg\min_{X_t} \sum_{i=1}^{N_t} \left[\Big(\widehat{O}_{it}(\theta, X_t) - O_{it} \Big) / F_t \right]^2 \ , \tag{7}$$

where O_{it} and $\widehat{O}_{it}(\theta; X_t)$ denote the observed and model-implied prices of option i, with $1 \leq i \leq N_t$. We can now define the vector $\mathbf{e}_t$ of relative option pricing errors at the conditionally optimal state with elements $e_{i,t} = \big(\widehat{O}_{it}(\theta, \widehat{X}_t|\theta) - O_{it}\big)/F_t$. Note that N_t, the length of vector $\mathbf{e}_t$, varies from trading day to trading day.

Our point estimate for parameter θ is given by the following pseudo Maximum

[4] In the benchmark case of the minimal 2×2 model, $n_\theta = 16$ and $n(n+1)/2 = 3$. In the benchmark data set, $T = 359$, thus the overall dimension is 1093.

[5] In the context of NLLS estimation, Huang and Wu (2004) successfully iterate between an optimization over the state given a parameter vector and an optimization over the parameters, given the (previous) state estimate. This approach does not produce stable results for the MAJD class and is furthermore not compatible with stochastic optimization algorithms. Furthermore, extended Kalman Filter (EKF) estimation is only conceivable as nested optimization.

Likelihood estimator:

$$\widehat{\theta} = \arg\max_{\theta} \; -\frac{1}{2}\sum_{t=1}^{T}\Big(\ln|\Omega_t| + \mathbf{e}_t'\,\Omega_t^{-1}\,\mathbf{e}_t\Big)\;. \tag{8}$$

where the $N_t \times N_t$ matrix Ω_t is the conditional covariance matrix of these errors which is obtained from calculating group-specific error covariances in three maturity and three moneyness groups, similar to Bates (2000).

In this optimization strategy, the outer optimization problem has a dimension of $n_\theta = 16$. The T inner optimizations of dimension $n(n+1)/2$ are independent and can therefore be parallelized. We furthermore note that θ does not change during the inner optimization step, a fact that will be used in Section 4. A drawback of the nested NLLS approach is that M_{NLLS}^{Xopt}, the number of optimization steps for (each) inner optimization, is quite large.[6] The total number of option prices that need to be calculated for one evaluation of the NLLS-likelihood function (8) is

$$M_{NLLS}^{\mathcal{L}} = T\,n_\tau\,n_k\,M_{NLLS}^{Xopt} \tag{9}$$

where n_τ denotes the average number of maturities per trading day and n_k the average number of strikes per option chain. In the benchmark example, $M_{NLLS}^{\mathcal{L}}$ is roughly 2 million.[7]

Extended Kalman Filter (EKF). This optimization approach follows, among others, Carr and Wu (2007) and Carr and Wu (2009). The inner optimization is replaced by an extended Kalman Filter, which produces $\widehat{X}_t|\theta$. Apart from obvious advantages such as consistency of the estimate with the assumed form of the process and the possibility to estimate the physical dynamics of the latent state, the EKF strategy usually requires the calculation of fewer option prices: one evaluation of the option price plus $n(n+1)/2+1$ evaluations to numerically calculate the Jacobian matrix, i. e. $M_{EKF}^{J} = 2 + n(n+1)/2$.

The point estimate for the parameter vector θ is now

$$\widehat{\theta} = \arg\max_{\theta} \; -\sum_{t=1}^{T}\mathbf{e}_t'\mathbf{e}_t\;. \tag{10}$$

where we define $\mathbf{e}_t$ analogously to the NLLS case. The total number of option prices

[6] For the 2×2 benchmark model, M_{NLLS}^{Xopt} is typically about 40.

[7] For the benchmark data set $T = 359$, $n_\tau = 6.09$ and n_k=23.

that need to be calculated for one evaluation of the likelihood function (10) is

$$M_{EKF}^{\mathcal{L}} = T\, n_\tau\, n_k\, M_{EKF}^{J} \tag{11}$$

which is ca. 200 000 for the benchmark example, i.e. ten times fewer than the NLLS approach. We note that during one evaluation of the filter, θ does not change, a fact that we will use in Section 4. A drawback of the EKF approach is that, unlike the NLLS case, the calculation of the state cannot be parallelized, as the estimate of X_t depends on $\widehat{X}_{t-1}|\theta$.

2.3 The Laplace Transform

In the canonical form, the Laplace transform of the log-return process is written as (see Leippold and Trojani (2008))[8]:

$$\Psi_{\gamma,t}(\tau) = \exp\{\gamma Y_t + Tr\left[A(\tau;\gamma)X_t\right] + B(\tau;\gamma)\} \tag{12}$$

Where the $n \times n$ matrix $A(\tau)$ and the scalar $B(\tau)$ are solutions to the following Riccati differential equations:

$$\begin{aligned}
\frac{\partial A(\tau)}{\partial \tau} &= A(\tau)M + M'A(\tau) + 2\gamma RQA(\tau) + 2A(\tau)Q'QA(\tau) + \\
&\quad + \frac{\gamma(\gamma-1)}{2} Id_n + \Lambda\Big[\Theta^Y(\gamma) - 1 + \gamma\big(\Theta^Y(1) - 1\big)\Big] \qquad (13)\\
\frac{\partial B(\tau)}{\partial \tau} &= tr[\Omega\Omega' A(\tau)] + (\gamma-1)r + \lambda_0\Big[\Theta^Y(\gamma) - 1 + \gamma\big(\Theta^Y(1) - 1\big)\Big] \qquad (14)
\end{aligned}$$

where $A(0) = B(0) = 0$. Note: $tr[\Omega\Omega' A] = tr[A\Omega\Omega'] = tr[AQ'Q\beta]$.

[8] We follow the notation of Leippold and Trojani (2008), where X denotes the variance factors and Y the returns. Jumps in the volatility factors are omitted, as they do not impact the results in this paper.

Equations (13) and (14) have the following solutions:

$$
\begin{aligned}
A(\tau) &= C_{22}^{-1}(\tau)C_{21}(\tau) && (15)\\
B(\tau) &= \int_0^\tau tr[\Omega\Omega' A(s)] + \gamma r + \lambda_0\Big[\Theta^Y(\gamma) - 1 + \gamma\big(\Theta^Y(1) - 1\big)\Big]ds \\
&= \tau\Big\{(\gamma-1)r + \lambda_0\Big[\Theta^Y(\gamma) - 1 + \gamma\big(\Theta^Y(1) - 1\big)\Big]\Big\} \\
&\quad + \int_0^\tau tr[\Omega\Omega' A(s)]ds && (16)\\
&= \tau\Big\{(\gamma-1)r + \lambda_0\big[\Theta^Y(\gamma) - 1 - \gamma\Theta^Y(1)\big]\Big\} \\
&\quad -\frac{\beta}{2}tr[\log(C_{22}(\tau)) + \tau(M' + \gamma R'Q)] && (17)
\end{aligned}
$$

with the $2n \times 2n$ matrix

$$
\begin{pmatrix} C_{11}(\tau)\ C_{12}(\tau) \\ C_{21}(\tau)\ C_{22}(\tau) \end{pmatrix} = \exp\left[\tau \begin{pmatrix} M + \gamma Q'R & -2Q'Q \\ C_0(\gamma) & -(M' + \gamma R'Q) \end{pmatrix}\right] := \exp(E) \qquad (18)
$$

and the $n \times n$ matrix C_0:

$$
C_0(\gamma) = \frac{\gamma(\gamma-1)}{2}Id_n + \Lambda\Big[\Theta^Y(\gamma) - 1 + \gamma\big(\Theta^Y(1) - 1\big)\Big]\,. \qquad (19)
$$

The univariate Laplace transform of the return jump size distribution is for the lognormal case

$$
\Theta^Y_{LN}(\gamma) = (1+\overline{k})^\gamma \exp\left(\gamma(\gamma-1)\frac{\delta^2}{2}\right) \qquad (20)
$$

and for the double exponential case

$$
\Theta^Y_{DX}(\gamma) = \frac{\lambda^+\lambda^-}{\lambda^+\lambda^- + \gamma(\lambda^+ - \lambda^-) - \gamma^2}. \qquad (21)
$$

Option prices obtain via the transform methods, i.e. the FFT of Carr and Madan (1999), or the COS method of Fang and Oosterlee (2008) or the quadrature method of Attari (2004).

3 The Matrix Rotation Count Algorithm

3.1 Multivalued complex logarithms

Ambiguity is a rare phenomenon in computational finance, but one such example is the complex logarithm embedded in the Laplace transform of most affine volatility models,

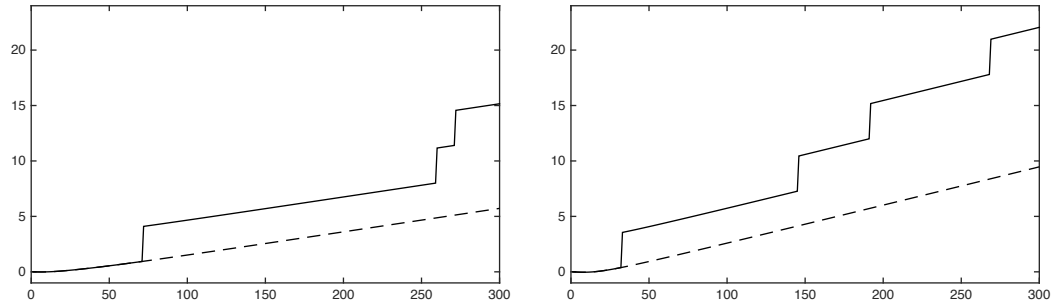

Figure 1: Imaginary part of $B(\tau, \gamma_k)$ in the benchmark model as a function of k in the COS inversion. Full lines depict the result for a direct evaluation of (17), dashed lines the result when using (16) or applying the rotation count algorithm. Left panel: $\tau =$ 9 months, right panel: $\tau = 2$ years. The values for γ_k are given in Table 1.

such as the models of Heston (1993), Bates (2000), Duffie, Pan and Singleton (2000) or Leippold and Trojani (2008). To illustrate the problem, consider a complex scalar $z = a+ib$ with exponential $e^z = e^a \cdot e^{ib}$. The complex exponential $e^{ib} = \cos(b)+i\sin(b)$ is a periodic function of b, therefore for any integer k

$$e^{ib} = e^{i(b+k\,2\pi)} \ .$$

If we observe only e^z, we cannot know the value of k in $z = a + i(b + k\,2\pi)$. The inverse function $\log(e^z)$ is multivalued and it is a convention to limit b to $(-\pi, \pi]$. This interval is called the "principal branch". If the imaginary part of a continuous function b passes $(2k + 1)\pi$, the logarithm "rotates", resulting in a discontinuity of the principal branch of the imaginary part of the logarithm. As Lord and Kahl (2010) observe, this may be "*leading to completely wrong option prices if options are priced by Fourier inversion.*" Figure 4 provides a quantitative illustration for magnitude of the problem.

The characteristic function in formulation (17) involves a matrix logarithm in $B(\gamma, \tau)$, and indeed we observe a discontinuity of B as function of γ, as illustrated in Figure 1. All discontinuities of B are jumps by π. They seem to follow no distinct

τ	γ_0	γ_1	γ_{199}	τ	γ_0	γ_1	γ_{199}
1 month	0	1.06	212.4	12 months	0	0.35	69.9
3 months	0	0.70	139.8	24 months	0	0.25	49.4
9 months	0	0.41	80.7	60 months	0	0.16	31.3

Table 1: Typical values for $\gamma_k(\tau)$

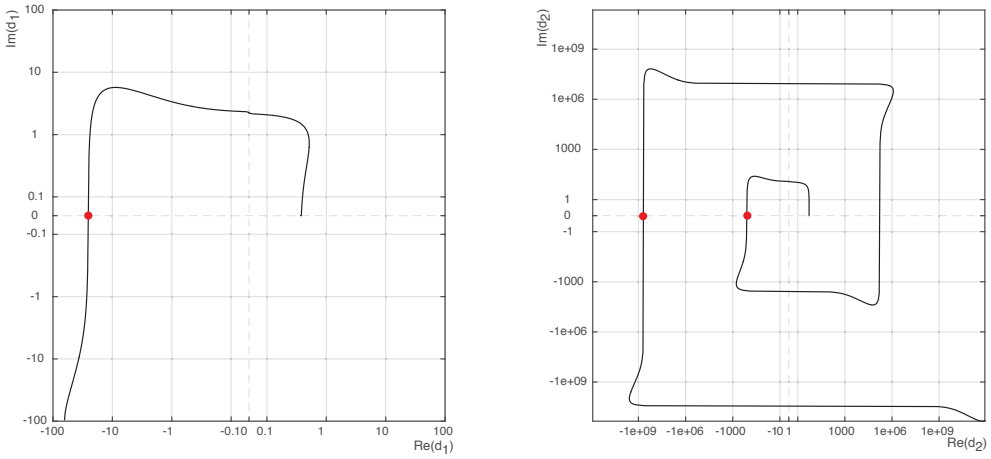

Figure 2: Complex eigenvalues of the matrix C_{22} of the benchmark model as a function of γ_k in the COS method for $\tau = 1yr$. Red dots denote transitions from the second to the third quadrant, where the discontinuity in the principal branch of the complex logarithm occurs. Left: first eigenvalue, right: second eigenvalue. The dashed lines illustrate the fact that each plot is composed of four log-log plots, omitting very small positive and negative values along both axes.

pattern except that there are more jumps if the duration τ is longer. If we apply the (computationally much slower) formulation (16), no jumps occur.

In order to better understand the phenomenon, we rewrite the matrix logarithm in (17) in terms of a PDP decomposition.

$$\begin{aligned} PDP^{-1} &= C_{22} \\ \log(C_{22}) &= P \log DP^{-1} \end{aligned} \tag{22}$$

Note that the matrix D is a diagonal matrix containing the eigenvalues of C_{22}, which means that the matrix logarithm of D is simply the diagonal matrix of the logarithms of the diagonal elements: $(\log(D)_{ii} = \log(D_{ii})$. Figure 2 illustrates the two eigenvalues of the 2×2 benchmark model each of these eigenvalues as a function of γ_k. Both eigenvalues start in the first quadrant and rotate counter-clockwise. When an eigenvalue passes from the second to the third quadrant, a "rotation" occurs and the imaginary part of the logarithm jumps. The fact that the two eigenvalues rotate independently is the reason why we did not see a regular pattern in Figure 1.

3.2 The algorithm

The only practical solution is to count the number of rotations, preferably separately for each eigenvalue, and to correct for it. This is what the following algorithm does:

Algorithm 1 (Matrix Rotation Count Algorithm)**.** The following algorithm solves the problem of the discontinuous logarithm of C_{22} in (17) and produces a continuous B-function. This algorithm must be run separately for every τ.

1. Initialize the number of rotations at $r_{1,i} := 0$ with $1 \leq i \leq n$.
2. For every γ_k, with $1 \leq k \leq N$ do
    3. Calculate $C_{22}(\tau, \gamma_k)$
    4. Perform a PDP decomposition $P_k D_k P_k^{-1} = C_{22}(\tau, \gamma_k)$, where D_k is a diagonal matrix containing the eigenvalues of C_{22}.
    5. For every eigenvalue $D_{k,ii}$ with $1 \leq i \leq n$ do
        6. Calculate the complex logarithm $d_{k,i} = \log(D_{k,ii})$.
        7. Produce the sawtooth-like function $m_{k,i} = \Im(d_{k,i}) \mod \pi$.
        8. Verify whether a rotation has occurred. Increase $r_{k,i}$ by 1 for every positive rotation, i.e. when $m_{k,i} - m_{k-1,i} > \pi/2$ and decreased by 1 for every negative rotation, i.e. when $m_{k,i} - m_{k-1,i} < -\pi/2$.
        9. The correct branch of the imaginary part of $\log(D_{k,ii})$ obtains as $\Im(d_{k,i}) := m_{k,i} + \pi \cdot r_{k,i}$. The real part $\Re(d_{k,i})$ is not changed. We obtain the diagonal matrix $D_{k,ii}^{\log} = \Re(d_{k,i}) + i\,(m_{k,i} + \pi r_{k,i}) = \log(D_k)$.

    enddo
10. Use $\log(C_{22}(\tau, \gamma_k)) = P D_k^{log} P^{-1}$ to calculate $B(\tau, \gamma_k)$ in (17).

enddo

where $k = 1 \ldots N$ counts the arguments of the Laplace transform γ_k and $i = 1 \ldots n$ counts the eigenvalues of C_{22}. The symbols $\Re(x)$ and $\Im(x)$ denote the real and imaginary parts of x. See Figure 3 for an illustration.

Remark 1. The rotations of the individual eigenvalues as illustrated in Figure 2 are counted in step 8 by calcualting $m_{k,i} - m_{k-1,i}$. The algorithm therefore requires the sequential evaluation of the characteristic function for increasing values of γ, starting form $\gamma_1 = 0$. Thus it is only suitable for inversion methods that sequentially evaluate the characteristic function on a uniform grid like FFT, COS or Attari's method and a

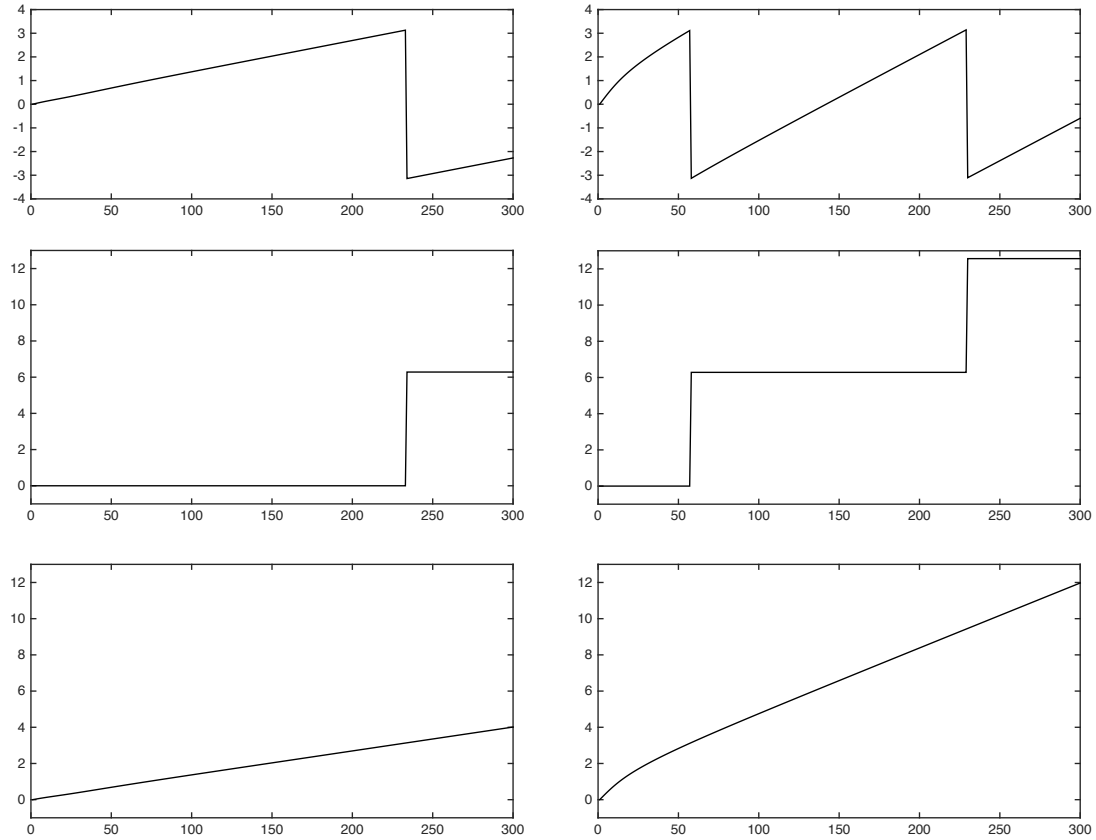

Figure 3: Illustration of the Matrix Rotation Count (MRC) algorithm. Top: sawtooth-function from step 6. Middle: step function from step 7. Bottom: continuous eigenvalues from step 8. Left: first eigenvalue, right: second eigenvalue. Illustration for the benchmark model and $\tau = 1yr$.

straight forward application to adaptive algorithms such as Gauss-Lobatto integration is not possible.

Remark 2. Some authors, e.g. Benabid et al. (2009) reformulate (17) using the identity $\log(\det(C_{22})) = tr(\log(C_{22}))$. In this formulation, knowledge of the individual eigenvalues $D_{k,ii}$ is lost, which makes it more difficult to count the individual rotations.

Remark 3. If the gradient of the imaginary part of the eigenvalues of $C_{22}(\gamma)$ is too steep, the algorithm will fail, as it would identify a rotation at every k. This can be avoided by choosing a more sophisticated identification of the rotations.

Algorithm 2 (Modified Matrix Rotation Count Algorithm)**.** This algorithm can be used if the $\Im(C_{22}(\gamma_k)) - \Im(C_{22}(\gamma_{k-1}))$ is typically larger than π.

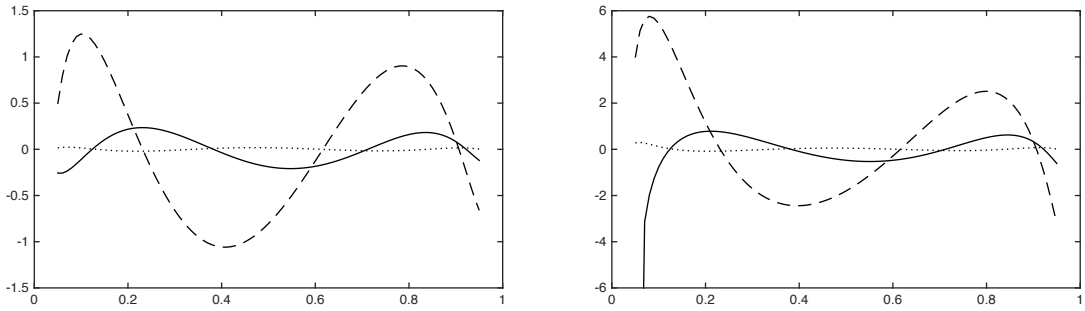

Figure 4: Typical corrections generated by the Matrix Rotation Count algorithm for call options in COS-inversions, as a function of the Black-Scholes delta. The values are shown here are for calls, for puts simply set $\Delta_{put} = 1 - \Delta_{call}$. Left panel: dollar pricing correction in percent of the price of the underlying. Right panel: implied volatility correction in volatility percentage points. Dotted lines denote a maturity of 0.75 years, full lines 1 year and dashed lines 1.25 years.

1–7. Perform steps $1 - 7$ like in algorithm 1.

8. Verify whether a rotation has occurred. The value of $r_{k,i}$ is increased (decreased) by 1 for every positive (negative) rotation, i.e. when $(m_{k,i} - m_{k-1,i}) - (m_{k-1,i} - m_{k-2,i}) > (<)\pi/2$ and $(m_{k,i+1} - m_{k,i}) - (m_{k,i} - m_{k1,i}) < (>)\pi/2$.

9–10. Perform steps $9 - 10$ like in algorithm 1.

The actual counting of rotations in this algorithm cannot be avoided. Lord and Kahl (2010) propose an algorithm for the scalar case which allows for the direct calculation of the number of rotations for γ_k without knowledge of γ_{k-1}. In the matrix case, this is not possible, as the following example for the 2×2 case shows. We start from (18) which states $C = \exp(E(\tau, \gamma))$. We can still calculate $PDP^{-1} = \tau E(\gamma)$ and anticipate any rotation in $C = P \exp(D) P^{-1}$, but the argument of the matrix logarithm is now

$$C_{22} = \begin{pmatrix} \sum_i \exp(D_{ii}) \bar{P}_{i3} P_{3i} & \sum_i \exp(D_{ii}) \bar{P}_{i4} P_{3i} \\ \sum_i \exp(D_{ii}) \bar{P}_{i3} P_{4i} & \sum_i \exp(D_{ii}) \bar{P}_{i4} P_{4i} \end{pmatrix}$$

with $\bar{P} = P^{-1}$. The elements of C_{22} are now weighted sums of the four eigenvalues of $E(\tau, \gamma)$. Thus the eigenvalues of C_{22} may pass a 2π-threshold even when no eigenvalue of $E(\tau, \gamma)$ does and the number of rotations for a single value of γ cannot be known.

3.3 Impact of the Matrix Rotation Count Algorithm

Choosing the principal branch of the complex matrix logarithm causes a bias in the option price which is potentially large. For a horizon of one year, the maximal dollar error is 0.3% of the price of the underlying, which makes it already impossible to calculate an implied volatility for a call with $\Delta = 0.05$ (viz. a put with $\Delta = -0.95$).

A numerical analysis reveals that in general, the bias is larger for longer maturities, for higher volatilities and for extreme structures of the state matrix, with the largest error obtained for singular state matrices.[9] The error is present in all Fourier inversion methods. Figure 4 provides an illustration of the magnitude of the correction for the COS method as function of the Black-Scholes delta. The sine-shaped form of the error derives from the central term in the COS inversion formula (36), which reads $Re\left\{\phi_{Levy}\left(\frac{k\pi}{b-a}\right)\exp\left(ik\pi\frac{y-a}{b-a}\right)\right\}$. The bias in $B(\tau,\gamma)$ causes a shift of the imaginary part of $\phi_{Levy}(\cdot)$, which is multiplied by the periodic function $\exp(ik\pi\frac{y-a}{b-a}) = \cos(ik\pi\frac{y-a}{b-a}) + i\sin(ik\pi\frac{y-a}{b-a})$.

After the application of the rotation count algorithm to the analytical expression (17), the results for $B(\tau,\gamma)$ are identical to the ones obtained through numerical integration in (16) up to precision of the numerical integration.

4 A Fast Evaluation Scheme for the Likelihood Function in the COS Inversion

4.1 Execution Speed of the Likelihood Function

With modern computer hardware, speed is normally not the issue in estimating option pricing models. Estimating the MAJD model is an exception. The dimension of the model necessitates a nested likelihood function. Its complexity requires the use stochastic optimization, with at least 10 000 evaluations[10] of the likelihood function (6).Furthermore, the matrix exponentials and logarithms in its characteristic function are computationally more costly by a factor of 30 to 40 compared to their scalar counterparts.

After the 70-fold speed gain afforded by the Matrix Rotation Count Algorithm and

[9] In the benchmark example, the state matrix is close to singular in 3.5% of the sample, with $\frac{X_{11}}{X_{22}}$ and $\frac{|X_{12}|}{X_{22}}$ both smaller than 0.02. As an extreme example, the filtered state for March 7, 2007, is $X = 0.0158\begin{pmatrix} 1.7\times10^{-6} & -1.3\times10^{-3} \\ -1.3\times10^{-3} & 1 \end{pmatrix}$.

[10] For example, a minimal optimization setup using differential evolution has a population size of 100 and an iteration count of 100 generations.

		COS		FFT	
		time	fraction	time	fraction
(1)	Parameter-dependent part of Ψ	144.1 ms	98.8%	2854 ms	99.4%
	Evaluation of $\exp(E)$ in (18)	77.9 ms	53.0%	1560 ms	54.3%
	Matrix Rotation Count Algorithm	46.1 ms	31.4%	860 ms	30.0%
(2)	State-dependent part of Ψ	0.8 ms	0.5%	5 ms	0.2%
(3)	Fourier inversion	1.1 ms	0.7%	11 ms	0.4%

Table 2: Breakdown of the execution time[11] for the pricing of one option chain.

the 20-fold speed gain from the use of the COS method of Fang and Oosterlee (2008), pricing one option chain takes 0.147 s in the benchmark case. This seems fast, but one evaluation of the likelihood function still takes 3.5 hrs (0.4 hrs) in the NLLS (EKF) scheme. Estimating the model still takes almost 4 (0.4) years.

In order to identify additional speed improvements, I profile the pricing of one option chain. There are three major program blocks: (1) Evaluation of the parameter-dependent part of the Laplace transform, i.e. of $A(\tau)$ and $B(\tau)$ in (15) viz. (17), including evaluation of the matrix exponential in (18) and the application of the Matrix Rotation Count Algorithm. (2) Evaluation of the state-dependent part of the Laplace transform (12) and (3) the Fourier Inversion for the option prices. The results for the 2×2 benchmark case are given in Table 2. About 99% of the total computational cost is caused by the parameter-dependent part of the Laplace transform. The evaluation of the matrix exponential (18) alone takes half of the execution time. It is therefore evident that any further improvement requires a reduction of the number of evaluations of $A(\tau)$ and $B(\tau)$.

Reconsider the Levy-part of the Laplace transform (35):

$$\Psi_{Levy}(\gamma; \tau, X_t) = \exp\left\{Tr\left[A(\tau, \gamma)X_t\right] + B(\tau, \gamma)\right\}$$

Only expression X_t is time-varying. The costly expressions $A(\tau, \gamma)$ and $B(\tau, \gamma)$ depend on τ and γ only. We can therefore envisage an evaluation scheme of the likelihood function where we re-use any value of $A(\tau, \gamma), B(\tau, \gamma)$ that has already been calculated. The cost of such a scheme depends on the number of different combinations of τ and γ.

[11] Omitting program overhead. See the appendix for the specification of the reference data set, the benchmark model and the reference computer system. One option chain contains $n_k = 23$ strikes.

4.2 The Evaluation Scheme

The first step towards a fast evaluation scheme is to reduce the number of different durations. Assuming a maximum duration of 1 year, this number can be trimmed to just 52 if we use weekly data sampled on a fixed day of the week, e.g. Wednesday.[12] The shortest duration will be 3 days (Wednesday until expiry on Saturday), the next one 10 days and so on. Two limitations have to be accepted: if there is no data for a given Wednesday, a whole trading week has to be omitted,[13] as replacing missing Wednesdays by the preceding Tuesdays potentially doubles the number of different durations in the sample. Quarterly options also have to be discarded, as they usually do not expire on a Saturday.

The second step is more delicate: in the COS expansion, γ_k depends on the integration limits $[a, b]$, which in turn depend on the second and fourth cumulants of the risk neutral distribution, see (37) and (38):

$$\gamma_k = \frac{k}{b_t - a_t} \quad \text{with} \quad -a_t = b_t = L\sqrt{\kappa_{2,t}(\theta, \tau, X_t) + \sqrt{\kappa_{4,t}(\theta, \tau, X_t)}}$$

Thus $\underline{\gamma} = \underline{\gamma}(\theta, \tau, X_t)$, which makes it usually impossible to re-use a previously calculated $A(\cdot), B(\cdot)$. We have to find a "one size fits all" approximation for a and b as a function of τ alone. We start from the observation that, for sufficiently long durations, κ_2 can be approximated by the variance level V_t times the duration:

$$\kappa_{2,t} \approx \tau\, V_t$$

Next, define $\overline{\kappa_2} = \tau\, \overline{V}$, with $\overline{V}$ the sample average of the short term, at the money variance. (In the benchmark example, $\overline{V} = 0.2$.) To accommodate the kurtosis created by jumps at short horizons, we correct τ for durations below 2 months: $\tau^c = \tau + 0.3(2/12 - \tau)$. We can now approximate the integration limits as function of τ exclusively

$$-a(\tau) = b(\tau) = L \cdot \sqrt{\overline{\kappa_2}} = L \cdot \sqrt{\tau^c\, \overline{V}} \tag{23}$$

This gives rise to the following algorithm:

Algorithm 3 (Fast Evaluation Scheme of the Likelihood Function)**.** Using approximation (23), the likelihood function (8) can be evaluated for a given parameter set θ

[12] Further reducing the frequency to monthly data reduces the number of possible durations only to 26, due to the fact that some months have four and some have five Wednesdays.

[13] For the S&P 500, there are four such occasions in the seven-year period 1996-2002: christmas 1996 and 2002, 9/11/2001 and 7/4/2001.

and time series $\{X_t\}$ of the state.

1. Make a list of all τ_i in the dataset $\mathcal{O}$.
2. For every τ_i, with $1 \leq i \leq n_\tau$ do
   3. Calculate vector $\underline{\gamma_i}(\tau_i) = \frac{k}{b(\tau_i)-a(\tau_i)}$ based on (23)
   4. Calculate $A(\tau_i, \underline{\gamma_i}), B(\tau_i, \underline{\gamma_i})$ using Algorithm 1, save in list L_1

   enddo
4. For every trading day t with $1 \leq t \leq T$ do
   5. For every duration τ_j on day t with $1 \leq j \leq n_{\tau,t}$ do
      6. Retrieve elements $A(\tau_j), B(\tau_j)$ from list L_1
      7. Calculate $\Psi_{Levy} = \exp(tr(A(\tau_j)X_t + B(\tau_j))$
      8. Perform the Fourier inversion

      enddo

   enddo

Remark 1. To accommodate the NLLS or EKF estimation strategy, simply augment steps $5. - 8.$ As the parameter-dependent part of the Laplace transform is evaluated in steps $1. - 3.$, the additional cost of the nested optimization is negligible. In the case of the NLLS scheme, the objective function of the inner optimization obtains from evaluating steps $5. - 8.$ The state $\widehat{X}_t$ is estimated from an optimization over this objective function, separately day-by-day. In the case of the Kalman filter, the Jacobian matrix is calculated via numeric differentiation. For this, steps $7. - 8.$ are repeated with $X_t + hE_j$, where E_j is a matrix that is one at the location of state component j and zero otherwise.

Remark 2. Table 3 presents a detailed timing analysis of the speedup enabled by Algorithm 3. Beyond a vast cut in execution time, the scaling behavior of the likelihood function with respect to the dimension of the process n is improved. For the Kalman Filter approach, the number of evaluations of the parameter-dependent part of the Laplace transform is reduced from $M^{\mathcal{L}}_{EKF} = T\, n_\tau\ (2 + n(n+1)/2)$ to 52. Already in the 2×2 benchmark case, this is a reduction by a factor of 168. The improvement for the NLLS approach is even larger, as the cost for the nested (inner) optimization is now greatly reduced. As a consequence, the difference in computational cost between NLLS and Kalman Filter shrinks from 10:1 to 2:1.

	Quadrature	FFT	COS
Pricing one option chain	207 s	2.88 s	0.147 s
One evaluation of the likelihood function			
NLLS standard scheme	4954 hrs	69.5 hrs	3.5 hrs
NLLS advanced scheme	n/a	562 s	138 s
EKF standard scheme	495 hrs	7 hrs	0.4 hrs
EKF advanced scheme	n/a	191 s	26 s
Marginal cost for adding ...			
One maturity	207 s	2.87 s	0.146 s
One state $\times$ maturity	207 s	4.76 ms	0.71 ms
One strike	9 s	1.25 μs	35 μs

Table 3: Execution time for one evaluation of the likelihood function and marginal computational costs for additional observations. FFT and COS methods use formulation (17) for $B(\tau)$ and apply the matrix rotation count algorithm.

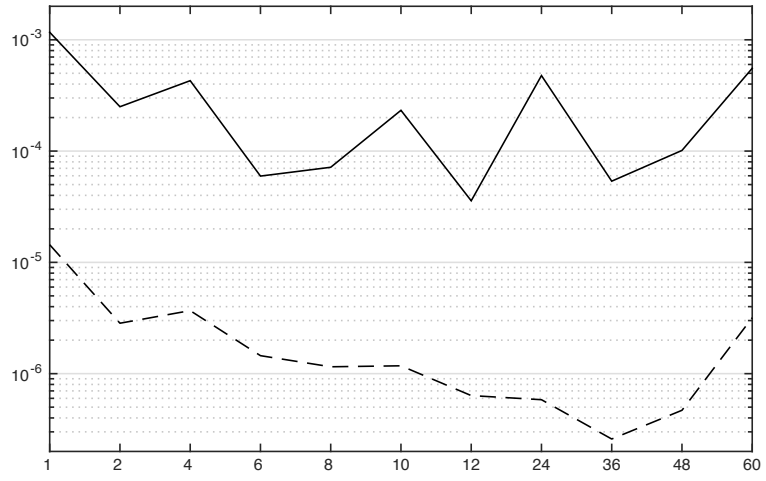

Figure 5: Error analysis of the fast evaluation scheme for the benchmark 2×2 model. The black (dashed) line depict the maximum (average) absolute implied volatility error of the fast evaluation scheme for the COS inversion, in terms of volatility percentage points. Volatility ranges from 10% to 63%, strikes are in the range $0.05 \leq |\Delta| \leq 0.95$, and all admissible state structures are scanned using the polar coordinate representation (29).

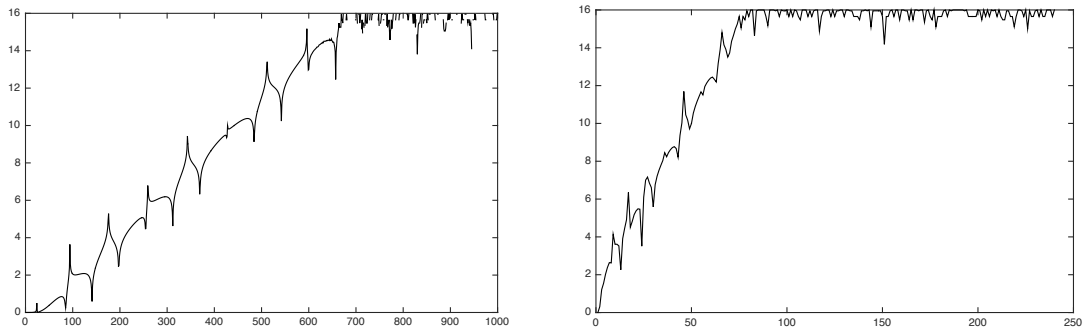

Figure 6: Loss of precision in calculating $\log C_{22}(\gamma)$ for the COS inversion. The graph depicts the number of significant digits (out of 16) lost when calculating $\log C_{22}$ as a function of k in the COS inversion. Left: $\tau = 1yr$, integration limits (23). Right: $\tau = 5yr$, sub-optimal integration limits $a(\tau), b(\tau)$ with $\tau = 1\,yr$.

4.3 Error analysis

Fixed integration limits introduce two types of error: An analytical error from the truncation of tails of the risk-neutral distribution at high volatilities and a numerical error as too wide integration limits are selected for low volatilities. To assess the quality of the approximation, I perform a detailed study of the pricing behavior for the benchmark model. Using the polar coordinate representation (29), I create state matrices with variances between 0.01 and 0.4 (volatilities from 10% to 63%) and all admissible structures.[14] I test the pricing for strikes in the range $0.05 \leq |\Delta| \leq 0.95$ (puts and calls) and durations from 1 month to 5 years.

The average and the maximal absolute approximation error in terms of implied volatility as a function of τ is quantified in Figure 5. It is always below 10^{-3} volatility percentage points, i.e. below one tenth of a volatility basis point, and averages below one-thousandth of a volatility basis point. Given that the best model in Gruber et al. (2010) has an *rms* model error of 69 volatility basis points, the numerical approximation error is negligible.

5 Loss of precision in $tr(\log(C_{22}))$

A numerical instability in the calculation of $tr(\log(C_{22}))$ arises, if the Laplace transform is evaluated for unsuitable, i.e. usually too large, values of γ. The problem is best analyzed for the 2×2 case using the identity $\log(\det(C_{22})) = tr(\log(C_{22}))$, though the

[14] All combinations $V_t = (0.01, 0.02, 0.05, 0.1, 0.2, 0.4)$, $\xi = (0, 0.25, 0.5)$ and $\alpha_1 = (0, \pi/8, \ldots \pi)$

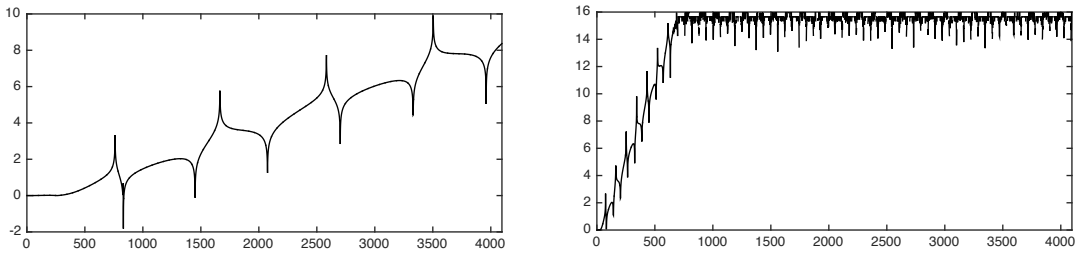

Figure 7: Loss of precision in calculating $\log C_{22}(\gamma)$ for the FFT inversion. The graph depicts the number of significant digits lost (out of 16) when calculating $\log C_{22}$ as a function of k in the FFT inversion for $\tau = 1\,yr$. Left: upper integration limit $a = 100$. Right: upper integration limit $a = 1024$, the original value of Carr and Madan (1999).

effect is the same when evaluating similar expressions. The determinant of C_{22}

$$\det(C_{22}) = C_{22}^{11}C_{22}^{22} - C_{22}^{12}C_{22}^{21} \tag{24}$$

contains a subtraction. For large values of γ, the two terms in (24) have very similar values, which makes the following theorem relevant:

Theorem 1 (Loss of precision theorem)**.** *In a subtraction $x - y$ with $0 < y < x$, a number of*

$$q \approx -log_{10}(1 - \frac{y}{x})$$

significant digits are lost.

Therefore, too large values of γ must be avoided in the evaluation of the Laplace transform. Figure 6 shows the number of significant digits (out of 16 for double precision numerics), for the COS inversion. The left panel is evaluated at a duration of 1 year, using the optimal integration limits (23). Up to six digits are lost for $N < 250$, which affords a sufficient precision for option prices.[15] However, we observe the phenomenon that adding more terms to the Fourier expansion can lead to worse results: for $N > 600$, some elements of the COS expansion loose all significant digits, resulting in a logarithm of zero and option prices that cannot be calculated. Theorem 1 also highlights the importance of the correct choice of integration limits. If we perform the COS inversion for a duration of 5 years using the integration limits (23) for $\tau = 1\,yr$, the argument γ of the Laplace transform becomes too large (see Table 1), and all significant digits are lost for $N > 80$, see right panel of Figure 6.

[15]Numerical experiments show that a similar number of digits is lost for durations from 1 month to 5 years, when using the optimal integration limits.

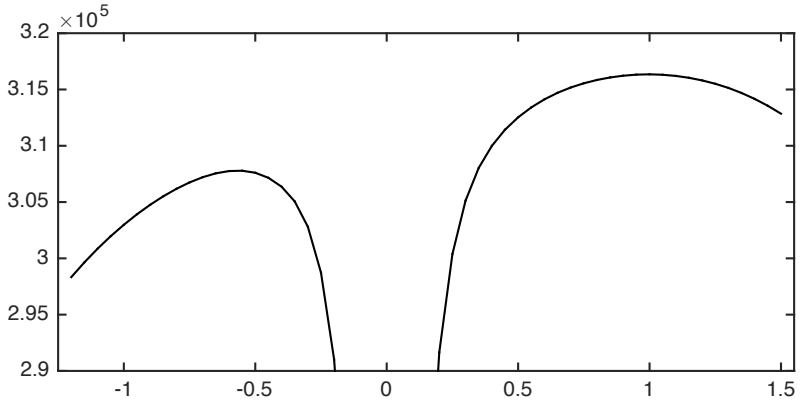

Figure 8: Example for the non-convexity of the objective function: Likelihood of the benchmark model as function of Q_{11}, expressed in multiples of the estimated value $\widehat{Q}_{11}$. All other parameters are set to their point estimates.

Worse results obtain for the FFT method, where the upper limit of the integration $a = N\eta$ (in the original notation of Carr and Madan (1999)) needs to be comparatively high in order to counter the effects of the oscillating summands. Carr and Madan (1999) propose $N = 4096$ and $\eta = 0.25$, i.e. $a = 1024$. Figure 7 shows that for $\tau = 1\,yr$, an upper limit of 1024 leads to a loss of all significant digits for $N > 600$. Numerical experiments show that a reasonable choice of a for the FFT is $100/\tau$leading to a loss of up to 8 digits, which is acceptable.

6 Numerical problems in the Model Estimation

6.1 Stochastic Optimization

The estimation of the MAJD model is formulated in (6) as a nested optimization problem:

$$\widehat{\theta} = \arg\max_{\theta} \mathcal{L}\Big(\theta, \arg\max_{\{X_t\}} \mathcal{L}(\theta, \{X_t\}; \mathcal{O}); \mathcal{O}\Big) \ .$$

Following the convention in the optimization literature, I implement the estimator of $\hat{\theta}$ by minimizing the negative likelihood. There is no theoretical evidence that this optimization problem is convex, on the contrary. As a simple counter-example, Figure 8 depicts the likelihood of the benchmark model as a function of Q_{11}, expressed in multiples of the estimated value $\widehat{Q}_{11}$. As expected, there is a maximum at 1, and the likelihood decreases rapidly for larger and smaller arguments. There is, however, a strong local maximum around $-0.55\,\widehat{Q}_{11}$. The presence of this local maximum precludes the use of convex optimization algorithms.

Futhermore, little is ex ante known about the role and interpretation of the parameter matrices, especially of their out-of-diagonal components. Most parameters may change sign as illustrated in Figure 8. This makes it difficult to derive useful starting values for classical optimization algorithms from economic considerations.

Both problems – lack of good starting values and non-convexity – are elegantly solved by using population-based stochastic optimization algorithms such as differential evolution of Storn and Price (1997). This non-convex optimization algorithm only requires an interval of plausible starting values for each element of the parameter vector. The elevated computational cost of this algorithm – Storn and Price propose a population size of five to ten times the number of parameters and 100 generations – is easily accommodated thanks to the speed increases afforded by the Matrix Rotation Count algorithm and the Fast Evaluation Scheme of the likelihood function. Moreover, differential evolution can be parallelized up to the order of the population size.

To increase the precision of the point estimate, the result of the stochastic optimization is used as starting value for a simplex optimization using the algorithm of Nelder and Mead (1965), assuming local convexity around the optimum.

6.2 Matrix Representations and Optimization Constraints

The standard convention in the optimization literature is optimization over a parameter vector. As the MAJD model feature several parameter matrices and a state matrix, we have to encode the argument of the objective function as a vector. In the unconstrained case, this can be done in a straight forward manner using the vectorization function $vec()$ viz. the half-vectorization function $vech()$ for symmetric matrices. The parameter vector is then $\theta = \{vec(M), vec(R),\ vec(Q),\ \beta, \lambda_0, vec(\Lambda), \theta_\xi; vech(X_t)\}$ where θ_ξ denotes the parameters of the return jump distribution.

However, considerations of identification[16] and admissibility of the stochastic process require several constraints. These are:

- Positive definiteness and symmetry of the state X_t
- Negative definiteness and lower triangularity of M
- Positive definiteness of Λ
- Existence of $\sqrt{Id_n - R'R}$

Some of these constraints are tedious to implement in cartesian coordinates. For example, positive definiteness of the 2×2 state matrix requires three constraints: $X_{11} >$

[16] See (Gruber et al. 2010) for a detailed discussion of the model identification.

0; $X_{22} > 0$ and $X_{11}X_{22} > (X_{21})^2$. In higher dimensions the number of constraints increases with $n + n(n-1)/2$.

I therefore propose the following matrix reparametrizations that allow for an easier implementation of constraints and support a better understanding of the volatility structure. The first reparametrization transforms a constrained optimization into an unconstrained one and the second transforms some unbounded variables to bounded ones.

6.2.1 Choleski decomposition

The Choleski decomposition of a symmetric, positive definite matrix X is the unique triangular matrix D such that

$$X = DD' \tag{25}$$

Conversely, the product DD' of any matrix D will be a positive definite, symmetric matrix. The Choleski decomposition is therefore a useful representation of the state matrix. It allows for an unconstrained optimization over $vech(D)$ using (25) to recover X.

6.2.2 Eigendecomposition

A more general decomposition is the eigendecomposition or PDP-decomposition. Any square matrix A with linearly independent eigenvectors can be written as

$$A = PDP^{-1} \tag{26}$$

where D is a diagonal matrix of the eigenvalues of A and P is a matrix whos column vectors are the normalized eigenvectors of A.

The PDP decomposition makes it straight forward to implement definiteness constraints on non-symmetric matrices by constraining the sign of the diagonal elements of D.

6.2.3 Polar coordinate representation of the state

The eigendecomposition can be taken further by representing the eigenvectors in terms of polar coordinates. An n-dimensional normalized eigenvector can be expressed in polar coordinates using $n-1$ angles. For example, a general 2×2 matrix $A = PDP^{-1}$

can be expressed as

$$A = \begin{pmatrix} \sin\alpha_1 & \sin\alpha_2 \\ \cos\alpha_1 & \cos\alpha_2 \end{pmatrix} \begin{pmatrix} D_{11} & 0 \\ 0 & D_{22} \end{pmatrix} \begin{pmatrix} \sin\alpha_1 & \sin\alpha_2 \\ \cos\alpha_1 & \cos\alpha_2 \end{pmatrix}^{-1} \tag{27}$$

Expression (29) can be used to understand the structure of A and to implement structural and definiteness constraints. It is simplified for symmetric matrices like the state X_t, as $\cos\alpha_1 = \sin\alpha_2$ implies $\alpha_2 = \alpha_1 + \pi/2$. Using the symbol

$$\xi := \frac{D_{11}}{tr(X_t)} = \frac{D_{11}}{D_{11} + D_{22}} \tag{28}$$

we define the following mapping the 2×2 state matrix X_t

$$(X_{11,t}, X_{12,t}, X_{22,t}) \to (V_t, \xi_t, \alpha_{1,t}) \tag{29}$$

where $V := tr(X)$ is the (diffusive) variance level. The remaining two variables describing the volatility structure are bounded: $0 \leq \xi \leq 1$ and $0 \leq \alpha \leq \pi$. This boundedness of ξ, α makes it possible to scan the space of admissible state matrices for a given volatility level.

7 Conclusion

The useful properties of matrix affine jump diffusion (MAJD) models, introduced by Leippold and Trojani (2008) come at the price that this class of models is notoriously difficult to estimate. The matrix nature of the state space is numerically unstable and renders the evaluation of the Laplace transform computationally more costly by two orders of magnitude compared to similar scalar models. The numerical methods presented in this paper are essential for rendering the estimation and evaluation of MAJD models feasible.

I first show that MAJD models require costly nested and stochastic optimization. I then reduce the computational cost in several steps. First I formulate the Matrix Rotation Count algorithm, which makes the use of transform methods possible, resulting in a 1400-fold speed increase. Next, I propose a fast evaluation scheme of the likelihood function based on an approximation of the integration limits in the COS inversion method. This evaluation scheme vastly reduces the number of required evaluations of the Laplace transform, resulting in a 50- to 500-fold speed increase, depending on the estimation procedure. Combined with moderate use of parallelization in the differential evolution optimization scheme, I achieve a reduction in the estimation time of a factor of approximately 10^6.

I also highlight a numerical instability of the Laplace transform, that requires a deliberate choice of integration bounds, especially in the FFT method of Carr and Madan (1999). This instability has the counter-intuitive consequence that fewer elements of the Fourier sum result in a more precise result.

I finally propose a reformulation of the Matrix state space in terms of polar coordinates that allows for an elegant implementation of identification and admissibility constraints for the matrix optimization.

A The COS method

The Cosine-Fast Fourier inversion (COS) method, introduced by Fang and Oosterlee (2008), is an efficient algorithm to approximate option prices given the Laplace transform. It reduces the number of required evaluations from 2^{12} for a standard FFT of Carr and Madan (1999) to typically 200. A second useful property of the Cosine-FFT method is the fact that it does not involve an interpolation between strikes, which allows for a more precise calculation of skewness measures like $\mathcal{S}_t = \lim_{T \to t} \frac{\partial IV(T,S_t)}{\partial K}$.

The COS algorithm approximates a (density) function $f(\tau; y)$ on a finite support $[a, b]$ via a truncated cosine-series expansion:

$$f(y) = \sum_{k=0}^{\infty}{}' A_k \cos\left(k\pi \frac{y-a}{b-a}\right) \approx \sum_{k=0}^{N-1}{}' A_k \cos\left(k\pi \frac{y-a}{b-a}\right) \tag{30}$$

with

$$\begin{aligned} A_k &= \frac{2}{b-a}\int_a^b f(y)\cos\left(k\pi\frac{y-a}{b-a}\right)dy \\ &\approx \frac{2}{b-a} Re\left[\int_{\mathbb{R}} f(y)\exp\left(ik\pi\frac{y-a}{b-a}\right)dy\right] \\ &\approx \frac{2}{b-a} Re\left[\phi\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{k\pi a}{b-a}\right)\right] \end{aligned} \tag{31}$$

The price $O(y)$ of a contingent claim with payoff $v_t(y)$ and time to maturity $\tau = T - t$

is:

$$
\begin{aligned}
O(y) &= e^{-r\tau}\int_{-\infty}^{\infty} v(y)f(y|X_t)dy \qquad (32)\\
&\approx e^{-r\tau}\int_a^b v(y)\sum_{k=0}^{N-1}{}'A_k\cos\left(k\pi\frac{y-a}{b-a}\right)dy_T\\
&= e^{-r\tau}\sum_{k=0}^{N-1}{}'\frac{1}{2}(b-a)A_k\int_a^b\frac{2}{b-a}v(y)\cos\left(k\pi\frac{y-a}{b-a}\right) \qquad (33)
\end{aligned}
$$

where $y=\ln(S_T/K)$ and we assume that $f(y)\approx 0$ outside $[a,b]$.

For a plain vanilla call viz. put with payoffs $v^c(y)=[K(e^y-1)]^+$ and $v^p(y)=[K(1-e^y)]^+$ the integral $U_k:=\frac{1}{K}\int_a^b\frac{2}{b-a}v(y,T)\cos\left(k\pi\frac{y-a}{b-a}\right)dy$ evaluates as

$$
U_k^c=\frac{2}{b-a}\left(\chi_k(0,b)-\psi_k(0,b)\right)\quad\text{and}\quad U_k^p=\frac{2}{b-a}\left(-\chi_k(a,0)+\psi_k(a,0)\right)
$$

with

$$
\begin{aligned}
\chi_k(c,d) &= \int_c^d e^y\cos\left(k\pi\frac{y-a}{b-a}\right)dy\\
&= \frac{1}{1+\left(\frac{k\pi}{b-a}\right)^2}\Big[\cos\left(k\pi\frac{d-a}{b-a}\right)e^d-\cos\left(k\pi\frac{c-a}{b-a}\right)e^c\\
&\qquad +\frac{k\pi}{b-a}\sin\left(k\pi\frac{d-a}{b-a}\right)e^d-\frac{k\pi}{b-a}\sin\left(k\pi\frac{c-a}{b-a}\right)e^c\Big]\\
\psi_k(c,d) &= \int_c^d\cos\left(k\pi\frac{y-a}{b-a}\right)dy \;=\; \begin{cases}\frac{b-a}{k\pi}\left[\sin\left(k\pi\frac{d-a}{b-a}\right)-\sin\left(k\pi\frac{c-a}{b-a}\right)\right] & k\neq 0\\ d-c & k=0\end{cases}
\end{aligned}
$$

This leaves us with

$$
O(y)=e^{-r\tau}K\sum_{k=0}^{N-1}{}'\frac{1}{2}(b-a)A_kU_k \qquad (34)
$$

This expression is not very efficient: We would have to recalculate A_k for every strike, as it depends on $y=\ln(S_T/K)$ via the characteristic function. We therefore need to separate the contract-dependent part of the Laplace transform from the model-dependent part. In our case, this Laplace transform (12) can be separated as

$$
\Psi(\gamma;\tau,X_t)=\exp\left(\gamma y+tr[A(\gamma;\tau)X_t]+B(\gamma;\tau)\right)=\exp(\gamma y)\,\Psi_{Levy}(\gamma) \qquad (35)
$$

with the according characteristic function $\phi(u)=\exp(iuy)\phi_{Levy}(u)$. We now insert

(31) into (34) to obtain the COS pricing formula

$$\begin{aligned} O(y) &= e^{-r\tau} K \sum_{k=0}^{\infty}{}' \frac{1}{2}(b-a)\frac{2}{b-a} Re\left\{\phi\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{k\pi a}{b-a}\right)\right\} U_k \\ &= e^{-r\tau} K \sum_{k=0}^{N-1}{}' Re\left\{\phi_{Levy}\left(\frac{k\pi}{b-a}\right)\exp\left(i\frac{k\pi y}{b-a}\right)\exp\left(-i\frac{k\pi a}{b-a}\right)\right\} U_k \\ &= e^{-r\tau} K \sum_{k=0}^{N-1}{}' Re\left\{\phi_{Levy}\left(\frac{k\pi}{b-a}\right)\exp\left(ik\pi\frac{y-a}{b-a}\right)\right\} U_k \end{aligned} \tag{36}$$

Note that the evaluation of $\psi_{Levy}(u)$ and U_k is independent of y and therefore need not be repeated for different strikes. This expression can be evaluated with arbitrary precision for any strike K without further interpolation.

The choice of the integration limits a, b is governed by a tradeoff between two errors: too tight limits introduce an analytical error from cutting off the tails of the distribution. Too wide limits cause a numerical error as the elements of the Fourier expansion are not employed efficiently. Fang and Oosterlee (2008) propose as integration limits

$$[a,b] = \left[\kappa_1 - L\sqrt{\kappa_2 + \sqrt{\kappa_4}},\ \kappa_1 + L\sqrt{\kappa_2 + \sqrt{\kappa_4}}\right] \tag{37}$$

with κ_i denoting cumulant i of the risk-neutral distribution and the scaling factor L chosen to be 10. All cumulants of the returns distribution and therefore the integration limits depend on the parameter vector θ, the duration τ and on the state X_t.

The values of γ_k also depend on a, b and are

$$\gamma_k = \frac{k}{b(\theta,\tau,X_t) - a(\theta,\tau,X_t)}, \qquad 0 \le k \le N-1. \tag{38}$$

B Additional Tables

Panel A: Summary statistics of the data

	In-sample	Out-of sample	Total
Time frame	1996-2002	2003-01/2013	1996-01/2013
Sampling frequency		weekly	
Trading days T	359	524	883
Number of observations	37'499	85'237	122'736
Average time to maturity (days)	141.5	124.9	130.0
Average moneyness (S/K)	0.99	0.98	0.99

Panel B: Number of observations by duration and delta

	$\tau < 30$	$30 < \tau < 75$	$75 < \tau < 180$	$180 < \tau$	all
$\lvert\Delta\rvert < 0.2$	1'761	4'647	3'679	3'858	13'945
$0.2 < \lvert\Delta\rvert < 0.4$	2'576	7'460	6'369	6'769	2'3174
$0.4 < \lvert\Delta\rvert < 0.6$	2'575	8'258	7'303	7'586	25'722
$0.6 < \lvert\Delta\rvert < 0.8$	3'479	10'808	9'399	10'446	34'132
$0.8 < \lvert\Delta\rvert$	2'981	8'651	6'947	7'184	25'763
all	13'372	39'824	33'697	35'843	122'736

Table 4: Main characteristics of our S&P500 option panel. We use out-of the money calls and puts.

Panel A: Diffusion parameters

	SV_{20}	SV_{30}	SV_{31}	SVJ_{20}	SVJ_{30}	SVJ_{31}
M_{11}	-0.3121	-0.0844	-1.0716	-0.3242	-0.1231	-0.0079
	(0.0063)	(0.0020)	(0.0185)	(0.0067)	(0.0023)	(0.0002)
M_{22}	-5.0719	-5.4283	-4.9213	-4.4564	-4.2041	-2.6808
	(0.1040)	(0.1254)	(0.0489)	(0.0895)	(0.0582)	(0.0261)
M_{33}		-1.4410			-0.5517	
		(0.0307)			(0.0104)	
M_{21}			14.3050			1.0265
			(0.2173)			(0.0120)
Q_{11}	0.2370	0.1957	0.0556	0.0903	0.0742	0.0698
	(0.0024)	(0.0026)	(0.0006)	(0.0015)	(0.0010)	(0.0009)
Q_{22}	0.4209	0.4498	0.5256	0.4204	0.2853	0.2924
	(0.0057)	(0.0062)	(0.0033)	(0.0054)	(0.0026)	(0.0024)
Q_{33}		0.0718			0.0738	
		(0.0019)			(0.0016)	
Q_{12}			-0.1440			-0.0770
			(0.0021)			(0.0012)
R_{11}	-1.0000	-1.0000	-0.0431	-1.0000	-0.9997	-0.2970
	(0.0131)	(0.0134)	(0.0008)	(0.0227)	(0.0189)	(0.0036)
R_{22}	-0.5348	-1.0000	-0.6405	-0.3823	-0.7111	-0.4057
	(0.0087)	(0.0192)	(0.0055)	(0.0069)	(0.0117)	(0.0048)
R_{33}		0.9633			-0.1178	
		(0.0255)			(0.0026)	
R_{12}			-0.7672			-0.8708
			(0.0110)			(0.0121)
β_{11}	1.0000	1.0031	1.0000	1.0006	1.0064	1.0012
	(0.0160)	(0.0169)	(0.0118)	(0.0191)	(0.0180)	(0.0116)
β_{22}	1.0000	1.0007		1.0000	1.0042	
	(0.0187)	(0.0219)		(0.0197)	(0.0153)	
β_{33}		1.0162			1.0146	
		(0.0235)			(0.0187)	
M^*_{11}	-1.4051	-1.2204	-0.6378	-0.7395	-0.8289	-0.5467
	(0.0266)	(0.0298)	(0.0091)	(0.0172)	(0.0134)	(0.0083)
M^*_{22}	-1.8593	-2.2558	-2.7528	-1.9462	-1.2661	-2.6808
	(0.0401)	(0.0584)	(0.0435)	(0.0477)	(0.0221)	(0.0334)
M^*_{33}		-0.4869			-0.5539	
		(0.0116)			(0.0093)	
M^*_{21}			1.9200			0.3982
			(0.0284)			(0.0051)
β^*_{11}	1.0000	1.0017	1.0000	1.0006	1.0064	1.0012
	(0.0203)	(0.0216)	(0.0162)	(0.0200)	(0.0190)	(0.0124)
β^*_{22}	1.0000	1.0046		1.0000	1.0042	
	(0.0201)	(0.0199)		(0.0251)	(0.0232)	
β^*_{33}		1.0693			1.0146	
		(0.0316)			(0.0208)	

Panel B: Jump parameters

	SVJ_{20}	SVJ_{30}	SVJ_{31}
λ_0	0.0000	0.0003	0.0000
	(0.0003)	(0.0002)	(0.0002)
Λ_{11}	43.8971	57.3248	25.6671
	(0.9240)	(0.9276)	(0.3193)
Λ_{22}	1.0566	11.9429	15.9795
	(0.0265)	(0.1899)	(0.1933)
Λ_{33}		0.0454	
		(0.0008)	
Λ_{12}			40.4278
			(0.6332)
$\bar{k}$	-0.1500	-0.1500	
	(0.0030)	(0.0019)	
δ	0.1500	0.1500	
	(0.0027)	(0.0020)	
λ^-			7.1518
			(0.0372)
λ^+			58.3547
			(0.7690)
β^*_Λ			0.3230
			(0.0553)

Table 5: In-sample (1996/01-2002/12) parameter estimates and standard errors. Panel A: diffusion parameters. Panel B: jump parameters. For consistency and for brevity, all parameter values are reported using a notation based on matrix AJD, i.e., by considering Bates- and Heston-type models as nested diagonal matrix AJD models.

References

Attari, Mukarram (2004) 'Option pricing using fourier transforms: A numerically efficient simplification.' *SSRN Working Paper 55042*

Bates, David (1996) 'Jumps and stochastic volatility: Exchange rate processes implicit in deutschemark options.' *Review of Financial Studies* 9, 69–108

—— (2000) 'Post-'87 crash fears in the S&P 500 futures option market.' *Journal of Econometrics* 94(1), 181–238

Benabid, Anas, Harry Bensusan, and Nicole El Karoui (2009) 'Wishart stochastic volatility : Asymptotic smile and numerical framework.' *Working paper March 2009*

Bru, Marie-France (1991) 'Wishart processes.' *Journal of Theoretical Probability* 4, 725–751

Carr, Peter, and Dilip B. Madan (1999) 'Option valuation using the fast fourier transform.' *Journal of Computational Finance* 2, 61–73

Carr, Peter, and Liuren Wu (2007) 'Stochastic skew in currency options.' *Journal of Financial Economics* 86 (1), 213–247

Carr, Peter, and Liuren Wu (2009) 'Leverage effect, volatility feedback, and self-exciting market disruptions: Disentangling the multi-dimensional variations in s&p500 index options.' *Bloomberg Portfolio Research Paper No. 2009-03*

Chen, Ren-Raw, and Louis Scott (1992) 'Pricing interest rate options in a two-factor cox-ingersoll-ross model of the term structure.' *Review of Financial Studies*

Christoffersen, Peter, Steve Heston, and Kris Jacobs (2009) 'The shape and term structure of the index option smirk: Why multifactor stochastic volatility models work so well.' *Management Science* 55, 1914–1932

Cont, Rama, and Sana Ben Hamida (2005) 'Recovering volatility from option prices by evolutionary optimization.' *Journal of Computational Finance, Vol. 8, No. 4, Summer 2005*

da Fonseca, Jose, Martino Grasselli, and Claudio Tebaldi (2008) 'A multifactor volatility heston model.' *Quantitative Finance* 8 (6), 591–604

Duffie, Darrell, Jun Pan, and Kenneth Singleton (2000) 'Transform analysis and asset pricing for affine jump-diffusions.' *Econometrica* 68 (6), 1343–1376

Fang, Fang, and Kees Oosterlee (2008) 'A novel pricing method for european options based on fourier-cosine expansions.' *SIAM Journal on Scientific Computing* 31(2), 826–848

Gruber, Peter H., Claudio Tebaldi, and Fabio Trojani (2010) 'Three make a smile – dynamic volatility, skewness and term structure components in option valuation.' Working Paper. University of Lugano and Bocconi University

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

Huang, Jing-Zhi, and Liuren Wu (2004) 'Specification analysis of option pricing models based on time-changed levy processes.' *Journal of Finance* 59, 1405–39

Kahl, Christian, and Peter Jäckel (2005) 'Not-so-complex logarithms in the heston model.' *Wilmott* pp. 94–103

Leippold, Markus, and Fabio Trojani (2008) 'Asset pricing with matrix jump diffusions.' *SSRN eLibrary*

Lord, Roger, and Christian Kahl (2010) 'Complex logarithms in heston-like models.' *Mathematical Finance* 20(4), 671–694

Mayerhofer, Eberhard (2014) 'Wishart processes and wishart distributios: An affine process point of view.' *Lecture Notes*

Muhle-Karb, Johannes, Oliver Pfaffel, and Robert Stelzer (2010) 'Option pricing in multivariate stochastic volatility models of ou type.' *Working paper*

Nelder, J.A., and R. Mead (1965) 'A simplex method for function minimization.' *The Computer Journal* 7, 308–313

Schöbel, Rainer, and Jianwei Zhu (1999) 'Stochastic volatility with an ornstein-uhlenbeck process: An extension.' *European Finance Review* 3, 23–46

Storn, Rainer, and Kenneth Price (1997) 'Differential evolution – a simple and efficient heuristic for global optimization over continuous.' *Journal of Global Optimization* 11, 341–359