An adaptive Filon quadrature for stochastic volatility models

Fabien Le Floc'h

(v1.1 released April 2016)

Abstract *This paper describes an adaptive Filon quadrature for the computation of option prices under the Heston stochastic volatility model. A comparison against popular alternatives in terms of accuracy and performance is then presented, ending with the concrete case of model calibration on different market data.*

Key Words: Filon, Heston, Cos method, stochastic volatility, calibration

1. Introduction

The valuation of European options under the Heston model, or under other stochastic volatility models where the characteristic function is known analytically, involves the computation of a Fourier transform type of numerical integration. Many variations exist. Heston derived the initial valuation formula from a probabilitic interpretation in (Heston, 1993), while Carr and Madan developped a more direct Fourier transform approach, which enabled the use of the fast Fourier transform (FFT) algorithm in (Carr and Madan, 1999). Their damping parameter α also spun off a family of alternative valuation formulae, with better convergence properties than the original Heston formula. The Lewis formula, corresponding to $\alpha = -\frac{1}{2}$ is particularly popular as it is simple to evaluate, is well defined everywhere and has quadratic denominator (Lewis, 2001). We should also mention the choice $\alpha = -1$ proposed by Attari, which requires a bit more care around zero, and the choice $\alpha = 0$ studied by Joshi and Yang (2011). Joshi and Yang combined this specific choice of α along with matching the value and first derivative of the characteristic function with a Black-Scholes control variate, which they found was improving the efficiency of real-time calibration. A procedure to find the optimal (strike and maturity dependent) α, leading to the ability of pricing extremely out-of-the-money options, including those whose prices are beyond machine epsilon, is described in (Lord and Kahl, 2007).

In terms of integration, Kilin (2007) showed that the FFT method was not competitive against a simple quadrature with cached characteristic function values. There is however no real consensus on the quadrature to be used. Andersen and Piterbarg advocate for the simplest trapezoidal method with a good truncation (Andersen and Piterbarg, 2010). Kahl and Jaeckel, as well as Lord and Kahl propose the adaptive Gauss-Lobatto of Gander and Gautschi combined with a log-transform (Kahl and Jäckel, 2005), while Joshi and Yang use the Gauss-Laguerre quadrature.

The Filon quadrature is particularly suited to the computation Fourier integrals (Filon, 1928; Tuck, 1967). A generalisation of this method has been applied to the problem of pricing options under the displaced lognormal Heston model in (Dickinson, 2011). Their proposed algorithm however involves many specific choices of parameters. In this paper, we will look at a particularly simple and effective adaptive Filon method, where the adaptive integration points are reusable across strikes, for a given maturity. A similar technique has been used in seismology in (Hai-Ming and Xiao-Fei, 2001) and in finance the context of volatility swap pricing in (Le Floc'h, 2015). We will evaluate its behavior against popular Heston integration methods on challenging Heston parameters. Finally, we will take a look at the full volatility surface calibration performance and stability.

While this paper focuses on the Heston stochastic volatility model, the proposed technique is more general and can be applied other models with closed form characteristic functions like Bates (Bates, 1996), Schobel-Zhu (Schöbel and Zhu, 1999), or Double-Heston (Christoffersen *et al.*, 2009).

2. The Filon quadrature

The Filon quadrature is used to evaluate integrals such as

$$I_c = \int_a^b f(p)\cos(xp)dp \ , \ I_s = \int_a^b f(p)\sin(xp)dp. \tag{1}$$

When x is not small, the rapid oscillations are particularly challenging for ordinary quadrature formulae such as Simpson's. The idea of Filon is to fit f by a parabola at three equidistant points as in Simpson's method, but to integrate exactly the terms in $p^k\cos(xp), p^k\sin(xp)$. Applied to a subdivision in $2n$ parts of the original interval $[a,b]$ this results in the following formula:

$$I_c = h\left\{\alpha\left[f(b)\sin(xb) - f(a)\sin(xa)\right] + \beta C_{ce} + \gamma C_{co}\right\}, \tag{2}$$

$$I_s = h\left\{-\alpha\left[f(b)\cos(xb) - f(a)\cos(xa)\right] + \beta C_{se} + \gamma C_{so}\right\}, \tag{3}$$

where $h = \frac{b-a}{2n}, \theta = hx$, C_{ce} is the sum of even ordinates of $f(p)\cos(px)$ less half the end ordinates, C_{co} is the sum of odd ordinates of $f(p)\cos(px)$, C_{se} is the sum of even ordinates of $f(p)\sin(px)$ less half the end ordinates, C_{so} is the sum of odd ordinates of $f(p)\sin(px)$, and

$$\theta^3\alpha = \theta^2 + \theta\sin\theta\cos\theta - 2\sin^2\theta, \tag{4}$$

$$\theta^3\beta = 2\left[\theta\left(1+\cos^2\theta\right) - 2\sin\theta\cos\theta\right], \tag{5}$$

$$\theta^3\gamma = 4\left[\sin\theta - \theta\cos\theta\right]. \tag{6}$$

Chase and Fosdick (1969) derived a robust numerical algorithm taking special care of the case where θ is small. A more thorough description of the Filon quadrature is presented in Tranter (1951).

3. Flinn's improvement

Instead of fitting a quadratic as in Filon's quadrature, Flinn (1960) uses a fifth-order polynomial to fit the middle and end points values and first derivatives, resulting not only in a higher order quadrature, but also in one that works better on larger intervals. The resulting formula is not much more complicated:

$$\begin{aligned} I_c = h\big\{&S\left[f(b)\sin(xb) - f(a)\sin(xa)\right] + hP\left[f'(b)\cos(xb) - f'(a)\cos(xa)\right] \\ &+RC_{ce} + hQC'_{se} + NC_{co} + hMC'_{so}\big\}, \end{aligned} \tag{7}$$

$$\begin{aligned} I_s = h\big\{&-S\left[f(b)\cos(xb) - f(a)\cos(xa)\right] + hP\left[f'(b)\sin(xb) - f'(a)\sin(xa)\right] \\ &+RC_{se} - hQC'_{ce} + NC_{so} - hMC'_{co}\big\}, \end{aligned} \tag{8}$$

where C'_{ce} is the sum of even ordinates of $f'(p)\cos(px)$ less half the end ordinates, C'_{co} is the sum of odd ordinates of $f'(p)\cos(px)$, C'_{se} is the sum of even ordinates of $f'(p)\sin(px)$ less half the end ordinates, C'_{so} is the sum of odd ordinates of $f'(p)\sin(px)$. We refer the reader to Flinn's paper for the values of the constants M, N, P, Q, R, S. It also provides expansions useful for the case where θ is small.

4. Characteristic Functions

4.1 Heston

The Heston stochastic volatility model of an asset F is described by the following system of stochastic differential equations (Heston, 1993):

$$dF = \sqrt{v}FdW_F \tag{9}$$

$$dv = \kappa(\theta - v)dt + \sigma\sqrt{v}dW_v \tag{10}$$

with $dW_F dW_v = \rho dt$. For an equity of spot S, and maturity T, F represents the forward to maturity $F(t) = S(t)e^{(r-q)(T-t)}$ with r the relevant interest rate, and q the dividend yield.

In order to avoid complex discontinuities, we rely on Gatheral formulation for the normalized Heston characteristic function (Lord and Kahl, 2006):

$$\phi(z) = e^{\frac{v_0}{\sigma^2}\frac{1-e^{-DT}}{1-Ge^{-DT}}(\kappa - i\rho\sigma z - D) + \frac{\kappa\theta}{\sigma^2}\left((\kappa - i\rho\sigma z - D)T - 2\ln(\frac{1-Ge^{-DT}}{1-G})\right)} \tag{11}$$

with

$$D = \sqrt{(\kappa - i\rho\sigma z)^2 + (z^2 + iz)\sigma^2} \tag{12}$$

$$G = \frac{\kappa - i\rho\sigma z - D}{\kappa - i\rho\sigma z + D} \tag{13}$$

The standard characteristic function is then just $\mathbb{E}[e^{iz\ln(F(T))}] = e^{iz\ln(F(0))}\phi(z)$.

4.2 Black-Scholes

The normalized Black-Scholes characteristic function with volatility σ_B is simply:

$$\phi_B(z) = e^{-\frac{1}{2}\sigma_B^2 T(z^2+iz)} \tag{14}$$

We will see it can be used as control variate for other stochastic volatility models.

5. Pricing formulae

Originally, Heston proposed a Black-Scholes like formula in (Heston, 1993). The vanilla call option price can be expressed as (Carr and Madan, 1999):

$$C(F,K,T) = Fe^{-rT}\left[\frac{1}{2} + \frac{1}{\pi}\int_0^\infty \Re\left(\frac{e^{-iuk}\phi(u-i)}{iu\phi(-i)}\right)du\right] + Ke^{-rT}\left[\frac{1}{2} + \frac{1}{\pi}\int_0^\infty \Re\left(\frac{e^{-iuk}\phi(u)}{iu}\right)du\right] \tag{15}$$

where $k = \ln(K)$. As the integrand is sometimes highly oscillating, a formula based on a damped option price is proposed in (Carr and Madan, 1999)[1]:

$$C(F,K,T) = R_\alpha(F,K) + F\frac{e^{-\alpha x}e^{-rT}}{\pi}\int_0^\infty \Re\left(e^{-iux}\frac{\phi(u-(\alpha+1)i)}{(\alpha+iu)(\alpha+1+iu)}\right)du \tag{16}$$

where α is a damping parameter, $x = \ln(\frac{K}{F})$, and $R_\alpha(F,K) = e^{-rT}\left[F\cdot 1_{\alpha\leq 0} - K\cdot 1_{\alpha\leq -1} - \frac{1}{2}(F\cdot 1_{\alpha=0} - K\cdot 1_{\alpha=-1})\right]$. A method to find the optimal α, allowing to price extremely out-of-the-money options is described in (Lord and Kahl, 2007). For put options, one can just use the same formula, but with $\alpha < -1$.

A popular alternative formulation with a quadratic denominator was found by Lewis in (Lewis, 2001):

$$C(F,K,T) = Fe^{-rT} - \frac{\sqrt{FK}e^{-rT}}{\pi}\int_0^\infty \frac{\Re\left(e^{-iux}\phi(u-\frac{i}{2})\right)}{u^2+\frac{1}{4}}du \tag{17}$$

This is equivalent to taking $\alpha = -\frac{1}{2}$ in Carr-Madan formula. It is a method of choice in (Andersen and Piterbarg, 2010), where they introduce a Black-Scholes control variate to significantly improve convergence:

$$C(F,K,T) = BS(F,K,T,\sqrt{v_0}) + \frac{\sqrt{FK}e^{-rT}}{\pi}\int_0^\infty \Re\left(e^{-iux}\frac{\phi_B(u-\frac{i}{2})-\phi(u-\frac{i}{2})}{u^2+\frac{1}{4}}\right)du \tag{18}$$

Put options can be priced with the same formula but using the Black-Scholes put option price instead of the call option price. Finally, a recent formula with Black-Scholes control-variate is studied in (Joshi and Yang, 2011).

$$C(F,K,T) = BS(F,K,T,\sigma_B) + \frac{e^{-rT}}{\pi}\int_0^\infty \Re\left(e^{-iux}\frac{\phi_B(u-i)-\phi(u-i)}{u(u-i)}\right)du \tag{19}$$

This is similar to taking $\alpha = 0$ in Carr-Madan formula. The authors also find that the optimal choice of volatility σ_B used in the control variate corresponds to $\phi'_B(-i) = \phi'(-i)$. Without this choice of control variate, the formula would require special treatment at 0.

5.1 Truncation

5.1.1 A domain transformation

One immediate issue with the integration is the infinity of the range of integration. One possibility is to use a log transform $z(u)$ as proposed in (Kahl and Jäckel, 2005; Lord and Kahl, 2007) combined with some adaptive integration algorithm, for example Gauss-Lobatto. Defining:

$$u(z) = -\frac{\ln(z)}{C_\infty}\ , \tag{20}$$

the Lewis pricing formula can be rewritten as:

$$\int_0^\infty \frac{\Re\left(e^{-iux}\phi(u-\frac{i}{2})\right)}{u^2+\frac{1}{4}}du = \int_0^1 \frac{\Re\left(e^{-iu(z)x}\phi(u(z)-\frac{i}{2})\right)}{(u(z)^2+\frac{1}{4})(u(z)C_\infty)}dz. \tag{21}$$

[1] This is not strictly the Carr-Madan formula: it is scaled by the forward.

The constant C_∞ is chosen so that the transformed variable $z(u)$ has same asymptotic behavior as the characteristic function at ∞:

$$\lim_{u\to+\infty}\frac{1}{u}\ln(\phi(u-(\alpha+1)i)-\phi_B(u-(\alpha+1)i)) = -(C_\infty+iD_\infty), \tag{22}$$

with

$$C_\infty = \frac{v_0+\kappa\theta T}{\sigma}\sqrt{1-\rho^2}\ , \tag{23}$$

$$D_\infty = \frac{v_0+\kappa\theta T}{\sigma}\rho. \tag{24}$$

This can be applied to any of the formulae.

5.1.2 Andersen-Piterbarg approach to Lewis

Another possibility is to find a good truncation and integrate directly as proposed in (Andersen and Piterbarg, 2010), relying on the following approximation when $u_{\max}$ is sufficiently large:

$$\int_{u_{max}}^{\infty}\left|\frac{\phi(u-\frac{i}{2})-\phi_B(u-\frac{i}{2})}{u^2+\frac{1}{4}}\right|du \leq e^{-C_\infty u_{\max}}\int_{u_{\max}}^{\infty}\frac{du}{u^2} \tag{25}$$

Both methods actually rely on the same number C_∞. For a relative tolerance of ϵ_u, the truncation limit $u_{\max}$ is found by solving:

$$\frac{e^{-C_\infty u_{\max}}}{u_{\max}} = \epsilon_u \tag{26}$$

or, to avoid numerical overflows:

$$-C_\infty u_{\max} - \ln(u_{\max}) = \ln(\epsilon_u) \tag{27}$$

5.1.3 Short expiries

The truncation can be invalid for short expiries ($T < 0.1$) as, then, $e^{-D(u_{\max})T} \not\ll 1$. A Taylor expansion around $T=0$ gives the Black-Scholes like characteristic function:

$$\ln(\phi(u)) = -\frac{1}{2}v_0T(u^2+iu)+\mathcal{O}(T^2).$$

Then our approximation for short expiries is $\hat{u}_{\max}$ that solves:

$$-\frac{1}{2}v_0T\hat{u}_{\max}^2 - \ln(\hat{u}_{\max}) = \ln(\epsilon_u). \tag{28}$$

We found that a good practical rule for the full range of expiries is just to use $\max(u_{\max}, \hat{u}_{\max})$.

Similarly, the log-transform is not always well behaved for short expiries. While the integrand is well behaved around $z=0$, this is not always true around $z=1$ as the characteristic function becomes Black-Scholes like for short expiries. In practice, an adaptive quadrature might require the evaluation of the integrand at very closely spaced points near $z=1$, sometimes of distance smaller than the machine epsilon. As a work-around, we can follow Lord and Kahl proposed transform for the Black-Scholes characteristic function and cap C_∞ to:

$$\bar{C}_\infty = \sqrt{\frac{v_0T}{2}}. \tag{29}$$

Let's illustrate the importance of our modifications for the short expiries with the Heston parameters $v_0 = 0.826, \kappa = 0.254, \theta = 0.320, \sigma = 0.344, \rho = -0.557$ on an option of maturity $T = 0.0182$ with forward and strike $F = 1000, K = 1400$. Table 1 shows that without the adjustment to the truncation, the price is just wrong. The adjustment to C_∞ is particularly important for non adaptive quadratures: a Simpson quadrature with 10000 points has a huge error (of around 40% of the reference price). The adaptive Gauss-Lobatto of Espelid (2003) still works with the non capped C_∞ but requires much more points (almost 10000).

Table 1.: Price with the Lewis formula and Black-Scholes control variate with the following Heston parameters: $v_0 = 0.826, \kappa = 0.254, \theta = 0.320, \sigma = 0.344, \rho = -0.557$ on an option of maturity $T = 0.0182$ with forward and strike $F = 1000, K = 1400$.

Quadrature	Points	Truncation	Price
Simpson	10000	$u_{\max} = 12.69$	0.0651248
Simpson	10000	$\hat{u}_{\max} = 56.04$	0.1073414
Simpson	10000	None / $C_\infty = 1.998$	0.0685957
Simpson	10000	None / $\bar{C}_\infty = 0.086$	0.1073414
Gauss-Lobatto	1487 (Adaptive)	$\hat{u}_{\max} = 56.04$	0.1073414
Gauss-Lobatto	9107 (Adaptive)	None / $C_\infty = 1.998$	0.1073414
Gauss-Lobatto	5087 (Adaptive)	None / $\bar{C}_\infty = 0.086$	0.1073414

This is a key weakness of the domain transformation. While our fix improves the behaviour in general, it is not clear if it stays always well behaved near $z = 0$, and our choice of applying the cap for $T < 0.1$ is somewhat arbitrary.

Interestingly, on this example, the non-transformed, truncated integral is much faster to evaluate with Gauss-Lobatto than the log-transformed equivalent. We found this to be generally the case for short expiries, while for longer expiries, the log transform was more beneficial.

5.2 *Filon on the Lewis Formula*

The Lewis formula (Equation 17) can be rewritten as:

$$C(F,K,T) = Fe^{-rT} - \frac{\sqrt{FK}e^{-rT}}{\pi}\int_0^\infty \frac{\Re\left(\phi(u-\frac{i}{2})\right)}{u^2+\frac{1}{4}}\cos(ux) + \frac{\Im\left(\phi(u-\frac{i}{2})\right)}{u^2+\frac{1}{4}}\sin(ux)du \quad (30)$$

To compute the vanilla option price, we can therefore apply the cos and sin Filon quadratures respectively to the functions

$$f(u) = \frac{\Re\left(\phi(u-\frac{i}{2})\right)}{u^2+\frac{1}{4}}\ ,\ g(u) = \frac{\Im\left(\phi(u-\frac{i}{2})\right)}{u^2+\frac{1}{4}}. \quad (31)$$

A significant property of the functions $f(u)$ and $g(u)$ is that they are not dependent on the log-moneyness x.

5.3 *Flinn on the Lewis Formula*

It is not much more costly to evaluate both the characteristic function ϕ and its derivative ϕ'. The first derivative can be computed at the same time through simple analytical differentiation, reusing most of the costly intermediate variables.

To compute the vanilla option price, we can therefore apply the cos and sin Flinn quadratures respectively to the functions

$$f(u) = \frac{\Re\left(\phi(u-\frac{i}{2})\right)}{u^2+\frac{1}{4}}\ ,\ f'(u) = \frac{\Re\left(\phi'(u-\frac{i}{2})\right)}{u^2+\frac{1}{4}} - 2\frac{\Re\left(\phi(u-\frac{i}{2})\right)}{\left(u^2+\frac{1}{4}\right)^2}u \quad (32)$$

and

$$g(u) = \frac{\Im\left(\phi(u - \frac{i}{2})\right)}{u^2 + \frac{1}{4}}\ ,\ g'(u) = \frac{\Im\left(\phi'(u - \frac{i}{2})\right)}{u^2 + \frac{1}{4}} - 2\frac{\Im\left(\phi(u - \frac{i}{2})\right)}{\left(u^2 + \frac{1}{4}\right)^2}u. \tag{33}$$

5.4 Making it adaptive

The similarity between Simpson's and Filon's method suggest to apply the same adaptive technique as for Simpson. We simply apply the adaptive Simpson algorithm to the computation of the two integrals

$$\bar{I}_c = \int_0^{u_{\max}} f(u)du\ ,\ \bar{I}_s = \int_0^{u_{\max}} g(u)du \tag{34}$$

and record the abscissae u_i and function values (f_i, g_i) (eventually along with the derivative values) in a map structure. We then sort the values by abscissa and use the Filon or the Flinn quadrature to the intervals defined by the abscissae, three points by three points. In practice, the Filon constants α, β, γ can be reused across adjacent intervals, unless the step size has changed. The cost of evaluating the trigonometric functions can be further reduced by taking advantage of the identities $\cos(u + \theta) = \cos(u)\cos(\theta) - \sin(u)\sin(\theta)$, $\sin(u + \theta) = \cos(u)\sin(\theta) + \sin(u)\cos(\theta)$ and evaluating the cos and sin integrals together.

The underlying assumption is that an adaptive Simpson algorithm is going to approximate f and g well enough so that the Filon or the Flinn quadrature on the same points is accurate enough. The two integrals of equation (34) correspond to the Lewis formula at-the-money, that is, where $x = 0$.

There are various adaptive Simpson algorithms. A somewhat popular one, `squank`, is based on (Lyness, 1969; 1970), which refines the Simpson method with a Richardson extrapolation and uses interval bisection. Gander and Gautschi (2000) propose an alternative algorithm `adaptsim` with a different local error control relying on an initial estimate of the integral by a five points Monte-Carlo integration. Espelid (2003) wrote a more recent algorithm `modsim` using directly a five points Newton-Cotes formula (which is equivalent to the extrapolated Simpson formula) with refined local error control based on Null rules. This later algorithm can be transformed to a globally adaptive algorithm the same way the author transforms its locally adaptive `coteda` to the globally adaptive `coteglob`.

While, in practice, the choice of adaptive algorithm matters little for the stability of the overall adaptive Filon method, we found that Gander and Gautchi `adaptsim` was in general failing much more than `squank` or `modsim` algorithms when applied directly to the Lewis formula instead of the Filon reduction. This is mostly due to a bad initial estimate of the integral. Furthermore `modsim` algorithm was in general faster than `squank` (requiring less points), even more so with global error control.

The precomputed abscissae and function values can be reused to value options at different strikes. In contrast with the standard adaptive quadrature methods applied directly to the Lewis formula, the adaptivity is independent of the strike here.

6. Numerical results

6.1 Lewis and the optimal α

Combined with the adaptive Filon quadrature, it could be interesting to use the optimal α for at-the-money options instead of Lewis choice $\alpha = -\frac{1}{2}$. Using Lord and Kahl optimal α search method described in (Le Floc'h, 2013), we find out that the optimal at-the-money α is in reality never too far from Lewis choice on a wide variety of Heston parameters (Table 2).

Table 2.: At-the-money optimal α for various Heston parameters.

T	v_0	κ	θ	σ	ρ	optimal α
0.05	0.090	1.000	0.090	1.000	-0.300	-0.500920
1.00	0.090	1.000	0.090	1.000	-0.300	-0.513340
5.00	0.090	1.000	0.090	1.000	-0.300	-0.526311
0.02	0.826	0.254	0.320	0.344	-0.557	-0.500226
20.0	0.826	0.254	0.320	0.344	-0.557	-0.559124
0.10	0.0225	0.100	0.010	2.000	0.500	-0.493715
1.00	0.0225	0.100	0.010	2.000	0.500	-0.434065
10.0	0.0225	0.100	0.010	2.000	0.500	-0.500000

6.2 Challenging Heston parameters

6.2.1 Medium maturity

We consider an option of maturity $T = 1$ and strike $K = 0.25$ on an asset following the Heston stochastic volatility model with parameters $v_0 = 0.0225, \kappa = 0.1, \theta = 0.01, \sigma = 2.0, \rho = 0.5, F = 1$. The option is therefore very out of the money. We have set the truncation level at 10^{-8}, and use the same accuracy for the various adaptive quadratures considered. Except for the Cos method of Fang and Oosterlee (2008) and the Lord and Kahl optimal α method, we rely on the Lewis formula with control variate.

Under those settings, Table 3 shows that the doubly adaptive Newton-Cotes quadrature `coteda` of Espelid (2003) has a very high error, well above the tolerance for a tolerance of 10^{-8} (but is fine with a tolerance of 10^{-7} or 10^{-9}). The adaptive Gauss-Lobatto `modlob` has no such issue but requires a large number of points (over 10000). The Cos method requires a truncation at $L = 30$ to achieve a reasonable accuracy with 1000 points, well above the recommended $L = 8$ of their paper. The Flinn method combined with a globally adaptive Simpson quadrature requires on 485 points to achieve an accuracy of 10^{-10} per unit notional, which is as accurate as the direct `modlob` with more than 10000 points.

Table 3.: 1Y Put option of notional 100,000 and strike 25% with Heston parameters $v_0 = 0.0225, \kappa = 0.1, \theta = 0.01, \sigma = 2.0, \rho = 0.5$ under various numerical methods.

Method	Tolerance	Points	Price
`modlob` Lord-Kahl	1e-8	7947	119.385327
`coteda`	1e-8	1361	99.666129
`modlob`	1e-8	10987	119.385324
`modlob` Log	1e-8	9527	119.385347
`modsim`	1e-8	8853	119.324143
`modsim` Filon	1e-8	733	119.385917
`globsim` Flinn	1e-8	485	119.385352
Cos	L=12	1000	0.0
Cos	L=16	1000	123.033165
Cos	L=30	1000	119.387924

Interestingly, using the adaptive Simpson `modsim` directly on the Lewis formula requires many more points than our adaptive Flinn method (close to 9000) for a lower overall accuracy.

6.2.2 Short maturity

We use here the same parameters as in table 1 but set the tolerance at 10^{-8} instead of 10^{-12}. The Gauss-Lobatto quadrature requires much less points with this tolerance level. Table 4 shows that the adaptive Flinn method still requires around four times less points than the direct `modlob`.

Table 4.: Call optionof maturity $T = 0.0182$ with forward and strike $F = 1000, K = 1400$ and Heston parameters $v_0 = 0.826, \kappa = 0.254, \theta = 0.320, \sigma = 0.344, \rho = -0.557$ under various numerical methods.

Method	Tolerance	Points	Price
`modlob`	1e-8	547	0.107341552
`modlob` Log	1e-8	737	0.107341448
`modsim`	1e-8	645	0.107341552
`modsim` Filon	1e-8	329	0.107341554
`globsim` Flinn	1e-8	125	0.107341552

On options of longer maturities the number of points used by `modlob` and by our adaptive Flinn is more similar.

6.3 Performance

Let us look at the performance and accuracy of the various quadratures, using the Lewis formula with control variate. We use an accuracy of 1E-6 for the adaptive quadratures while the truncation is done for a relative tolerance of 1E-9. The reference is Lord-Kahl with an adaptive quadrature of accuracy 1E-10.

We price hundred put options of different strikes but same maturity with standard Heston parameters for an equity. Table 5 shows that the adaptive Flinn method is up to ten times faster than a direct adaptive Gauss-Lobatto quadrature. This should not be too surprising since the adaptivity is independent of the strike with the Flinn method, but not with the direct approach.

Table 5.: Performance and Accuracy of the various integrations to price 100 put options of strikes between 0.4 and 1.6 with the following Heston parameters: $\kappa = 1.0, \theta = 0.1, \sigma = 1.0, v_0 = 0.1, \rho = -0.9$.

Method	RMSE	Total Time(s)
Maturity of 2 weeks		
`modlob` Log	1.74E-11	0.022
`modlob`	2.91E-11	0.015
`modsim` Filon	1.82E-10	0.004
`globsim` Flinn	4.15E-12	0.002
Maturity of 2 years		
`modlob` Log	2.27E-9	0.011
`modlob`	4.45E-12	0.008
`modsim` Filon	7.29E-10	0.005
`globsim` Flinn	6.57E-12	0.003

6.4 Heston Calibration

6.4.1 Error measure

The calibration of the Heston model consists in minimizing the difference between the market implied volatilities (or option prices) and the model implied volatilities (or option prices).

Our implied volatility error measure will simply be the weighted root mean square error in implied volatilities:

$$M_\sigma = \frac{\sqrt{\sum_{i=0}^{N} w_i^2 \left(\sigma_i^H - \sigma_i^M\right)^2}}{\sqrt{\sum_{i=0}^{N} w_i^2}} \tag{35}$$

where N is the total number of option quotes we calibrate against and σ_i^H is the Black implied volatility[1] obtained from the Heston model price and σ_i^M is the market implied volatility. In our numerical examples, we will just take $w_i = 1$.

An alternative is to use the root mean square error in option prices:

$$M_V = \frac{\sqrt{\sum_{i=0}^{N} w_i^2 \left(V_i^H - V_i^M\right)^2}}{\sqrt{\sum_{i=0}^{N} w_i^2}} \tag{36}$$

where V_i^H is the Heston model option price and V_i^M is the market option price. We use only out-of-the-money option prices: Calls for $F \leq K$ and Puts for $F > K$. With equal weights, such a measure results in a significantly low quality of fit for shorter maturities as option prices increase with the maturity. We will instead use an inverse vega weighting:

$$w_i = \min\left(\frac{1}{\nu_i^M}, \frac{1000}{F}\right) \tag{37}$$

where $\nu_i^M = \frac{\partial V_i^M}{\partial \sigma}$ is the Black Vega corresponding the market option price V_i^M. The measure should then be not so different from the measure on volatilities M_σ with equal weights as the partial derivative towards a model parameter p_j can be written as:

$$\frac{\partial V}{\partial p_j} = \frac{\partial V}{\partial \sigma}\frac{\partial \sigma}{\partial p_j}. \tag{38}$$

The cap to $\frac{1000}{F}$ is necessary to avoid taking into account too out-of-the money prices, which won't be all that reliable numerically.

6.4.2 A smart initial guess

From five well chosen option implied volatilities, corresponding to the shortest-expiry at-the-money volatility, 2 calendar spreads between t_1 and t_2 of log-moneyness $-x_0$ and x_0, Forde *et al.* (2012) show how find the Heston parameters by solving a simple linear system for a Heston small-time expansion that passes exactly through those points.

Chosen option strikes and maturities can make a big difference on the initial guess. A good rule is +/- 20% (and sometimes +/- 5%) around the money and discard the first maturities. In practice, we found that taking the best guess of the two log-moneyness +/-5% and +/-20% was successfull as initial guess on a wide variety of surfaces. When the options are chosen closer to the money, we noticed that a virtual butterfly spread arbitrage could arise, modifying the

[1] A fast and robust algorithm to obtain the implied volatility from an option price is the one of Jäckel (2013). When no implied volatility corresponds to the model option price, which can happen because of numerical error, we just fix the implied volatility to zero.

formula to ensure the value of the butterfly (noted C in (Forde *et al.*, 2012, p. 8)) is positive was enough to make the algorithm robust.

We can then apply a Levenberg-Marquardt minimizer (Moré, 1978) with our choice of error measure to find the optimal Heston parameters for given market option prices.

An alternative would be to rely on the differential evolution algorithm of Storn and Price (1997) to find a good initial guess. This has however the risk to be less stable and is much slower than the smart initial guess that was found to work well in general in (Le Floc'h, 2013).

6.4.3 Results

We first consider the market data of Kahalé (2004), $r = 0.06$. We use a relative tolerance of 10^{-6} for the adaptive quadratures, and a truncation tolerance of 10^{-9}. We remove the options of price (normalized by the spot) lower than the quadrature tolerance from the data. On this data, this corresponds to a single point of expiry $T = 0.175$ and strike $K = 826$. This cleaning step is particularly important to ensure the stability of the calibration using the volatility error measure.

The calibration of Heston using the volatility error measure M_σ is not slower than the calibration using the inverse vega weighted price error measure M_V (Table 6 vs Table 7). It could actually be faster when the initial guess is not as good. But it is also less stable: slightly different Heston parameters (especially v_0 and κ) can lead to a very similar error measure and as a consequence, the calibrated parameters can fluctuate depending on the method or its tolerance level, this is especially true if small option prices are not filtered out. In (Le Floc'h, 2013) it was found necessary to add more weights for near-the-money options in order to stabilise the calibration.

Table 6.: Heston calibration under the volatility error measure for the option data of Kahalé (2004).

Method	v_0	κ	θ	σ	ρ	RMSE	Time(s)
Guess	0.0128	1.581	0.0329	0.381	-0.400	N/A	N/A
`globsim`	0.0095	5.446	0.0232	0.933	-0.584	5.199E-3	0.79
`modlob`	0.0095	5.441	0.0232	0.932	-0.584	5.199E-3	0.38
`modlob` Log	0.0095	5.436	0.0232	0.932	-0.584	5.199E-3	0.57
`globsim` Flinn	0.0095	5.441	0.0232	0.932	-0.584	5.199E-3	0.14
`modsim` Filon	0.0095	5.403	0.0232	0.927	-0.584	5.200E-3	0.15

The calibration using the weighted price error measure is much more stable, we found that changing the weights to increase the importance of near-the-money options did not change much the outcome of the calibration.

Table 7.: Heston calibration under the weighted price error measure for the option data of Kahalé (2004).

Method	v_0	κ	θ	σ	ρ	RMSE	Time(s)
Guess	0.0128	1.581	0.0329	0.381	-0.400	N/A	N/A
`globsim`	0.0131	2.421	0.0243	0.455	-0.644	1.445E-2	0.75
`modlob`	0.0131	2.421	0.0243	0.455	-0.644	1.445E-2	0.36
`modlob` Log	0.0131	2.421	0.0243	0.455	-0.644	1.445E-2	0.62
`globsim` Flinn	0.0131	2.421	0.0243	0.455	-0.644	1.445E-2	0.13
`modsim` Filon	0.0131	2.421	0.0243	0.455	-0.644	1.445E-2	0.21

On a 2.5Ghz Intel Core i5-3210M with Go 1.6, the calibration with the adaptive Flinn method is between two times and six times faster than with `modlob` depending if we use the truncation or the log-transformation.

As second example we consider SPX500 options in October 2010. The calibration on equally weighted volatilities would be even more unstable without removing too small option prices from the data. With the cleaning up procedure, the calibration results are very stable and slightly faster than the calibration on the weighted price error measure (Table 8 against Table 9).

Table 8.: Heston calibration under the volatility error measure for options on SPX500 in October 2010.

Method	v_0	κ	θ	σ	ρ	RMSE	Time(s)
Guess	0.0736	1.303	0.0817	0.214	-0.755	N/A	N/A
`globsim`	0.0718	1.542	0.0762	0.582	-0.352	4.907E-3	0.76
`modlob`	0.0718	1.542	0.0762	0.582	-0.352	4.907E-3	0.36
`modlob` Log	0.0718	1.542	0.0762	0.582	-0.352	4.907E-3	0.81
`globsim` Flinn	0.0718	1.542	0.0762	0.582	-0.352	4.907E-3	0.13
`modsim` Filon	0.0718	1.542	0.0762	0.582	-0.352	4.907E-3	0.27

Again, the price error measure makes the problem better behaved (we could even not filter out any option if desired as the inverse vega cap is going to minimize the impact of very small option prices in the calibration). The adaptive Flinn method is three to five times faster than `modlob`.

Table 9.: Heston calibration under the price error measure for options on SPX500 in October 2010.

Method	v_0	κ	θ	σ	ρ	RMSE	Time(s)
Guess	0.0736	1.303	0.0817	0.214	-0.755	N/A	N/A
`globsim`	0.0714	1.559	0.0774	0.499	-0.358	4.244E-2	0.90
`modlob`	0.0714	1.559	0.0774	0.499	-0.358	4.244E-2	0.43
`modlob` Log	0.0714	1.559	0.0774	0.499	-0.358	4.244E-2	0.83
`globsim` Flinn	0.0714	1.559	0.0774	0.499	-0.358	4.244E-2	0.15
`modsim` Filon	0.0714	1.559	0.0774	0.499	-0.358	4.244E-2	0.26

7. Conclusion

We have described a simple adaptive Filon method with better performance against accuracy behavior than popular alternatives to price options under the Heston model. It is particularly interesting in the context of model calibration where many options of different strike but same maturity are priced.

Being adaptive, it does not suffer from having to choose a non obvious parameter value, typically the number of points of non adaptive quadratures, or the truncation level for the Cos method.

The technique can easily be transposed to any stochastic volatility model that possesses an analytical characteristic function.

References

Andersen, L. B. and Piterbarg, V. V. (2010) *Interest Rate Modeling, Volume I: Foundations and Vanilla Models*, (Atlantic Financial Press London).

Bates, D. S. (1996) Jumps and stochastic volatility: Exchange rate processes implicit in Deutsche Mark options, *Review of financial studies*, 9(1), pp. 69–107.

Carr, P. and Madan, D. (1999) Option valuation using the fast Fourier transform, *Journal of Computational Finance*, 2(4), pp. 61–73.

Chase, S. M. and Fosdick, L. D. (1969) An algorithm for Filon quadrature, *Communications of the ACM*, 12(8), pp. 453–457.

Christoffersen, P., Heston, S. and Jacobs, K. (2009) The shape and term structure of the index option smirk: Why multifactor stochastic volatility models work so well, *Management Science*, 55(12), pp. 1914–1932.

Dickinson, A. S. (2011) Numerical Approximation of Option Premia in Displaced-Lognormal Heston Models, *Available at SSRN 1833438*.

Espelid, T. O. (2003) Doubly adaptive quadrature routines based on Newton–Cotes rules, *BIT Numerical Mathematics*, 43(2), pp. 319–337.

Fang, F. and Oosterlee, C. W. (2008) A novel pricing method for European options based on Fourier-cosine series expansions, *SIAM Journal on Scientific Computing*, 31(2), pp. 826–848.

Filon, L. N. G. (1928) On a quadrature formula for trigonometric integrals. In: *Proc. Roy. Soc. Edinburgh*, Vol. 49, pp. 38–47.

Flinn, E. (1960) A modification of Filon's method of numerical integration, *Journal of the ACM (JACM)*, 7(2), pp. 181–184.

Forde, M., Jacquier, A. and Lee, R. (2012) The small-time smile and term structure of implied volatility under the Heston model, *SIAM Journal on Financial Mathematics*, 3(1), pp. 690–708.

Gander, W. and Gautschi, W. (2000) Adaptive quadrature revisited, *BIT Numerical Mathematics*, 40(1), pp. 84–101.

Hai-Ming, Z. and Xiao-Fei, C. (2001) Self-adaptive Filon's integration method and its application to computing synthetic seismograms, *Chinese Physics Letters*, 18(3), p. 313.

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

Jäckel, P. (2013) Let's be rational. , `http://www.pjaeckel.webspace.virginmedia.com/LetsBeRational.pdf` (accessed 03-April-2014).

Joshi, M. and Yang, C. (2011) Fourier Transforms, Option Pricing and Controls, *Option Pricing and Controls (October 9, 2011)*.

Kahalé, N. (2004) An arbitrage-free interpolation of volatilities, *Risk*, 17(5), pp. 102–106.

Kahl, C. and Jäckel, P. (2005) Not-so-complex logarithms in the Heston model, *Wilmott magazine*, 19(9), pp. 94–103.

Kilin, F., Accelerating the calibration of stochastic volatility models. (2007) , Technical report, CPQF Working Paper Series.

Le Floc'h, F. (2013) Fourier Integration and Stochastic Volatility Calibration, *Available at SSRN 2362968*.

Le Floc'h, F. (2015) Volatility Derivatives Practical Notes, *Available at SSRN 2620166*.

Lewis, A. (2001) A simple option formula for general jump-diffusion and other exponential Lévy processes, *Available at SSRN 282110*.

Lord, R. and Kahl, C. (2006) Why the rotation count algorithm works, *Available at SSRN 921335*.

Lord, R. and Kahl, C. (2007) Optimal Fourier inversion in semi-analytical option pricing, *SSRN papers.ssrn.com/abstract=921336*.

Lyness, J. N. (1969) Notes on the adaptive Simpson quadrature routine, *Journal of the ACM (JACM)*, 16(3), pp. 483–495.

Lyness, J. (1970) Algorithm 379: Squank (Simpson Quadrature used adaptivitynoise killed)[D1], *Communications of the ACM*, 13(4), pp. 260–262.

Moré, J. J. (1978) The Levenberg-Marquardt algorithm: implementation and theory. *Numerical analysis*, pp. 105–116 (Springer).

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

Storn, R. and Price, K. (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *Journal of global optimization*, 11(4), pp. 341–359.

Tranter, C. J. (1951) Integral transforms in mathematical physics, .

Tuck, E. (1967) A simple" Filon-trapezoidal" rule, *Mathematics of Computation*, 21(98), pp. 239–241.

Appendix A. Example code

For illustration purpose, we detail here Octave code (also working with Matlab) for pricing vanilla options under the Heston model using the adaptive Flinn quadrature. It relies on the adaptive Simpson `modsim` available on Espelid website.

Note that the code is not optimized at all. Our performance tests are run with a more optimized implementation in the Go language.

Listing 1: Matlab/Octave code for the Heston and Black characteristic functions as used in the Lewis formula

```
function [value, derivative] = hestonLewisLog(T,kappa,theta,v0,omega,rho, z)
  alphahat = -omega*omega*(z .*z+0.25); alphahat_d = -2*omega*omega*z;
  beta = kappa-0.5*omega*rho -omega*rho*z*1i; beta_d = -omega*rho*1i;
  Dsq = beta .*beta - alphahat; Dsq_d = 2*beta_d .*beta - alphahat_d;
  D = sqrt(Dsq); D_d = 0.5 * Dsq_d ./ D;
  flag = abs(real(alphahat)) < 1e-4*abs(beta).^2;
  betamD = flag .*(alphahat ./ (beta + D)) + (!flag).*(beta - D);
  betamD_d = flag .*((alphahat_d - alphahat .*(beta_d+D_d) ./(beta+D)) ...
  ./ (beta + D)) + (!flag) .*( beta_d - D_d);
  G = betamD ./ (beta + D);
  G_d = (betamD_d - betamD .*(beta_d+D_d) ./(beta+D)) ./ (beta + D);
  eDT = exp(-D * T); eDT_d = -D_d .* T .* eDT;
  oneGeDT = 1.0 - G .*eDT; oneGeDT_d = -G .*eDT_d - G_d .*eDT;
  logv = oneGeDT ./ (1 - G);
  logv_d = (oneGeDT_d + oneGeDT .* G_d ./ (1-G)) ./ (1 - G);
  logv_d = logv_d ./ logv; logv = log(logv);

  Afactor = kappa*theta/(omega*omega);
  A = (betamD*T - 2*logv) * Afactor; A_d = (betamD_d*T - 2*logv_d) * Afactor;
  Bfactor = 1.0/omega^2;
  Bratio = (1.0 - eDT) ./ oneGeDT;
  Bratio_d = (-eDT_d - Bratio .*oneGeDT_d) ./ oneGeDT;
  B = Bratio .* betamD * Bfactor;
  B_d = (Bratio_d .*betamD + betamD_d .*Bratio) * Bfactor;
  value = A + B*v0; derivative = A_d + B_d*v0;
end

function [phi, phi_d] = hestonLewisCF(T, kappa, theta, v0, omega, rho, u)
  [arg, arg_d] = hestonLewisLog(T, kappa, theta, v0, omega, rho, u);
  phi = exp(arg); phi_d = arg_d .* phi;
  x = -u .*u - 0.25; # for the Black variate with variance v0*T
  logPhib = 0.5 * v0 * T * x; logPhib_d = -v0 * T * u;
  phib = exp(logPhib); phib_d = phib .*logPhib_d;
  phi -= phib; phi_d -= phib_d;
end

function [c,s, c_d, s_d]=integrandLewisCosSin(T,kappa,theta,v0,omega,rho, x)
  [phi,phi_d] = hestonLewisCF(T,kappa,theta,v0,omega,rho, x);
  frac = 1.0 ./ (0.25 + x .*x);
  c = real(phi) .* frac; s = imag(phi) .* frac;
  c_d = (real(phi_d) - real(phi) .*frac*2 .* x) .* frac;
  s_d = (imag(phi_d) - imag(phi) .*frac*2 .* x) .* frac;
end
```

Listing 2: Flinn integration

```
function integral = integrateFlinn(A, t)
#A has 2n rows of 5 columns: x, f(x), g(x), f'(x), g'(x)
  integral = 0.0;
  for i = 1+ 2*(0:(rows(A)/2-1))
    c0 = A(i,:); c1 = A(i+1,:); c2 = A(i+2,:);
    x0 = c0(1); fcos0 = c0(2); fsin0 = c0(3); fpcos0 = c0(4); fpsin0 = c0(5);
    x1 = c1(1); fcos1 = c1(2); fsin1 = c1(3); fpcos1 = c1(4); fpsin1 = c1(5);
    x2 = c2(1); fcos2 = c2(2); fsin2 = c2(3); fpcos2 = c2(4); fpsin2 = c2(5);
    h = (x2 - x0) * 0.5; theta = t .* h; theta2 = theta.^2;
    if abs(theta) <= 0.8
      M = theta .* (-16.0/105.0 + theta2 .*(8.0/945+theta2 .* (-2.0/10395+theta2 .*
          (1.0/405405.0 - theta2 /48648600.0))));
      N = 16.0/15.0 + theta2 .* (-8.0/105+theta2.* (2.0/945 + theta2 .* (-1/31185.0
          + theta2 / 3243240.0)));
      P = -1.0/15.0 + theta2 .* (2.0/105+theta2 .* (-1.0/315 + theta2 .* (2.0/7425+
         theta2 *(-62.0/4729725.0))));
      Q = theta .* (-8/105.0 + theta2 .*(16.0/945+theta2 .* (-104.0/51975+theta2 .*
          (256.0/2027025-theta2 * 16.0/3274425))));
      R = 14.0/15.0 + theta2 .* (-16.0/105.0+theta2 .* (22.0/945+theta2
          .*(-304.0/155925+theta2 * (268.0/2837835))));
      S = theta .* (19.0/105 + theta2 .* (-2.0/63+theta2 .* (1.0/275+theta2 .*
         (-2.0/8775+theta2 * 34.0/3869775))));
    else
      sint = sin(theta); cost = cos(theta);
      theta4 = theta2 .^2; theta6 = theta2 .* theta4;
      M = (16*theta .* (15-theta2) .* cost + 48 * (2*theta2-5).* sint) ./ theta6;
      N = (16*theta .* (3-theta2) .* sint - 48*theta2 .*cost) ./ theta6;
      P = (2*theta .* (theta2-24) .* sint .* cost + 15*(theta2-4).* cost .* cost +
          theta4 - 27*theta2 + 60) ./ theta6;
      Q = (2 * (theta .*(12-5*theta2) + 15*(theta2-4) .*sint .*cost + 2 .* theta .*
           (24-theta2) .* cost.^2)) ./ theta6;
      R = (2 * (theta .* (156-7*theta2) .* sint .* cost + 3 * (60-17*theta2) .*
          cost .* cost - 15*(12-5 .* theta2))) ./ theta6;
      S = (theta .*(theta4+8*theta2-24) + theta .* (7*theta2-156) .* cost .* cost +
           3*(60-17 .* theta2) .* sint .* cost)./ theta6;
    end
    st0 = sin(t * x0); st1 = sin(t * x1); st2 = sin(t * x2);
    ct0 = cos(t * x0); ct1 = cos(t * x1); ct2 = cos(t * x2);

    s2n = 0.5 * fsin0 .* st0 + 0.5 * fsin2 .* st2;
    sp2n = 0.5 * fpsin0 .* ct0 + 0.5 * fpsin2 .* ct2;
    s2nm1 = fsin1 .* st1; sp2nm1 = fpsin1 .* ct1;
    integral += h .* (S .* (-fsin2 .* ct2+fsin0 .* ct0) + h .* P .* (fpsin2 .*  ...
      st2-fpsin0 .* st0) + R .*s2n - h .*Q .* sp2n + N .* s2nm1 - h .* M .*sp2nm1);

    c2n = 0.5 * fcos0 .* ct0 + 0.5 * fcos2 .* ct2;
    cp2n = 0.5 * fpcos0 .* st0 + 0.5 * fpcos2 .* st2;
    c2nm1 = fcos1 .* ct1; cp2nm1 = fpcos1 .* st1;
    integral += h .* (S .*(fcos2 .*st2-fcos0 .*st0) + h .*P .*(fpcos2 .* ...
      ct2-fpcos0 .*ct0) + R .*c2n + h .*Q .*cp2n + N .*c2nm1 + h .*M .*cp2nm1);
  end
end

function result=uniformIntegral(truncation, N, strike, forward, T,kappa,theta,v0,
    omega,rho)
  k = log(strike/forward);
  x = truncation/N*(0:N);
  [c, s, c_d, s_d]=integrandLewisCosSin(T,kappa,theta,v0,omega,rho, x);
  M = [];
  M(:,1) = x'; M(:,2) = c'; M(:,3) = s'; M(:,4) = c_d'; M(:,5) = s_d';
  result = integrateFlinn(M,k);
end

function [c,s] =integrandLewisCosSinCached(T,kappa,theta,v0,omega,rho, x)
  global A #A has 5 columns: x, f(x), g(x), f'(x), g'(x)
  [c, s, c_d, s_d]=integrandLewisCosSin(T,kappa,theta,v0,omega,rho, x);
  xt = x; ct = c; st = s; c_dt=c_d; s_dt=s_d;
  if columns(x) > 1 #quadrature processes in columns, we want rows
    xt = x'; ct = c'; st = s'; c_dt = c_d'; s_dt = s_d';
  end
  i = rows(A)+1;  j = i+rows(xt)-1;
  A(i:j,1) = xt;  A(i:j,2) = ct; A(i:j,3)=st; A(i:j,4)=c_dt; A(i:j,5)=s_dt;
end

function result=adaptiveFlinn(truncation, strike, forward, T,kappa,theta,v0,omega,
    rho)
global A = []
  integrandCos = @(x) nthargout(1, @integrandLewisCosSinCached, T,kappa,theta,v0,
      omega,rho, x);
  integrandSin = @(x) nthargout(2, @integrandLewisCosSinCached, T,kappa,theta,v0,
      omega,rho, x);
  value = modsim(integrandCos,0, truncation,1e-8);
  value = modsim(integrandSin,0,truncation,1e-8);
  A = unique(A,"rows"); #poor's man hashtable
  sortrows(A,1);
  result = integrateFlinn(A, log(strike/forward))
end
```

Listing 3: Price options with the Lewis integral with Black control variate.

```
function [priceCall,pricePut]=price(strike, fwd, df, v0, T, integral)
  d1 = 1.0/sqrt(v0*T)*log(fwd/strike) + 0.5*sqrt(v0*T);
  blackCall = fwd*normcdf(d1) - strike*normcdf(d1-sqrt(v0*T));
  priceCall = df*(-sqrt(strike * fwd) / pi * integral + blackCall);
  pricePut = -(fwd-strike)*df + priceCall;
end
```

Listing 4: Example use

```
v0=0.826; kappa=0.254; theta=0.320; omega=0.344; rho=-0.557; T=0.0182;
strike=1400; forward=1000; truncation=44.123129; format long;
integral = uniformIntegral(truncation,100, strike, forward, T,kappa,theta,v0,omega,
    rho)
[priceCall,pricePut]=price(strike, forward, 1.0, v0, T, integral)
global A = [];
integralA = adaptiveFlinn(truncation, strike, forward, T,kappa,theta,v0,omega,rho)
[priceCall,pricePut]=price(strike, forward, 1.0, v0, T, integralA)
```