Random grids

In the context of a stochastic local volatility model, ***Jesper Andreasen*** *and* ***Brian Huge*** *present a numerical solution scheme that achieves full consistency between calibration, finite difference solution and Monte Carlo simulation. The method is based on a fully implicit finite difference scheme for the model*

Derivatives models are generally specified in continuous form as a stochastic differential equation (SDE), and implementation of a model will typically involve a number of different discrete approximations of the SDE. For example, an implementation of the Heston (1993) model might have calibration to European-style option prices via numerical inversion of discrete Fourier transforms, backward pricing of exotics handled in a Craig-Sneyd (1988) finite difference scheme, and Monte Carlo pricing using the Andersen (2006) method for simulation. In this case, the different numerical schemes will only be fully consistent with each other in the limit when the number of Fourier steps, the number of time and spatial steps in the finite difference grid, and the number of time steps in the Monte Carlo all tend to infinity and the numerical schemes converge to the continuous time and space solution.

In this article, we present an approach that achieves full discrete consistency between calibration, finite difference solution and Monte Carlo simulation. A continuous time stochastic model has a backward partial differential equation (BPDE) associated with it. We derive a discrete model based on a finite difference scheme for this BPDE. We term this scheme the backward finite difference (BFD) scheme, which we take as our base model.

For calibration purposes, we develop a forward finite difference (FFD) scheme that is dual to and fully consistent with the BFD scheme. It is important to stress that the FFD scheme is not a direct discretisation of the forward (Fokker-Planck) partial differential equation (FPDE) of the continuous time model and that by construction our FFD scheme eliminates the need for specification of the non-trivial boundary conditions that are normally associated with numerical solution of FPDEs (see Lucic, 2008).

Next, we use results in Nabben (1999) to devise an efficient algorithm for calculation of the transition probabilities implicit in the BFD scheme, which is then used for simulating the model in a way fully consistent with the BFD scheme. The numerical work associated with identifying the transition probabilities in the BFD scheme is equivalent to numerical solution of one BFD scheme.

As calibration, backward finite difference solution and simulation are based on the same discretisation, the prices generated by the model are the same, up to Monte Carlo noise, regardless of which numerical scheme is used. Our implementation methodology is presented in the context of a stochastic local volatility model but is applicable to a variety of models.

The rest of the article is organised as follows: the following section describes the stochastic local volatility model and its associated forward and backward partial differential equations (PDEs), the third section introduces the backward and forward finite difference methods, and in the fourth section we present the simulation algorithm. In the fifth section, we describe our implementation and give numerical examples. The article is rounded off with a section that discusses possible extensions and a conclusion.

The model and the PDEs

For simplicity we assume zero rates. We let s be the price of the underlying stock and assume it evolves according to a stochastic local volatility model with zero correlation between spot and volatility:

$$\begin{aligned} ds(t) &= \sqrt{z(t)}\sigma(t,s(t))dW(t) \\ dz(t) &= \theta(1-z(t))dt + \varepsilon z(t)^{\gamma} dZ(t), \quad z(0)=1 \\ dW(t)\cdot dZ(t) &= 0 \end{aligned} \tag{1}$$

where W, Z are independent Brownian motions under the risk-neutral measure.

The FFD equations used for calibration can still be derived for the case of non-zero correlation between underlying stock and its volatility. However, as it stands, our simulation methodology cannot directly be applied for the non-zero correlation case. We will discuss this in a subsequent section.

Equation (1) leads to the BPDE:

$$\begin{aligned} 0 &= \frac{\partial V}{\partial t} + D_x V + D_y V \\ D_x &= \frac{1}{2}\sigma^2 y \frac{\partial^2}{\partial x^2} \\ D_y &= \theta(1-y)\frac{\partial}{\partial y} + \frac{1}{2}\varepsilon^2 y^{2\gamma}\frac{\partial^2}{\partial y^2} \end{aligned} \tag{2}$$

where $V(t, x, y)$ is the price of a claim at time t with current stock price $s(t) = x$ and volatility $z(t) = y$. The boundary conditions are defined by the payout of the claim in question.

The joint density (or Green's function) $q(t, x, y) = \Pr(s(t) \in dx, z(t) \in dy)$ satisfies the FPDE:

$$
\begin{aligned}
&0 = -\frac{\partial q}{\partial t} + D_x^* q + D_y^* q, \quad q(0,x,y) = \delta\big(x - s(0)\big)\cdot\delta\big(y - z(0)\big) \\
&D_x^* q = \frac{1}{2}\frac{\partial^2}{\partial x^2}\left[\sigma^2 y q\right] \\
&D_y^* q = -\frac{\partial}{\partial y}\left[\theta(1-y)q\right] + \frac{1}{2}\frac{\partial^2}{\partial y^2}\left[\varepsilon^2 y^{2\gamma} q\right]
\end{aligned} \tag{3}
$$

where δ is the Dirac function. The operator pair D_x^*, D_y^* are the adjoint operators of D_x, D_y.

The FPDE (3) can be seen as the dual of the BPDE (2) in the sense that European-style options satisfy (2) but also satisfy:

$$V\big(0,x(0),y(0)\big) = \int\int V(t,x,y)\,q(t,x,y)\,dx\,dy \tag{4}$$

The BPDE is solved backward in time and the solution describes the future prices of a particular derivative for all times, spots and volatility levels, whereas the FPDE is solved forward in time. The solution gives the current marginal densities to all future times, spot and volatility levels, and thereby the current prices of all European-style options on spot and volatility.

The European-style call option prices are given by:

$$c(t,k) = E\left[\big(s(t) - k\big)^+\right] = \int\int (x-k)^+ q(t,x,y)\,dx\,dy \tag{5}$$

Double integration of the FPDE (3) or local time arguments can be used to find the extended Dupire equation (1994) that relates initially observed option prices to the local volatility function:

$$0 = -\frac{\partial c}{\partial t} + \frac{1}{2}E\big[z(t)\big|s(t) = k\big]\sigma(t,k)^2\frac{\partial^2 c}{\partial k^2} \tag{6}$$

where:

$$E\big[z(t)\big|s(t) = k\big] = \frac{\int y q(t,k,y)\,dy}{\int q(t,k,y)\,dy} \tag{7}$$

A typical approach for implementing the model (1) would be to discretise (3) to find the local volatility function from (6) that can then, in turn, be used in a discretisation of the BPDE or in a Monte Carlo simulation of a discretisation of the SDE (1).

There are several problems with this approach. The approximations are not mutually consistent. Specifically, direct application of the same type of finite difference scheme to (2) and (3) would not lead to the same results. Further, and more importantly, naive Euler discretisation of the SDE for Monte Carlo may necessitate very fine time stepping for the Monte Carlo prices to be close to the finite difference prices. Finally, application of a finite difference scheme to the FPDE requires specification of non-trivial boundary conditions along the edges of the grid. The latter is particularly a problem for the discretisation in the z direction for parameter choices when the domain of z is closed and includes $z = 0$ (see Lucic, 2008).

The finite difference schemes

Let $0 = t_0 < t_1 < \ldots$ be a discretisation of the time axis. A finite difference scheme for the backward PDE (2) is the locally one-dimensional (LOD) scheme:

$$
\begin{aligned}
\left(1 - \Delta t\bar{D}_x\right)v\left(t_{h+1/2}\right) &= v\left(t_{h+1}\right) \\
\left(1 - \Delta t\bar{D}_y\right)v\left(t_h\right) &= v\left(t_{h+1/2}\right)
\end{aligned} \tag{8}
$$

where $\Delta t = t_{h+1} - t_h$, and:

$$
\begin{aligned}
\bar{D}_x &= \frac{1}{2}y\sigma^2\delta_{xx} \\
\delta_{xx}f(x) &= \frac{1}{\Delta x^2}\left[f(x+\Delta x) - 2f(x) + f(x-\Delta x)\right]
\end{aligned} \tag{9}
$$

and:

$$
\begin{aligned}
\bar{D}_y &= \theta(1-y)\delta_y + \frac{1}{2}y^{2\gamma}\varepsilon^2\delta_{yy} \\
\delta_y g(y) &= \frac{1_{y<1}}{\Delta y}\left[g(y+\Delta y) - g(y)\right] + \frac{1_{y>1}}{\Delta y}\left[g(y) - g(y-\Delta y)\right] \\
\delta_{yy}g(y) &= \frac{1}{\Delta y^2}\left[g(y+\Delta y) - 2g(y) + g(y-\Delta y)\right]
\end{aligned} \tag{10}
$$

Note that we use an upwind operator that switches between forward and backward approximations of the first-order derivative with respect to y depending on the sign of the drift of y.

To close the system (8), boundary conditions at the edges of the grid have to be defined. Here, setting the second derivative to zero, that is, $\delta_{xx}v = 0$ and $\delta_{yy}v = 0$, can be used. For $A_x \equiv 1 - \Delta t\bar{D}_x$, we have:

$$
A_x = \begin{bmatrix}
1 & 0 & & & & \\
-a_1 & 1+2a_1 & -a_1 & & & \\
 & -a_2 & 1+2a_2 & -a_2 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & -a_{m-2} & 1+2a_{m-2} & -a_{m-2} \\
 & & & & 0 & 1
\end{bmatrix}
$$

$$a_i = \frac{1}{2}\frac{\Delta t}{\Delta x^2}y\sigma(t,x_i)^2, \quad x_i = x_0 + i\Delta x \tag{11}$$

and for $A_y \equiv 1 - \Delta t\bar{D}_y$ we have:

$$
A_y = \begin{bmatrix}
1+\bar{c}_0 & -\bar{c}_0 & & & & \\
-\bar{a}_1 & 1+\bar{b}_1 & -\bar{c}_1 & & & \\
 & -\bar{a}_2 & 1+\bar{b}_2 & -\bar{c}_2 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & -\bar{a}_{n-2} & 1+\bar{b}_{n-2} & -\bar{c}_{n-2} \\
 & & & & -\bar{a}_{n-1} & 1+\bar{a}_{n-1}
\end{bmatrix}
$$

$$
\begin{aligned}
&\bar{c}_0 = \frac{\Delta t}{\Delta y}\theta(1-y_0), \qquad \bar{a}_k = \frac{1}{2}\frac{\Delta t}{\Delta y^2}\varepsilon^2 y_k^{2\gamma} - \frac{\Delta t}{\Delta y}\theta(1-y_k)1_{y_k>1} \\
&\bar{b}_k = \bar{a}_k + \bar{c}_k, \qquad \bar{c}_k = \frac{1}{2}\frac{\Delta t}{\Delta y^2}\varepsilon^2 y_k^{2\gamma} + \frac{\Delta t}{\Delta y}\theta(1-y_k)1_{y_k<1} \\
&\bar{a}_{n-1} = -\frac{\Delta t}{\Delta y}\theta(1-y_{n-1}), \qquad y_k = y_0 + k\cdot\Delta y
\end{aligned} \tag{12}
$$

In (8), the first equation has to be solved for each y and the second equation for each x. As usual, the solution of the tridiagonal matrix equations can be performed in linear time using the *tridag*() algorithm in Press *et al* (1988). The LOD scheme (8) can be seen as an alternating direction implicit extension for two spatial dimensions of the fully implicit finite difference scheme for one-dimensional problems. It is not the most sophisticated finite difference scheme and only achieves $O(\Delta t + \Delta x^2 + \Delta y)$ accuracy. The drop in accuracy in the y direction is due to the upwind operator used for the first derivative of y. The strength of the scheme is that it has strong stability properties that, in conjunction with the upwind operator used for the δ_y, guarantee positivity of the inverse of the operators, that is, (1 –

$\Delta t\bar{D}_x)^{-1} \geq 0$ and $(1 - \Delta t\bar{D}_y)^{-1} \geq 0$. This will be a crucial property for the Monte Carlo scheme that we present in the next section.

As our schemes for calibration, backward finite difference and simulation are mutually consistent, low (formal) order of accuracy and (relatively) poor approximation of the SDE is not critical. We can say that the basis of our modelling is the BFD scheme (8) rather than the SDE (1).

The LOD scheme (8) can be rewritten (in transition form) as:

$$v(t_h) = \left(1 - \Delta t\bar{D}_y\right)^{-1}\left(1 - \Delta t\bar{D}_x\right)^{-1} v(t_{h+1}) \tag{13}$$

Multiplying (13) by a vector p yields[1]:

$$p(t_h)' v(t_h) = p(t_h)'\left(1 - \Delta t\bar{D}_y\right)^{-1}\left(1 - \Delta t\bar{D}_x\right)^{-1} v(t_{h+1}) \tag{14}$$

From (12) we see that if p is chosen to satisfy:

$$\begin{aligned} p(t_0) &= 1_{x=x(0)} \cdot 1_{y=y(0)} \\ \left(1 - \Delta t\bar{D}_y\right)' p(t_{h+1/2}) &= p(t_h) \\ \left(1 - \Delta t\bar{D}_x\right)' p(t_{h+1}) &= p(t_{h+1/2}) \end{aligned} \tag{15}$$

then:

$$p(t_h)' v(t_h) = p(t_{h+1})' v(t_{h+1}) \tag{16}$$

for every h. In particular, for European-style options we have:

$$v(0, s(0), z(0)) = p(t_0)' v(t_0) = p(t_n)' v(t_n) \tag{17}$$

We conclude that p is the discrete Green's function for the BFD scheme (8). Note that (15) is completely consistent with the BFD scheme (8), since it is constructed directly from that equation. The adjoint operators in the FPDE (3) are in (15) replaced by the transpose of the matrix. There is no need to take special care of the boundary conditions at the edges because these are implicitly defined by the backward scheme. We also note that direct application of the LOD scheme to the FPDE (3) would result in a different order of the application of the x and y operators.

Consistent FFD schemes can be developed for any BFD scheme, including schemes that are second-order accurate in time such as, for example, the Craig-Sneyd (1988) scheme. In fact, for any BFD scheme that can be written as:

$$v(t_h) = Hv(t_{h+1}) \tag{18}$$

for some matrix H, we have a consistent FFD scheme that is given by:

$$p(t_{h+1}) = H'p(t_h) \tag{19}$$

Further, we note that the matrix H can be seen as a transition probability matrix, which in the context of Markov chain models would be termed as a discrete time generator matrix. The latter observation is what will be used for developing our Monte Carlo simulation scheme in the following section.

A similar result exists for the continuous time and discrete space case. If the prices solve the backward differential equation:

$$0 = \frac{\partial v}{\partial t} + Lv \tag{20}$$

for some matrix L, then the associated Green's function solves the forward differential equation:

[1] *Here, A' denotes the transpose of a matrix A*

$$0 = -\frac{\partial p}{\partial t} + L'p \tag{21}$$

This can be used for deriving boundary conditions for the FPDE (3) (see Andreasen, 2009).

Multiplying the last equation in (15) by $(x - k)^+$ and summing over x, y leads to the following discrete version of the Dupire equation (6):

$$\begin{aligned} 0 &= -\frac{1}{\Delta t}\left[c(t_{h+1}, x) - c\left(t_{h+\frac{1}{2}}, x\right)\right] \\ &\quad + \frac{1}{2}\sigma(t_h, x)^2 E\left[y(t_{h+1})\middle| x\right]\delta_{xx} c(t_{h+1}, x) \\ c\left(t_{h+\frac{1}{2}}, k\right) &= \sum_x \sum_y (x - k)^+ p\left(t_{h+\frac{1}{2}}, x, y\right) \\ E\left[y(t_{h+1})\middle| x\right] &= \frac{\sum_y y p(t_{h+1}, x, y)}{\sum_y p(t_{h+1}, x, y)} \end{aligned} \tag{22}$$

Equation (22) can be used in conjunction with (15) for calibrating the local volatility surfaces so that we achieve consistency between the calibration scheme and the BFD scheme.

Monte Carlo simulation

As mentioned earlier, the representation in (13) of the BFD scheme can be viewed as a description of the transition probabilities of (x, y) over a single time step, and thereby as a recipe for simulation of (x, y). Over the first half time step we simulate y while x is frozen, over the second half time step we simulate x while y is frozen:

$$v(t_h) = \underbrace{\left(1 - \Delta t\bar{D}_y\right)^{-1}}_{simulate\ y} \cdot \underbrace{\left(1 - \Delta t\bar{D}_x\right)^{-1}}_{simulate\ x} v(t_{h+1}) = A_y^{-1} \cdot A_x^{-1} v(t_{h+1}) \tag{23}$$

The conditional transition probabilities are given by:

$$\begin{aligned} \Pr\left[x(t_{h+1}) = x_j \middle| x(t_h) = x(t_{h+1/2}) = x_i\right] &= \left(A_x^{-1}\right)_{ij} \\ \Pr\left[y(t_{h+1}) = y(t_{h+1/2}) = y_l \middle| y(t_h) = y_k\right] &= \left(A_y^{-1}\right)_{kl} \end{aligned} \tag{24}$$

From (11) we see that:

$$A_x \iota = \iota, \quad \iota = (1, \ldots, 1)' \tag{25}$$

Hence:

$$A_x^{-1}\iota = \iota \tag{26}$$

Further, as $a_j \geq 0$, the matrix A_x is diagonally dominant with positive diagonal and negative off-diagonals, and results in Nabben (1999) and Achdou & Pironneau (2005) can be used to show that:

$$A_x^{-1} \geq 0 \tag{27}$$

Inspecting (12) reveals that the use of upwind operators for the first derivative ensures that:

$$\bar{a}_k, \bar{b}_k, \bar{c}_k \geq 0 \tag{28}$$

and thereby that A_y is diagonally dominant with positive diagonal and negative off-diagonals. This, in turn, ensures that the properties (26) and (27) also are satisfied for A_y.

We conclude that the transition probabilities (24) are indeed non-negative and sum to one. For constant parameters the rows in A_x^{-1}

and A_y^{-1} are Laplace densities. Figure 1 shows an example of this.

Generally, the calculations needed to find the inverse of a matrix are quite time-consuming, of the order $O(N^3)$, for a matrix in $\mathbb{R}^{N\times N}$. However, for a tridiagonal matrix $A \in \mathbb{R}^{N\times N}$, Nabben (1999) shows that the inverse can be represented as:

$$A_{ij}^{-1} = \begin{cases} u_i v_j & i \le j \\ r_j s_i & i > j \end{cases} \tag{29}$$

where the vectors r, s, u, v can be found in linear time, that is, at a computational cost of $O(N)$. The drawback of the Nabben algorithm is that the elements of u, r tend to be very large and the elements of v, s tend to be very small.

Instead, we base our simulation algorithm on Huge (2010), who provides an $O(N)$ algorithm for identifying the vectors $\tilde{x}, \tilde{y}, \tilde{d} \in \mathbb{R}^N$ satisfying:

$$\begin{aligned} &\sum_{j\le i} A_{ij}^{-1} = \tilde{d}_i, \qquad A_{ii}^{-1} = \tilde{x}_i\tilde{y}_i / A_{ii} \\ &A_{ij}^{-1} = A_{i,j+1}^{-1}\cdot\frac{\tilde{x}_j}{\tilde{x}_{j+1}}\cdot\frac{-A_{j+1,j}}{A_{jj}}, \quad j<i \\ &A_{ij}^{-1} = A_{i,j+1}^{-1}\cdot\frac{\tilde{y}_j}{\tilde{y}_{j-1}}\cdot\frac{-A_{j-1,j}}{A_{jj}}, \quad j>i \end{aligned} \tag{30}$$

A C-implementation of the Huge (2010) algorithm is given in figure 2. The algorithm is very similar in structure and numerical complexity to the *tridag*() algorithm in Press *et al* (1988). Computing the vectors $\tilde{x}, \tilde{y}, \tilde{d}$ for all steps in the finite difference grid (8) has a computational cost that is roughly equivalent to solving one finite difference grid.

For the cumulative distribution function:

$$Q_{ij} \equiv \Pr\left[x(t_{h+1}) \le x_j \,\middle|\, x(t_h) = x_i\right] = \sum_{k\le j} A_{ik}^{-1} \tag{31}$$

we have:

$$Q_{ij} = Q_{ii} - \sum_{k=j+1}^{i} A_{ik}^{-1} + \sum_{k=i+1}^{j} A_{ik}^{-1} \tag{32}$$

The results in (30) can be used for identifying each of the elements in (32) by recursion from the diagonal $j = i$.

This leads to the following simulation algorithm:

```
0. Suppose $x(t_h) = x_i$. Set $j = i$.
1. Draw a uniform $\tilde{u} \sim U(0,1)$.
2. If $\tilde{u} \le Q_{ii}$ : while $\tilde{u} < Q_{i,j-1}$ set $j := j-1$. Go to 4.        (33)
3. If $\tilde{u} > Q_{ii}$ : while $\tilde{u} < Q_{i,j-1}$ set $j := j+1$. Go to 4.
4. Set $x(t_{h+1}) = x_j$.
```

Once the vectors $\tilde{x}, \tilde{y}, \tilde{d}$ have been calculated using the Huge algorithm, the actual simulation algorithm is very fast. There are two reasons for this. First, the updating in the while loops in steps two and three is quick due to the recursive nature of (28) and (30). Second, as we start in the diagonal, the number of steps in the while loops will be very limited, say zero, one or two, for most draws of the uniform variable.

Numerical implementation and examples

In our implementation, we use the algorithm of Andreasen & Huge (2011) to interpolate and extrapolate discrete sets of observed European-style option prices to a continuum of arbitrage-consistent European-style option prices before calibration of the dynamic model.

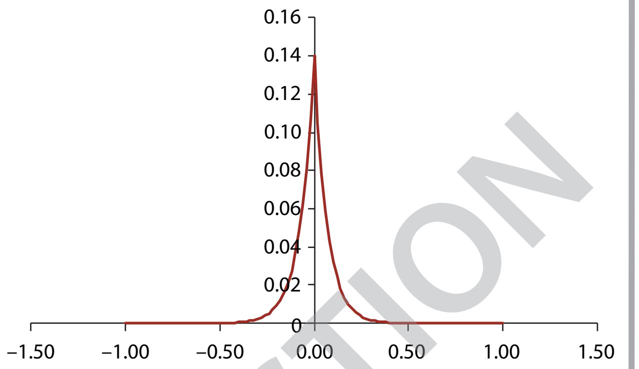
In a conventional model implementation where different approximations of the original SDE are used, it is important to verify the convergence and to control the approximation error. With our approach, this is no longer as important since the calibration, BFD scheme pricing and Monte Carlo simulation all are done within the same discrete model. That said, it is interesting to quantify the inherent discrepancies between the continuous and the discrete models.

To this end, we consider a Heston (1993) model with flat volatility and the parameters:

$$\begin{aligned} &\sigma(t,s) = 0.3\cdot s \\ &\varepsilon = 3, \theta = 1, \gamma = 0.5 \end{aligned} \tag{34}$$

We calculate option prices using the following numerical methods: BFD scheme solution using the LOD scheme (8), Monte Carlo simulation as described in the previous section, that is, (23), and numerical inversion of the Fourier transform, as in Lipton (2002). The option prices calculated with different numerical resolution are reported in table A. We note that the prices gener-

1 A row in the transition matrix



Note: graph shows the values of a row in A_x^{-1} with $a_i = 12$

2 Algorithm for decomposition of tridiagonal matrix

```
//  the vectors a, b, c are the three diagonals of the matrix A
void
hugeDecomp(
    int n, const double* a, const double* b, const double* c,    //  input
    double* x, double* y, double* d)                             //  output
{
    //  dimension check
    if(n<=0) return;
    if(n==1)
    {
        x[0] = 1.0;
        y[0] = 1.0;
        d[0] = 1.0/b[0];
        return;
    }

    //  help
    int i;

    //  fwd
    x[0] = 1.0;
    x[1] = 1.0;
    for(i=2;i<n;++i)
    {
        x[i] = x[i-1] - (a[i-1]/b[i-1])*(c[i-2]/b[i-2])*x[i-2];
    }

    //  bwd
    y[n-1] = 1.0/(x[n-1] - (a[n-1]/b[n-1])*(c[n-2]/b[n-2])*x[n-2]);
    y[n-2] = y[n-1];
    for(i=n-3;i>=0;--i)
    {
        y[i] = y[i+1] - (a[i+2]/b[i+2])*(c[i+1]/b[i+1])*y[i+2];
    }

    //  set d
    d[0] = x[0]*y[0]/b[0];
    for(i=1;i<n;++i)
    {
        d[i] = -(a[i]/b[i])*(y[i]/y[i-1])*d[i-1] + x[i]*y[i]/b[i];
    }
}
```

A. Pricing on flat parameters

Strike	50%	100%	200%	CPU
Fourier	29.69%	26.56%	29.69%	< 0.001s
FD				
25 × 50 × 12	29.53%	25.46%	29.58%	0.04s
25 × 100 × 25	29.60%	25.78%	29.62%	0.11s
25 × 200 × 50	29.63%	25.88%	29.63%	0.39s
MC				
25 × 50 × 12	29.50%	25.45%	29.59%	0.69s
25 × 100 × 25	29.61%	25.78%	29.64%	0.80s
25 × 200 × 50	29.67%	25.89%	29.67%	1.09s
FD				
100 × 100 × 25	29.63%	26.22%	29.64%	0.35s
1,000 × 100 × 25	29.64%	26.38%	29.65%	2.80s
10,000 × 100 × 25	29.64%	26.40%	29.65%	27.23s
CS 25 × 100 × 25	29.67%	26.51%	29.69%	0.15s

Note: the table reports the implied Black volatilities for five-year European-style options with strikes given as 50%, 100% and 200% of the initial forward priced in a (Heston) model with flat parameters. Results are reported for solution by Fourier inversion, and backward finite difference solution (FD) and Monte Carlo (MC) for different grid sizes. Here 25 × 100 × 25 refers to a grid with a total of 25 *t*-steps, 100 *x*-steps and 25 *y*-steps. The term 'CS' refers to pricing in a second-order accurate Craig-Sneyd scheme. MC prices are generated by 32 randomly seeded batches of 16,384 Sobol paths (see Glasserman, 2003). MC pricing error is approximately 0.02% in implied Black volatility terms. For the FD case, the reported CPU times include one forward sweep (calibration) and one backward sweep (pricing) of the finite difference grid. For the MC case, it includes one forward sweep, one decomposition sweep (MC initialisation) and simulation of 16,384 paths. Hardware is a standard 2.66GHz Intel PC

B. Pricing with stochastic local volatility

	50%	100%	200%	CPU
FD 25 × 200 × 50				
EPS = 0	36.96%	26.73%	19.19%	0.05s
EPS = 1	36.96%	26.73%	19.19%	0.42s
EPS = 2	36.96%	26.73%	19.19%	0.42s
EPS = 3	36.96%	26.72%	19.18%	0.42s
MC 25 × 200 × 50				
EPS = 0	36.94%	26.74%	19.17%	0.68s
EPS = 1	36.90%	26.72%	19.14%	1.20s
EPS = 2	36.95%	26.72%	19.14%	1.20s
EPS = 3	36.99%	26.74%	19.14%	1.20s
FD 1,000 × 200 × 50				
EPS = 0	36.98%	26.75%	19.21%	0.41s
EPS = 1	36.98%	26.75%	19.21%	14.14s
EPS = 2	36.98%	26.75%	19.21%	14.14s
EPS = 3	36.98%	26.74%	19.21%	14.14s

Note: the table reports the implied Black volatilities for five-year European-style options with strikes of 50%, 100% and 200% of the initial forward priced when the model is calibrated to the SX5E equity option market, for different levels of the volatility of variance parameter ε. Here, we use the same terminology for grid sizes as in table A. Monte Carlo (MC) prices are generated the same way as for table A, again with an MC pricing error of approximately 0.02% in implied Black volatility terms. CPU times are also measured in the same way as in table A. The SX5E equity option data is as of July 28, 2010

C. Pricing exotics

Epsilon	0.0	1.0	2.0	3.0	MC error
Variance	0.1891	0.1862	0.1843	0.1854	0.0030
Floored variance	0.2423	0.2466	0.2567	0.2665	0.0027
Straddles	1.8123	1.7155	1.5230	1.3460	0.0089

Note: the table reports prices of three different exotics on the SX5E equity index for different levels of ε. Prices are generated from 16,384 paths in a grid of dimensions 25 × 200 × 25. Market data is as of July 28, 2010

ated by the BFD scheme and Monte Carlo on the same finite difference grid match. But we also note that the convergence is relatively crude in the time domain. This is as expected as the LOD method only achieves $O(\Delta t)$ accuracy.

However, this is of little practical importance when the discrete models are calibrated to the same observed option prices. This is illustrated in table B, where we compare European-style option prices in discrete models with different number of time steps and different levels of ε. The models are all calibrated to the SX5E equity option data as of July 28, 2010.

The default configuration in our implementation uses a finite difference grid with a total size of 25 × 200 × 50 ($t \times x \times y$) steps. As can be seen in tables A and B, a BFD scheme solution on such a grid can be done in approximately 0.4 seconds of CPU time on a standard computer. This includes a forward sweep to calibrate the model and a backward sweep of the grid to price the actual contract. Monte Carlo simulation on the finite difference grid carries an overhead to set up the simulation that is similar to the BFD scheme solution: a forward sweep to calibrate the model and decomposition to identify the vectors $\tilde{x}$, $\tilde{y}$, $\tilde{d}$ at all steps in the grid. The total CPU time for simulation of 16,384 paths on a 25 × 200 × 50 grid is approximately 1.2 seconds. So in this case, roughly 0.8 seconds is spent inside the simulation algorithm described in (33). Profiling our code reveals that roughly 80% of the time spent in the simulation algorithm involves drawing the random numbers in step 1. So in terms of speed, the simulation methodology is, step by step, almost as fast as naive Euler discretisation. Since our algorithm reproduces the exact distribution of the BFD scheme (8), there will, however, be no need for more steps in the simulation than in the BFD scheme.

Though two models produce the same prices for European-style options, they can produce markedly different prices for exotics. To illustrate this, we consider three different exotic options on the SX5E equity index. Let $t_i = i/12$ and define the returns:

$$R_i = \frac{s(t_i)}{s(t_{i-1})} - 1, \quad i = 1,\ldots,36 \tag{35}$$

We consider three different exotic options:

$$\begin{aligned} &\text{Variance:} && U = \sum_i R_i^2 \\ &\text{Floored variance:} && \max(H, U) \\ &\text{Straddles:} && \sum_i |R_i| \end{aligned} \tag{36}$$

Table C reports the prices of these exotics for different levels of volatility of variance ε. We see that the variance contract is almost invariant to the level of ε. The intuition here is that if there are no jumps in the underlying stock, then a contract on the continuously observed variance can be statically hedged by a contract on the logarithm of the underlying stock (see Dupire, 1993). Hence, if European-style option prices are the same in two models with continuous evolution of the stock, then the value of the variance contract should be the same. The floored variance contract, on the other hand, includes an option on the variance and should therefore increase with the volatility of variance parameter ε. Finally, for each period, the forward starting straddle payout is the square root of the realised variance:

$$|R_i| = \sqrt{R_i^2}$$

As the square root is a concave function, we should expect to see the value of the straddle contract decrease with the level of ε. We conclude that the ε dependence of the exotic option prices in table C are in line with what we expect.

Extensions

An easy way to extend the model to the multi-asset case is to use a joint volatility process and correlate the increments of the underlying stocks using a Gaussian copula. Specifically, if $\tilde{u}_i$ is the uniform used for propagating stock i at a given time step, we can set:

$$\tilde{u}_i = \Phi(\tilde{w}_i), \quad i = 1,\dots,l, \quad \tilde{w} = (\tilde{w}_i) = P\xi \tag{37}$$

where $\xi_1, \dots, \xi_l$ are independent with $\xi_i \sim N(0, 1)$, and PP' is a correlation matrix in $\mathbb{R}^{l\times l}$.

The BFD scheme (8) can be extended to include correlation between stock and volatility, in the following way:

$$\begin{aligned}\left[1-\Delta t\bar{D}_x\right]v(t_{h+1/2}) &= \left[1+\frac{1}{2}\Delta t\bar{D}_{xy}\right]v(t_{h+1})\\ \left[1-\Delta t\bar{D}_y\right]v(t_h) &= \left[1+\frac{1}{2}\Delta t\bar{D}_{xy}\right]v(t_{h+1/2})\end{aligned} \tag{38}$$

where:

$$\begin{aligned}\bar{D}_{xy} &= \sigma\varepsilon\rho y^{\gamma+1/2}\delta_{xy}\\ \delta_{xy}f(x,y) &= \frac{f(x+\Delta x, y+\Delta y)-f(x+\Delta x, y-\Delta y)}{4\Delta x\Delta y}\\ &\quad -\frac{f(x-\Delta x, y+\Delta y)-f(x-\Delta x, y-\Delta y)}{4\Delta x\Delta y}\end{aligned} \tag{39}$$

The scheme (38) is unconditionally stable, in the von Neumann sense, and has accuracy of order $O(\Delta t + \Delta x^2 + \Delta y)$. The BDF scheme (38) leads to the dual FFD scheme:

$$\begin{aligned}p(t_0) &= 1_{x=x(0)}\cdot 1_{y=y(0)}\\ \left(1-\Delta t\bar{D}_y\right)' p(t_{h+1/4}) &= p(t_h)\\ p(t_{h+1/2}) &= \left(1+\frac{1}{2}\Delta t\bar{D}_{xy}\right)' p(t_{h+1/4})\\ \left(1-\Delta t\bar{D}_x\right)' p(t_{h+3/4}) &= p(t_{h+1/2})\\ p(t_{h+1}) &= \left(1+\frac{1}{2}\Delta t\bar{D}_{xy}\right)' p(t_{h+3/4})\end{aligned} \tag{40}$$

The FFD scheme (40) is then to be used as the basis for calibration instead of (15).

For simulation, we note that (38) can be rearranged as:

$$\begin{aligned}v(t_h) &= \left[1-\Delta t\bar{D}_y\right]^{-1}\cdot\left[1+\frac{1}{2}\Delta t\bar{D}_{xy}\right]\\ &\quad\cdot\left[1-\Delta t\bar{D}_x\right]^{-1}\cdot\left[1+\frac{1}{2}\Delta t\bar{D}_{xy}\right]v(t_{h+1})\end{aligned} \tag{41}$$

with x or y simulated for the first-order factors, and both for the second order.

The matrix:

$$B \equiv \left[1+\frac{1}{2}\Delta t\bar{D}_{xy}\right] \tag{42}$$

specifies weights that sum to one and link the left-hand side values at state (x, y) to the right-hand side values at the states:

$$\{(x-\Delta x, y-\Delta y), (x-\Delta x, y+\Delta y), (x+\Delta x, y-\Delta y), (x+\Delta x, y+\Delta y)\} \tag{43}$$

Like A_x^{-1}, A_y^{-1}, it can therefore be viewed as a transition matrix for joint transition of (x, y) into one of the states in (43). This suggests simulation according to the (conditional) probabilities:

$$\Pr\left[(x,y) = (x_i, y_j)\right] = B_{ij} \tag{44}$$

The trouble is that some of the entries of B are negative. This can be handled by simulation according to the transition probabilities:

$$\Pr\left[(x,y) = (x_i, y_j)\right] = \frac{|B_{ij}|}{\sum_k\sum_l |B_{kl}|} \tag{45}$$

in combination with a numeraire that over each B-step is updated according to:

$$N := N\,\mathrm{sgn}(B_{ij})\sum_i\sum_j |b_{ij}| \tag{46}$$

if entry ij is simulated. The numeraire will then have to be multiplied on the terminal payout.

Conclusion

We have presented a methodology that achieves full discrete consistency between calibration, backward finite difference pricing and Monte Carlo simulation, in the context of a stochastic local volatility model. The methods extend to the multi-asset case and to the case of non-zero correlation between the underlying and the volatility process, as well as to other model types. ■

Jesper Andreasen is head of and Brian Huge is chief analyst in the quantitative research department at Danske Markets in Copenhagen. They would like to thank two anonymous referees for helpful comments and suggestions. Email: kwant.daddy@danskebank.com, brno@danskebank.com

References

Achdou Y and O Pironneau, 2005
Computational methods for option pricing
SIAM Frontiers in Applied Mathematics

Andersen L, 2006
Efficient simulation of the Heston process
Working paper, Bank of America

Andreasen J, 2009
Planck-Fokker boundary conditions
Working paper, Danske Markets

Andreasen J and B Huge, 2011
Volatility interpolation
Risk March, pages 86–89

Craig I and A Sneyd, 1988
An alternating-direction implicit scheme for parabolic equations with mixed derivatives
Computers and Mathematics with Applications 16(4), pages 341–350

Dupire B, 1993
Model art
Risk July, pages 118–121

Dupire B, 1994
Pricing with a smile
Risk January, pages 18–20

Glasserman P, 2003
Monte Carlo methods in financial engineering
Springer

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

Huge B, 2010
The inverse of a tridiagonal matrix
Working paper, Danske Markets

Lipton A, 2002
The vol smile problem
Risk February, pages 61–65

Lucic V, 2008
Boundary conditions for computing densities in hybrid models via PDE methods
Working paper, Barclays Capital

Nabben R, 1999
On decay rates of tridiagonal and band matrices
SIAM Journal on Matrix Analysis and Applications 20, pages 820–837

Press W, W Vetterling, S Teukolsky and B Flannery, 1988
Numerical recipes in C: the art of scientific computing
Cambridge University Press