Finite Difference Techniques for Arbitrage Free SABR

Fabien Le Floc'h* and Gary Kennedy†

* *Calypso Technology, 106 rue de La Boétie, 75008 Paris*
† *Clarus Financial Technology, London*

(v2.0 released August 2013)

Abstract *This paper presents various finite difference schemes applied to the SABR arbitrage free density problem with a focus on stability and speed.*

Key Words: stochastic volatility, SABR, TR-BDF2, Crank-Nicolson, finite difference, finance

1. Introduction

It is now well known that original SABR analytic formula from (Hagan *et al.*, 2002) used to compute option implied volatility is not arbitrage free as the probability density can become negative for low strikes and long maturities. Given the current low rates environment, many authors have proposed various improvements to the original formula (Oblój, 2008; Johnson and Nonas, 2009; Paulot, 2009; Benaim *et al.*, 2008). A single step finite difference method is proposed in (Andreasen and Huge, 2011) which leads to an arbitrage free 'SABR-like' model. Whilst the approach from Andreasen and Huge converges for short maturities to the original SABR analytic formula, it is (deliberately) different for longer expiries, even at the money.

Hagan recently proposed a new arbitrage free SABR solution, based on a finite difference discretisation of the probability density in (Hagan *et al.*, 2013). This approach provides a solution very close to the original SABR analytic formula, well known and widely used, while being arbitrage free, and thus allowing pricing with low rates. The authors use a Crank-Nicolson time-stepping scheme, which is known to have oscillation issues (Duffy, 2004; Giles and Carter, 2006) as it is only A-stable but not L-stable (LeVeque, 2007). We will show that this issue arises in the context of SABR pricing, and propose alternative schemes that are not very well known in computational finance, and yet are effective on this problem. One such scheme is TR-BDF2, used in the semiconductor industry as well as more generally in computational physics (Bank *et al.*, 1985; Bathe and Baig, 2005; Edwards *et al.*, 2011; Flavell *et al.*, 2013). Another scheme is due to Lawson and Swayne; whilst somewhat obscure, it is simple and effective on this problem (Lawson and Swayne, 1976).

Speed and accuracy were key ingredients in popularising the original SABR formula. Given that for a 30 year cap on a 3M LIBOR, there are potentially 119 PDEs to solve, we will focus our attention on the performance of the proposed schemes, as well as to what extent the discretisation grid can be reduced in size.

Correspondence Address: Calypso Technology, 106 rue de La Boétie, 75008 Paris. Email: fabien_lefloch@calypso.com

ISSN: Print/ISSN Online/YY/issnum1stp–ppcnt
DOI:

2. Arbitrage Free SABR

In (Hagan *et al.*, 2013), pricing with SABR parameters α, β, ρ, ν and forward f at τ_{ex} relies on the solution of a PDE on the probability density Q:

$$\frac{\partial Q}{\partial T}(T,F) = \frac{\partial^2 M(T,F)Q(T,F)}{\partial F^2} \text{ and } \begin{cases} \frac{\partial Q_L}{\partial T}(T) = \lim_{F\to F_{min}} \frac{\partial M(T,F)Q(T,F)}{\partial F} \\ \frac{\partial Q_R}{\partial T}(T) = \lim_{F\to F_{max}} \frac{\partial M(T,F)Q(T,F)}{\partial F} \end{cases} \tag{1}$$

with

$$M(T,F) = \frac{1}{2}D^2(F)E(T,F),\ E(T,F) = e^{\rho\nu\alpha\Gamma(F)T},\ \Gamma(F) = \frac{F^\beta - f^\beta}{F-f} \tag{2}$$

$$D(F) = \sqrt{\alpha^2 + 2\alpha\rho\nu y(F) + \nu^2 y(F)^2}F^\beta,\ y(F) = \frac{F^{1-\beta} - f^{1-\beta}}{1-\beta} \tag{3}$$

It is suggested that the lower boundary F_{min} for the standard SABR model is placed at or near zero. However, the finite difference grid described in Appendix C of their paper starts at $F_0 = F_{min} - \frac{h}{2}$, where, h is the asset forward discretisation step size, potentially requiring the evaluation of functions not well-defined for negative values of F_0. Fortunately, only the product M_0Q_0 is used in the discretisation of equation (1) and it is entirely defined by M_1Q_1 because of the mirror-like boundary condition (imposed at the fictitious point F_0):

$$M_0Q_0 + M_1Q_1 = 0 \tag{4}$$

As long as $M_0 \neq 0$, M_0Q_0 will take the value $-M_1Q_1$. For example, we can use $|F_0|$ to compute M_0 and this will result in a symmetry around F_{min}.

Another alternative would be to place the grid so that $F_0 = F_{min}$ and use boundary condition $Q_0 = 0$ there. The probability of absorption Q_L could then be evaluated using an forward finite difference first derivative estimate. This would result in the exact same equation as (C.10a) of their paper and the scheme would still be moment preserving. However this comes at a cost of a slight loss of accuracy as, effectively, the derivative will be estimated using $Q_1 = Q(h)$ instead of $Q_1 = Q(\frac{h}{2})$.

The formula for Γ is also undefined for $F = f$, in which case we just use $\Gamma(f) = \frac{\partial C}{\partial F}(f)$.

3. Change of Variable

One practical difficulty that arises with the Arbitrage Free PDE described in equations (1) is the choice of F_{max}. Hagan *et al.* (2013) proposes a formula to estimate the required F_{max}, that, for some parameters is not suitable. It can be very large for long term deals and as a consequence the discretization requires a very large number of points to obtain an acceptable accuracy. The technique becomes inefficient. A change of variable is proposed in (Hagan, 2013) as a remedy:

$$z(F) = \int_f^F \frac{dF'}{D(F')} \tag{5}$$

This leads to a PDE in $\theta(z) = Q(T,F(z))D(F(z)) = Q(T,F(z))C(z)$ with $C(z) = D(F(z))$:

$$\frac{\partial \theta}{\partial T}(T,z) = \frac{1}{2}\frac{\partial}{\partial z}\left\{\frac{1}{C(z)}\frac{\partial C(z)E(T,z)\theta(T,z)}{\partial z}\right\} \text{ and } \begin{cases} \theta(T,z) = 0 \text{ as } z \to z^- = z(F_{min}) \\ \theta(T,z) = 0 \text{ as } z \to z^+ = z(F_{max}) \end{cases} \tag{6}$$

with

$$y(z) = \frac{\alpha}{\nu}\left[\sinh(\nu z) + \rho(\cosh(\nu z) - 1)\right] \tag{7}$$

$$F(y) = \left[f^{1-\beta} + (1-\beta)y\right]^{\frac{1}{1-\beta}} \tag{8}$$

The probability at the boundaries accumulates according to:

$$\frac{\partial P_L}{\partial T}(T) = \lim_{z\to z^-} \frac{1}{2}\frac{1}{C(z)}\frac{\partial C(z)E(T,z)\theta(T,z)}{\partial z} \tag{9}$$

$$\frac{\partial P_R}{\partial T}(T) = \lim_{z\to z^+} -\frac{1}{2}\frac{1}{C(z)}\frac{\partial C(z)E(T,z)\theta(T,z)}{\partial z} \tag{10}$$

As a result, the effect of the diffusion term D has almost been cancelled, and the probability density θ is closer to a Gaussian in z. z^+ and z^- are then naturally chosen to be $\pm n_{sd}\sqrt{\tau_{ex}}$, corresponding to n_{sd} standard deviations above and below the forward located at $z = 0$ (taking care of truncating at the barrier $F = 0$ if necessary). Figure 1 shows that the probability density will be computed with a high concentration of points around the forward, and a much lower one near the upper boundary.

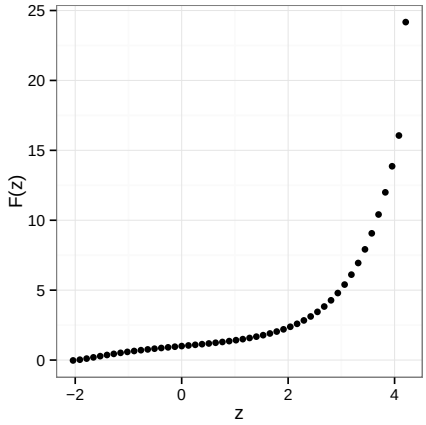


Figure 1.: F(z) with $\alpha = 35\%, \beta = 0.25, \rho = -10\%, \nu = 100\%, \tau_{ex} = 1$

For some extreme SABR parameters, the change of variable allows high accuracy with a small number of points. The uniform discretization of Q in F can require approximately 1000 times more points to reach a similar accuracy (Table 1). The number of points used can not

Uniform discretization of Q in F					Discretization of θ in z				
F_{max}	Points	Steps	Price	Vol	n_{sd}	Points	Steps	Price	Vol
5	10	5	0.65010	87.205	3	10	5	0.79848	198.504
50	100	10	0.78769	155.773	3	100	10	0.79853	199.148
500	1000	20	0.79782	191.658	4	100	20	0.79847	198.338
5000	10000	160	0.79835	196.930	10	10000	160	0.79845	198.134

Table 1.: Price by the Lawson-Swayne method without and with variable change for extreme SABR parameters: $\alpha = 100\%, \beta = 0.30, \rho = 90\%, \nu = 100\%, \tau_{ex} = 10, f = 1$

be too small: the forward should not be on the boundary. This restriction is much stricter for the uniform discretization of Q than for the discretization in the changed variable θ.

4. Discretization of the the PDE in θ

Let's define for $j = 1, ..., J$:

$$z_j = z^- + jh \ , \ \hat{y}_j \ = y(z_j - \frac{1}{2}h) \ , \ \hat{F}_j = F(\hat{y}_j) \tag{11}$$

$$\hat{C}_j = D(\hat{F}_j) \ , \ \hat{\Gamma}_j \ = \frac{\hat{F}_j^\beta - f^\beta}{\hat{F}_j - f} \ , \ \hat{E}_j(T) = e^{\rho\nu\alpha\hat{\Gamma}_j T} \tag{12}$$

We also define for $n = 0, ..., N-1$

$$t_n = n\delta \text{ with } \delta = \frac{\tau_{ex}}{N} \tag{13}$$

$$\theta_j^n = \theta(z_j, t_n) \tag{14}$$

To preserve the zero-eth and first moment of F, the PDE (Equation 6) is discretized in z as (Hagan, 2013):

$$\frac{\partial\theta}{\partial T}(z, t_n) = \mathcal{L}_j^n \theta(z, t_n) \tag{15}$$

for $j = 1, ..., J$ with $\mathcal{L}_j^n$ the discrete operator defined by

$$\mathcal{L}_j^n \theta(z_j, t_n) = \frac{1}{h}\frac{\hat{C}_{j-1}}{\hat{F}_j - \hat{F}_{j-1}}\hat{E}_{j-1}(t_n)\theta(z_{j-1}, t_n) \tag{16}$$

$$- \frac{1}{h}\left(\frac{\hat{C}_j}{\hat{F}_{j+1} - \hat{F}_j} + \frac{\hat{C}_j}{\hat{F}_j - \hat{F}_{j-1}}\right)\hat{E}_j(t_n)\theta(z_j, t_n) \tag{17}$$

$$+ \frac{1}{h}\frac{\hat{C}_{j+1}}{\hat{F}_{j+1} - \hat{F}_j}\hat{E}_{j+1}(t_n)\theta(z_{j+1}, t_n) \tag{18}$$

and for the boundaries at $j = 0$ and $j = J + 1$:

$$\frac{\hat{C}_0}{\hat{F}_1 - \hat{F}_0}\hat{E}_0(T)\theta(z_0, T) = -\frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0}\hat{E}_1(T)\theta(z_1, T) \tag{19a}$$

$$\frac{\hat{C}_{J+1}}{\hat{F}_{J+1} - \hat{F}_J}\hat{E}_{J+1}(T)\theta(z_{J+1}, T) = -\frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J}\hat{E}_J(T)\theta(z_J, T) \tag{19b}$$

The boundary condition described by equations (19) is applicable to all schemes considered in this section as it is independent of the time-stepping.

The probability accumulated at the boundaries is discretized as:

$$\frac{\partial P_L}{\partial T}(T) = \frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0}\hat{E}_1(T)\theta(z_1, T) \tag{20}$$

$$\frac{\partial P_R}{\partial T}(T) = \frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J}\hat{E}_J(T)\theta(z_J, T) \tag{21}$$

4.1 Moment Preserving Implicit Euler

$$\theta_j^{n+1} - \theta_j^n = \delta \mathcal{L}_j^{n+1} \theta_j^{n+1} \tag{22a}$$

$$P_L(t_{n+1}) - P_L(t_n) = \delta \frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0} \hat{E}_1(t_{n+1}) \theta_1^{n+1} \tag{22b}$$

$$P_R(t_{n+1}) - P_R(t_n) = \delta \frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J} \hat{E}_J(t_{n+1}) \theta_J^{n+1} \tag{22c}$$

for $j = 1, ..., J$ and $n = 0, ..., N-1$.

4.2 Moment Preserving Crank-Nicolson

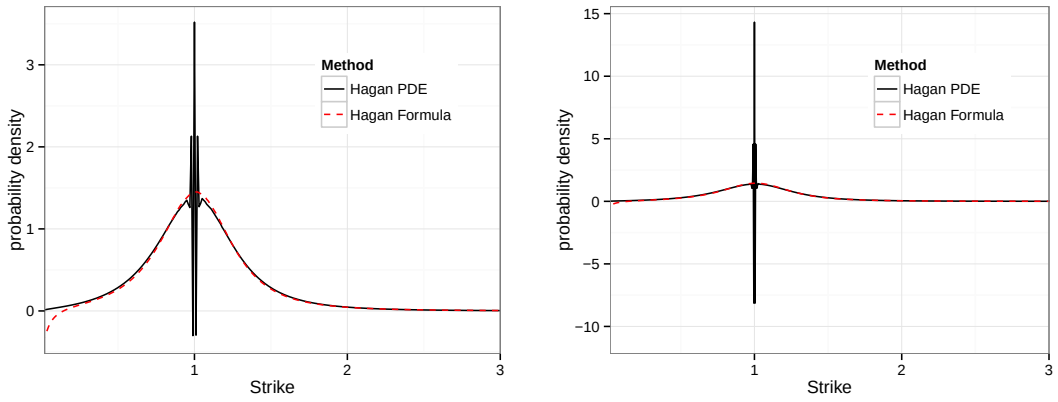
$$\theta_j^{n+1} - \theta_j^n = \frac{\delta}{2} \left(\mathcal{L}_j^{n+1} \theta_j^{n+1} + \mathcal{L}_j^n \theta_j^n \right) \tag{23a}$$

$$P_L(t_{n+1}) - P_L(t_n) = \frac{\delta}{2} \frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0} \left(\hat{E}_1(t_{n+1}) \theta_1^{n+1} + \hat{E}_1(t_n) \theta_1^n \right) \tag{23b}$$

$$P_R(t_{n+1}) - P_R(t_n) = \frac{\delta}{2} \frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J} \left(\hat{E}_J(t_{n+1}) \theta_J^{n+1} + \hat{E}_J(t_n) \theta_J^n \right) \tag{23c}$$

for $j = 1, ..., J$ and $n = 0, ..., N-1$.

5. Crank-Nicolson Oscillations with SABR



(a) PDE in Q with 40 time steps and $F_{max} = 5$

(b) PDE in θ with 80 time steps and $n_{sd} = 4$

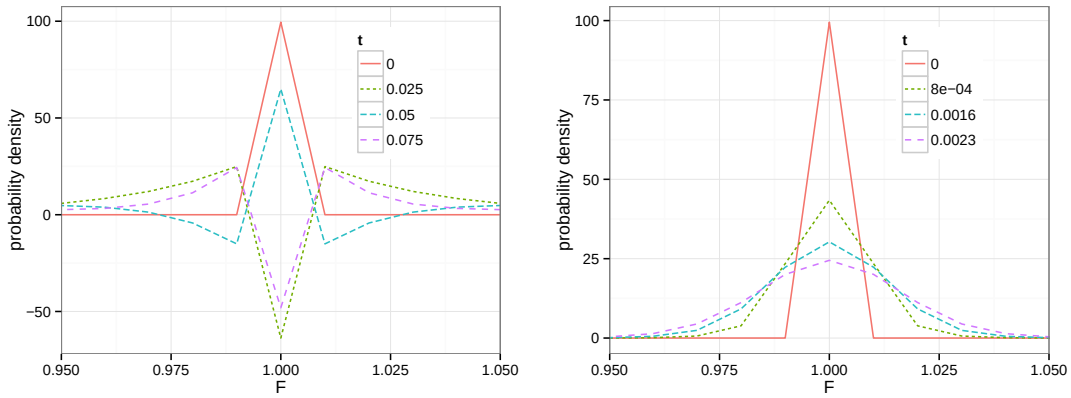
Figure 2.: Probability density in Hagan PDE discretised with Crank-Nicolson with 500 points. $\alpha = 35\%, \beta = 0.25, \rho = -10\%, \nu = 100\%, \tau_{ex} = 1$

We use the same parameters as the example of negative density with the standard SABR formula in (Hagan *et al.*, 2013): $\alpha = 35\%, \beta = 0.25, \rho = -10\%, \nu = 100\%$ and forward $f = 1$ at $\tau_{ex} = 1$, a relatively fine discretisation in the rate dimension (500 points, that is $h = 0.01005$) and large time-steps (40 steps, that is $\delta = 0.025$). Hagan *et al.* (2013) recommend between 200 and 500 points and 30 to 100 time-steps.

Figure 2 shows strong oscillations around the forward. To guarantee the absence of oscillations, the *Courant number* should be small enough $\Psi \leq 1$ (Theorem 2.2 in Morton and Mayers (2005)). For the uniform discretization of $Q(F)$, $\Psi_Q = M\frac{\delta}{h^2}$. This corresponds directly to the stability of the explicit Euler part of Crank-Nicolson. In practice, a higher value is acceptable because of a slight damping in Crank-Nicolson (Lawson and Morris, 1978). Although M depends on F, we can use the at-the-money value at f, as the spike is located there; that is,

$$\Psi_Q = \frac{1}{2}\alpha^2 f^{2\beta}\frac{\delta}{h^2} \tag{24}$$

$$\Psi_\theta = f^\beta \frac{\delta}{h^2} \tag{25}$$



(a) with 40 time steps $\Psi_Q = 15.16$ (b) with 1280 time steps $\Psi_Q = 0.47$

Figure 3.: First 4 time steps of the probability density in Hagan PDE discretised with Crank-Nicolson

In our example, in Figures 2 and 3(a) $\Psi_Q \approx 15$ while Figure 3(b) shows that indeed when $\Psi < 1$ there are no oscillations. The Crank-Nicolson oscillations are even stronger with the PDE in θ as for $\alpha \ll 1$, we have $\Psi_\theta \gg \Psi_Q$. Using the same SABR parameters and $n_{sd} = 3$ (corresponding to $F_{max} \approx 5$), $\Psi_\theta \approx 250$. In the next sections it is shown that a much smaller number of time steps can be used with other finite difference time-stepping techniques whilst still preserving good accuracy.

6. Alternative Schemes

We will focus our analysis on the PDE in θ, but similar conclusion can be drawn for the PDE in Q.

6.1 Rannacher

A common fix for Crank-Nicolson oscillations related to non smooth initial data is Rannacher time-stepping (Rannacher, 1984; Giles and Carter, 2006). It consists of introducing two half time-steps of implicit Euler time-stepping before applying Crank-Nicolson, because implicit Euler has much stronger damping properties. This comes at a cost in accuracy as implicit Euler is an order-1 scheme in time, especially when only a few time-steps are needed. The

SABR density discretisation will still be moment preserving if we discretise the Euler half steps as:

$$\theta_j^{n+\frac{1}{2}} - \theta_j^n = \frac{\delta}{2}\mathcal{L}_j^{n+\frac{1}{2}}\theta_j^{n+\frac{1}{2}} \tag{26a}$$

$$P_L(t_{n+\frac{1}{2}}) - P_L(t_n) = \frac{\delta}{2}\frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0}\hat{E}_1(t_{n+\frac{1}{2}})\theta_1^{n+\frac{1}{2}} \tag{26b}$$

$$P_R(t_{n+\frac{1}{2}}) - P_R(t_n) = \frac{\delta}{2}\frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J}\hat{E}_J(t_{n+\frac{1}{2}})\theta_J^{n+\frac{1}{2}} \tag{26c}$$

for $j = 1, ..., J$ and $n = 0, \frac{1}{2}, 1, \frac{3}{2}$. The next steps are Crank-Nicolson for $n = 2, ..., N-1$.

6.2 BDF2

The second order backward difference scheme (BDF2) is A-stable and L-stable multistep implicit scheme and will therefore damp any oscillation very quickly. Multisteps schemes have however severe limitations: instabilities will occur for sudden changes in the system variables, and initialization by another method is needed for the first steps. For example they can't be applied to linear complimentary problems like the pricing of American options through the discretization of the Black-Scholes PDE (Le Floc'h, 2013). Those issues don't arise with the SABR density PDE. The first moments will be preserved with the following discretization:

$$3\theta_j^{n+2} - 4\theta_j^{n+1} + \theta_j^n = 2\delta\mathcal{L}_j^{n+2}\theta_j^{n+2} \tag{27a}$$

$$3P_L(t_{n+2}) - 4P_L(t_{n+1}) + P_L(t_n) = 2\delta\frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0}\hat{E}_1(t_{n+2})\theta_1^{n+2} \tag{27b}$$

$$3P_R(t_{n+2}) - 4P_R(t_{n+1}) + P_R(t_n) = 2\delta\frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J}\hat{E}_J(t_{n+2})\theta_J^{n+2} \tag{27c}$$

for $j = 1, ..., J$ and $n = 0, ..., N-2$. Implicit Euler is used to compute θ_j^1, P_L^1, P_R^1 at $n = 0$.

6.3 Implicit Richardson Extrapolation

A simple Richardson extrapolation in time (Richardson, 1911) on implicit Euler will also provide a nearly order-2 scheme in time, keeping strong damping properties of the implicit Euler scheme at the cost of increased computational load: the implicit Euler scheme (equations 26) is evaluated with $\frac{\delta}{2}$ and δ. In practice it is around twice as slow as Crank-Nicolson. At $T = N\delta = \tau_{ex}$, we apply:

$$\theta(z) = 2\bar{\theta}^{\frac{\delta}{2}}(z) - \bar{\theta}^{\delta}(z) \tag{28}$$

$$P_L = 2\bar{P}_L^{\frac{\delta}{2}} - \bar{P}_L^{\delta} \tag{29}$$

$$P_R = 2\bar{P}_R^{\frac{\delta}{2}} - \bar{P}_R^{\delta} \tag{30}$$

where $\bar{\theta}^{\delta}, \bar{P}_L^{\delta}, \bar{P}_R^{\delta}$ are θ, P_L, P_R computed by implicit Euler with a time step of δ.

6.4 Lawson-Morris-Gourlay

A local Richardson extrapolation in time of second and third order is proposed in (Lawson and Morris, 1978) and (Gourlay and Morris, 1980). In practice, it is a faster alternative to the

standard Richardson extrapolation because the tridiagonal matrix stemming out of the finite difference discretisation can be reused, while keeping L-stability and thus strong damping properties.

For the second order scheme, at each time-step, equation (28) is applied. For the third order scheme, at each time-step we apply:

$$\theta(F) = 4.5\bar{\theta}^{\frac{\delta}{3}}(F) - 4.5\bar{\theta}^{\frac{2\delta}{3}}(F) + \bar{\theta}^{\delta}(F) \tag{31}$$

$$P_L = 4.5\bar{P}_L^{\frac{\delta}{3}} - 4.5\bar{P}_L^{\frac{2\delta}{3}} + \bar{P}_L^{\delta} \tag{32}$$

$$P_R = 4.5\bar{P}_R^{\frac{\delta}{3}} - 4.5\bar{P}_R^{\frac{2\delta}{3}} + \bar{P}_R^{\delta} \tag{33}$$

where $\bar{\theta}^{\frac{\delta}{3}}$ is θ computed by implicit Euler with 3 time steps of $\frac{\delta}{3}$ and $\bar{\theta}^{\frac{2\delta}{3}}$ is θ computed by implicit Euler with a time step of $\frac{2\delta}{3}$ and $\frac{\delta}{3}$. Being linear combinations of implicit Euler, those schemes are moment preserving.

6.5 Lawson-Swayne

A slightly faster second order unconditionally stable scheme is presented as a remedy to Crank-Nicolson in (Lawson and Swayne, 1976; Lawson and Morris, 1978). Let $b = 1 - \frac{\sqrt{2}}{2}$, it consists in applying two implicit Euler steps with time-step of $b\delta$ and an extrapolation on the values at those two steps.

First stage:

$$\begin{aligned}
\theta_j^{n+b} - \theta_j^n &= b\delta\mathcal{L}_j^{n+b}\theta_j^{n+b} \\
P_L(t_{n+b}) - P_L(t_n) &= b\delta\frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0}\hat{E}_1(t_{n+b})\theta_1^{n+b} \\
P_R(t_{n+b}) - P_R(t_n) &= b\delta\frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J}\hat{E}_J(t_{n+b})\theta_J^{n+b}
\end{aligned} \tag{34a}$$

Second stage:

$$\begin{aligned}
\theta_j^{n+2b} - \theta_j^{n+b} &= b\delta\mathcal{L}_j^{n+2b}\theta_j^{n+2b} \\
P_L(t_{n+2b}) - P_L(t_{n+b}) &= b\delta\frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0}\hat{E}_1(t_{n+2b})\theta_1^{n+2b} \\
P_R(t_{n+2b}) - P_R(t_{n+b}) &= b\delta\frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J}\hat{E}_J(t_{n+2b})\theta_J^{n+2b}
\end{aligned} \tag{34b}$$

And finally:

$$\begin{aligned}
\theta_j^{n+1} &= (\sqrt{2}+1)\theta_j^{n+2b} - \sqrt{2}\theta_j^{n+b} \\
P_L(t_{n+1}) &= (\sqrt{2}+1)P_L(t_{n+2b}) - \sqrt{2}P_L(t_{n+b}) \\
P_R(t_{n+1}) &= (\sqrt{2}+1)P_R(t_{n+2b}) - \sqrt{2}P_R(t_{n+b})
\end{aligned} \tag{34c}$$

for $j = 1, ..., J$ and $n = 0, ..., N-1$.

The scheme is moment preserving as it can also be seen as a linear combination of implicit Euler schemes.

6.6 TR-BDF2

TR-BDF2 is a two-stage method where the first stage consists in applying the (weighted) trapezoidal rule (Crank-Nicolson) and the second stage consists in applying the second order backward difference scheme (BDF2) on the first stage result and the first stage initial input (Bank *et al.*, 1985; LeVeque, 2007). It is second order accurate in time and L-stable. It is not to be confused with the simpler multistep method BDF2: the full step only depends on the previous full step while BDF2 depends on the two previous timesteps and can lose its accuracy (Windcliff *et al.*, 2001) with variable timesteps and linear complimentary problems. This scheme does not suffer from such drawbacks. The scheme has been applied to finance in the context of American option pricing (Le Floc'h, 2013).

First stage:

$$\begin{aligned}
\theta_j^{n+\alpha} - \theta_j^n &= \frac{\alpha\delta}{2}\left(\mathcal{L}_j^{n+\alpha}\theta_j^{n+\alpha} + \mathcal{L}_j^n\theta_j^n\right) \\
P_L(t_{n+\alpha}) &= P_L(t_n) + b\delta\frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0}\left(\hat{E}_1(t_{n+\alpha})\theta_1^{n+\alpha} + \hat{E}_1(t_n)\theta_1^n\right) \\
P_R(t_{n+\alpha}) &= P_R(t_n) + b\delta\frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J}\left(\hat{E}_J(t_{n+\alpha})\theta_J^{n+\alpha} + \hat{E}_J(t_n)\theta_J^n\right)
\end{aligned} \tag{35a}$$

Second stage:

$$\begin{aligned}
\theta_j^{n+1} &= \frac{1}{2-\alpha}\left(\frac{1}{\alpha}\theta_j^{n+\alpha} - \frac{(1-\alpha)^2}{\alpha}\theta_j^n + \delta(1-\alpha)\mathcal{L}_j^{n+1}\theta_j^{n+1}\right) \\
P_L(t_{n+1}) &= \frac{1}{2-\alpha}\left(\frac{1}{\alpha}P_L(t_{n+\alpha}) - \frac{(1-\alpha)^2}{\alpha}P_L(t_n) + \delta(1-\alpha)\frac{\hat{C}_1}{\hat{F}_1 - \hat{F}_0}\hat{E}_1(t_{n+1})\theta_1^{n+1})\right) \\
P_R(t_{n+1}) &= \frac{1}{2-\alpha}\left(\frac{1}{\alpha}P_R(t_{n+\alpha}) - \frac{(1-\alpha)^2}{\alpha}P_R(t_n) + \delta(1-\alpha)\frac{\hat{C}_J}{\hat{F}_{J+1} - \hat{F}_J}\hat{E}_J(t_{n+1})\theta_J^{n+1})\right)
\end{aligned} \tag{35b}$$

The weight α can be chosen to match Crank-Nicolson ($\alpha = \frac{1}{2}$) or to have proportional Jacobians ($\alpha = 2 - \sqrt{2}$). The later provides optimal stability (Dharmaraja *et al.*, 2009).

This can be extended to three-stages, with two stages of the trapezoidal rule and one stage of third order backward difference scheme (BDF3) as in (Bathe and Baig, 2005), resulting in a method with even stronger damping properties that we will name "Bathe":

$$\theta_j^{n+\frac{1}{3}} = \theta_j^n + \frac{\delta}{6}\left(\mathcal{L}_j^n\theta_j^n + \mathcal{L}_j^{n+\frac{1}{3}}\theta_j^{n+\frac{1}{3}}\right) \tag{36a}$$

$$\theta_j^{n+\frac{2}{3}} = \theta_j^n + \frac{\delta}{6}\left(\mathcal{L}_j^{n+\frac{1}{3}}\theta_j^{n+\frac{1}{3}} + \mathcal{L}_j^{n+\frac{2}{3}}\theta_j^{n+\frac{2}{3}}\right) \tag{36b}$$

$$\theta_j^{n+1} = \frac{1}{11}\left(18\theta_j^{n+\frac{2}{3}} - 9\theta_j^{n+\frac{1}{3}} + 2\theta_j^n + 2\delta\mathcal{L}_j^{n+1}\theta_j^{n+1}\right) \tag{36c}$$

6.7 Optimising for Performance

The function $M(F,T)$ needs to be computed for every grid point (F_j, t_n). The performance of the overall algorithm can be greatly improved by minimising the calls to the `pow` and `exp` functions as those are expensive. The quantities $\frac{1}{2}\alpha^2(1 + 2\rho\nu z + \nu^2 z^2)C^2(F)$ and $\rho\nu\alpha\Gamma$ are

constant in time and can be thus be cached between time-steps. A further improvement is to decompose t_{n+1} as $t_n + \delta$, then

$$e^{\rho\nu\alpha\Gamma t_{n+1}} = e^{\rho\nu\alpha\Gamma t_n} e^{\rho\nu\alpha\Gamma\delta} \tag{37}$$

We can therefore just compute the initial value $\frac{1}{2}\alpha^2(1 + 2\rho\nu z + \nu^2 z^2)C^2(F)$ together with $e_j = e^{\rho\nu\alpha\Gamma(F_j)\delta}$ for $j = 0, ..., J + 1$ once, and at each step simply update M as:

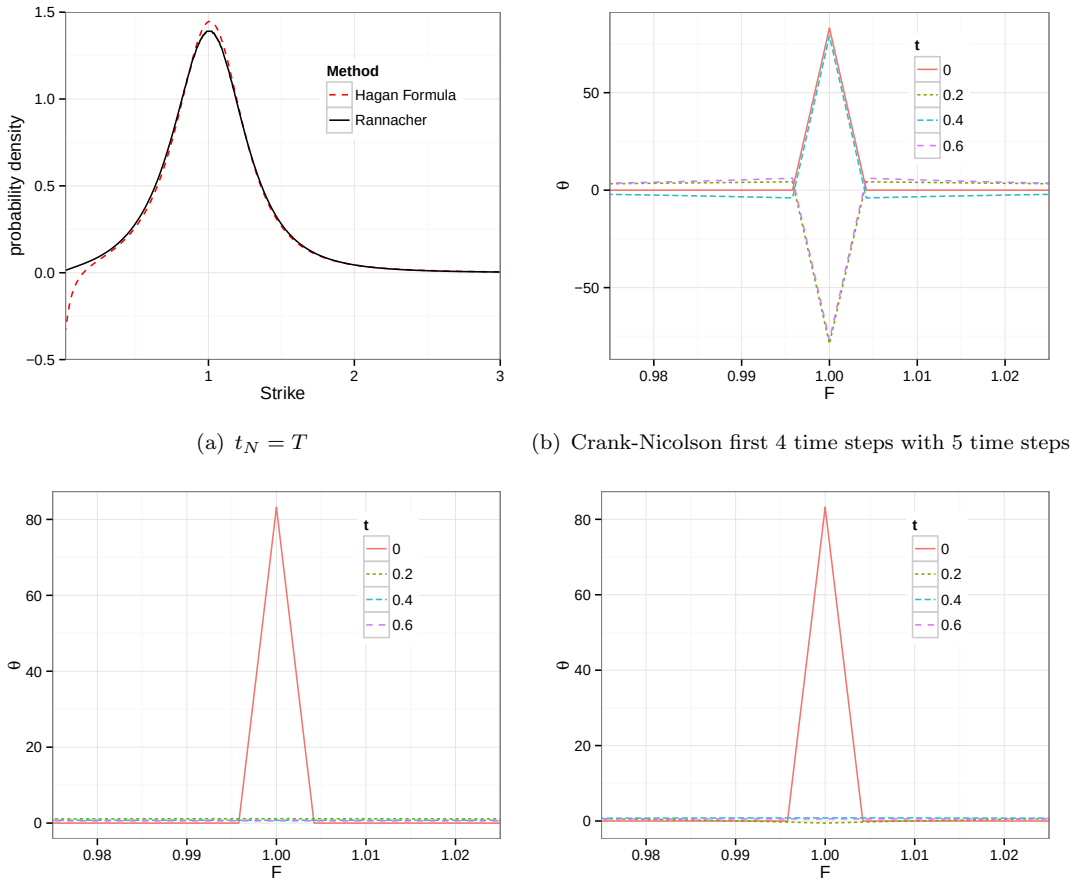
$$M_j^{n+1} = e_j M_j^n \tag{38}$$

This can be easily extended to multiple time-step sizes used in multi-stage schemes.

For multi-stage schemes, it is also possible to consider M as piecewise constant between full time-steps and thus to avoid its computation for fractions of time-steps. In our tests, this led to a slightly decreased accuracy and little performance gain. The increase in error was particularly visible for long term options and large time-steps. We did not make that approximation for the tests presented in the next section.

7. Numerical Results

7.1 Oscillations



(a) $t_N = T$

(b) Crank-Nicolson first 4 time steps with 5 time steps

(c) Rannacher first 4 time steps with 5 time steps

(d) Lawson-Swayne first 4 time steps with 5 time steps

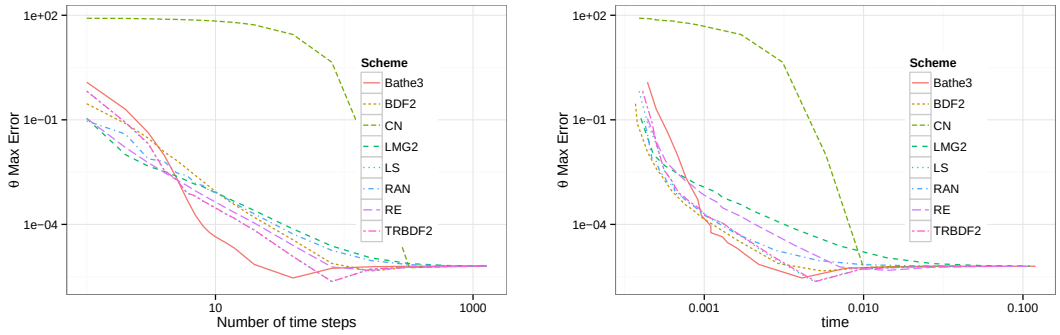
Figure 4.: Probability density in Hagan PDE using a total of 5 time-steps

With the same parameters as in section 5, Figure 4(a) shows a smooth positive probability density using only a total of 5 time-steps when Rannacher smoothing is applied to Crank-Nicolson. The density computed using second or third order Lawson-Morris-Gourlay (LMG2, LMG3), BDF2, Lawson-Swayne (LS), TR-BDF2 or Richardson extrapolation on implicit Euler (RE) would look very similar. Figure 4(c) show no apparent oscillations in the first steps for the Rannacher scheme. BDF2 and LMG2 would look the same. Lawson-Swayne shows a nearly imperceptible oscillation at the first step, and no more afterwards (Figure 4(d). TR-BDF2 and Bathe schemes behave similarly. In contrast, Crank-Nicolson had strong oscillations visible at $T = \tau_{ex}$ with 40 time steps for the PDE in Q and even stronger oscillations with 80 time steps for the PDE in θ.

7.2 Performance

7.2.1 Hagan Example

With the same parameters as in section 5, we look at the maximum error in the probability density with a varying number of time-steps compared to a Crank-Nicolson scheme with 5120 points for the rate dimension enough time-steps to ensure that $\Phi_\theta < 1$.



(a) Accuracy vs. number of time steps

(b) Accuracy vs. time

Figure 5.: Performance on Hagan example

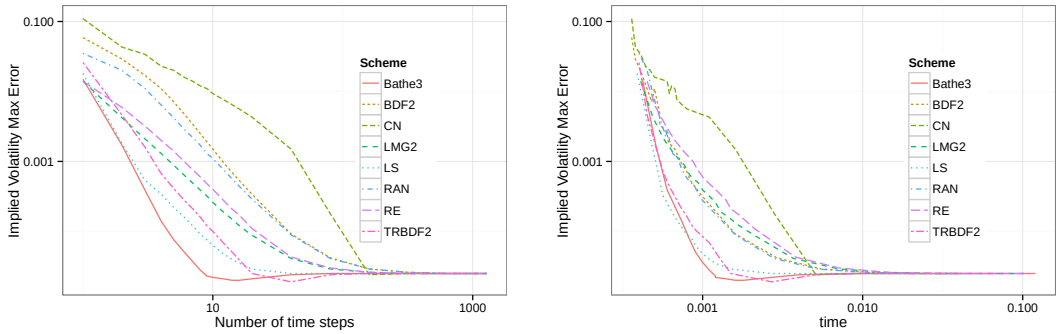
Other tests we performed indicate that the implied volatility maximum error or even the at-the-money implied volatility error would lead to similar conclusions. Furthermore, a Black implied volatility with an absolute error under 0.1% was achieved with only 3 time-steps for Bathe, 6 for Lawson-Swayne and TR-BDF2 Bathe, 10 for LMG2, and 12 for BDF2 and RAN. Lawson-Swayne is the most efficient scheme on this problem, closely followed by TR-BDF2, Rannacher and BDF2.

Higher order schemes like BDF3 or LMG3 were found to be no better performing than their second order variation on this problem.

7.2.2 Andreasen-Huge Example

We consider the SABR parameters used in (Andreasen and Huge, 2011): $\alpha = 0.0873, \beta = 0.7, \rho = -0.48, \nu = 0.47$ with a forward of $f = 0.025$ and a maturity $\tau_{ex} = 10.0$ and look at the maximum error in implied volatility between $0.2f$ and $2f$.

Only 3 time-steps are enough to achieve a Black implied volatility accuracy better than 0.1% with Lawson-Swayne and Bathe schemes, 4 time-steps for TR-BDF2, 5 for LMG2, 12 for RAN and BDF2.



(a) Accuracy vs. number of time steps

(b) Accuracy vs. time

Figure 6.: Performance on Andreasen-Huge example

8. Conclusion

It is possible to accurately compute option prices under the arbitrage free SABR approach with very few time-steps, even for long maturities. The Rannacher smoothing is a particularly simple and efficient way to improve accuracy significantly compared to Crank-Nicolson on this problem. BDF2 is simpler still and more efficient. Other schemes such as TR-BDF2 or Lawson-Swayne can further increase efficiency. In our experiments TR-BDF2 and Lawson-Swayne were robust and had similar stability and convergence properties.

Thus, with a careful choice of finite difference scheme, (Hagan *et al.*, 2013) is particularly competitive to the one step finite difference approach of (Andreasen and Huge, 2011).

References

Andreasen, J. and Huge, B. (2011) ZABR–Expansions for the Masses, *Available at SSRN 1980726*.

Bank, R., Coughran, W., Fichtner, W., Grosse, E., Rose, D. and Smith, R. (1985) Transient simulation of silicon devices and circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 4(4), pp. 436–451.

Bathe, K. J. and Baig, M. M. I. (2005) On a composite implicit time integration procedure for nonlinear dynamics, *Computers & Structures*, 83(31), pp. 2513–2524.

Benaim, S., Dodgson, M. and Kainth, D., An arbitrage-free method for smile extrapolation. (2008) , Technical report, Working Paper, QuaRC, Royal Bank of Scotland.

Dharmaraja, S., Wang, Y. and Strang, G. (2009) Optimal stability for trapezoidal-backward difference split-steps, *IMA Journal of Numerical Analysis*.

Duffy, D. (2004) A Critique of the Crank Nicolson Scheme Strengths and Weaknesses for Financial Instrument Pricing, *Wilmott Magazine*.

Edwards, J. D., Morel, J. E. and Knoll, D. A. (2011) Nonlinear variants of the TR/BDF2 method for thermal radiative diffusion, *Journal of Computational Physics*, 230(4), pp. 1198–1214.

Flavell, A., Machen, M., Eisenberg, B., Liu, C. and Li, X. (2013) A Conservative Finite Difference Scheme for Poisson-Nernst-Planck Equations, *arXiv preprint arXiv:1303.3769*.

Giles, M. and Carter, R. (2006) Convergence analysis of Crank-Nicolson and Rannacher time-marching, *Journal of Computational Finance*, 9(4), p. 89.

Gourlay, A. and Morris, J. L. (1980) The extrapolation of first order methods for parabolic partial differential equations, II, *SIAM Journal on Numerical Analysis*, 17(5), pp. 641–655.

Hagan, P. S. (2013) Change of Variables and Conservative Numerical Schemes, *Not published*.

Hagan, P. S., Kumar, D., Lesniewski, A. S. and Woodward, D. E. (2002) Managing smile risk, *Wilmott magazine*.

Hagan, P. S., Kumar, D., Lesniewski, A. S. and Woodward, D. E. (2013) Arbitrage free SABR, *Not published*.

Johnson, S. and Nonas, B. (2009) Arbitrage-free construction of the swaption cube, *Wilmott Journal*, 1(3), pp. 137–143.

Lawson, J. D. and Morris, J. L. (1978) The extrapolation of first order methods for parabolic partial differential equations. I, *SIAM Journal on Numerical Analysis*, 15(6), pp. 1212–1224.

Lawson, J. and Swayne, D. (1976) A simple efficient algorithm for the solution of heat conduction problems. In: *Proc. 6th Manitoba Conf. Numer. Math*, pp. 239–250.

Le Floc'h, F. (2013) TR-BDF2 for Stable American Option Pricing, *Journal of Computational Finance (to appear)*.

LeVeque, R. J. (2007) *Finite Difference Methods for Ordinary and Partial Differential Equations*, (Society for Industrial and Applied Mathematics (SIAM)).

Morton, K. W. and Mayers, D. F. (2005) *Numerical solution of partial differential equations: an introduction*, (Cambridge university press).

Oblój, J. (2008) Fine-tune your smile: Correction to Hagan et al, *Wilmott Magazine*.

Paulot, L. (2009) Asymptotic implied volatility at the second order with application to the SABR model, *Available at SSRN 1413649*.

Rannacher, R. (1984) Finite element solution of diffusion problems with irregular data, *Numerische Mathematik*, 43(2), pp. 309–327.

Richardson, L. F. (1911) The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam, *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210, pp. 307–357.

Windcliff, H., Forsyth, P. and Vetzal, K. (2001) Shout options: A framework for pricing contracts which can be modified by the investor, *Journal of Computational and Applied Mathematics*, 134(1-2), pp. 213–241.

Appendix A. Example Code

For illustration purpose, we detail here Octave code (also working with Matlab) for pricing vanilla options under the Arbitrage Free SABR model using Lawson-Swayne method.

Listing 1: makeTransformedSABRDensityLawsonSwayne.m - Matlab/Octave code computing the arbitrage free SABR density with Lawson-Swayne

```
function [P,PL,PR, zm, zmin, zmax, h] = makeTransformedSABRDensityLawsonSwayne(
    alpha, beta, nu, rho, forward, T, N, timesteps, nd)
  %init h,F and Q
  zmin = -nd*sqrt(T); zmax = -zmin;
  if (beta < 1)
    ybar = -forward^(1-beta)/(1-beta);
    zbar = -1/nu*log((sqrt(1-rho^2+(rho+nu*ybar/alpha)^2)-rho-nu*ybar/alpha)/(1-rho
        ));
    if (zbar > zmin)
      zmin = zbar;
    end
  end
  J = N-2;  h0 = (zmax-zmin)/(J);
  j0 = int32((0-zmin)/h0);
  h = (0-zmin)/(double(j0)-0.5);
  z = (0:(J+1))*h + zmin; zmax = z(J+2); zm = z - 0.5*h;
  ym = Y(alpha, nu, rho, zm); ymax = Y(alpha, nu, rho, zmax); ymin = Y(alpha, nu,
      rho, zmin);
  Fm = F(forward, beta, ym); Fmax = F(forward, beta, ymax); Fmin = F(forward, beta,
       ymin);
  Fm(1) = 2*Fmin-Fm(2); Fm(J+2)= 2*Fmax - Fm(J+1);
  Cm = sqrt(alpha^2+2*rho*alpha*nu*ym+nu^2*ym.^2).*Fm.^(beta);
  Cm(1) = Cm(2); Cm(J+2) = Cm(J+1);
  Gammam = (Fm.^beta-forward^beta)./(Fm-forward); Gammam(j0+1) = beta/forward.^(1-
      beta);
  dt = T/timesteps; Em = ones(1,J+2);
  b = 1 - sqrt(2)/2; %Lawson Swayne param
  dt1 = dt*b; dt2 = dt*(1-2*b);
  Emdt1 = exp(rho*nu*alpha*Gammam*dt1); Emdt1(1)= Emdt1(2); Emdt1(J+2)= Emdt1(J+1);
  Emdt2 = exp(rho*nu*alpha*Gammam*dt2); Emdt2(1)= Emdt2(2); Emdt2(J+2)= Emdt2(J+1);
  PL = 0.0; PR = 0.0; P = zeros(J+2,1); P(j0+1,1)=1.0/h;
  for t = 1:timesteps
    Em = Em .* Emdt1; [P1, PL1, PR1] = solveStep(Fm, Cm, Em, dt1, h, P, PL, PR);
    Em = Em .* Emdt1; [P2, PL2, PR2] = solveStep(Fm, Cm, Em, dt1, h, P1, PL1, PR1);
    P=(sqrt(2)+1)*P2-sqrt(2)*P1;
    PL=(sqrt(2)+1)*PL2-sqrt(2)*PL1;
    PR=(sqrt(2)+1)*PR2-sqrt(2)*PR1;
    Em = Em .* Emdt2;
    %Ptotal = sum(h*P(2:J+1))+PL+PR
    %Ftotal = Fm(2:J+1)*P(2:J+1)*h+Fmin*PL+Fmax*PR
  end
end
function [P, PL, PR] = solveStep(Fm, Cm, Em, dt, h, P, PL, PR)
  frac = dt/(2*h); M = length(P);
  B(2:M-1) = 1.0 + frac*(Cm(2:M-1).*Em(2:M-1).*(1./(Fm(3:M)-Fm(2:M-1))+1./(Fm(2:M
      -1)-Fm(1:M-2))));
  C(2:M-1) = -frac* Cm(3:M).*Em(3:M)./(Fm(3:M)-Fm(2:M-1));
  A(1:M-2) = -frac* Cm(1:M-2).*Em(1:M-2)./(Fm(2:M-1)-Fm(1:M-2));
  B(1) = Cm(1)/(Fm(2)-Fm(1))*Em(1); C(1) = Cm(2)/(Fm(2)-Fm(1))*Em(2);
  B(M) = Cm(M)/(Fm(M)-Fm(M-1))*Em(M); A(M-1) = Cm(M-1)/(Fm(M)-Fm(M-1))*Em(M-1);
  tri = diag(sparse(B))+diag(sparse(A),-1)+diag(sparse(C),1);
  P(1) = 0; P(M) = 0;
  P = tri\P;
  PL = PL + dt*Cm(2)/(Fm(2)-Fm(1))*Em(2)*P(2);
  PR = PR + dt*Cm(M-1)/(Fm(M)-Fm(M-1))*Em(M-1)*P(M-1);
end
function Y = Y(alpha, nu, rho, zm)
  Y = alpha/nu*(sinh(nu*zm)+rho*(cosh(nu*zm)-1));
end
function F = F(forward, beta, ym)
  F = (forward^(1-beta)+(1-beta)*ym).^(1/(1-beta));
end
```

The performance numbers in this paper come from an optimized Scala implementation, not from this Octave code.

Listing 2: priceCallTransformedSABRDensity.m - price a call option using Arbitrage Free SABR density

```
function p = priceCallTransformedSABRDensity(strike, Q, QL,QR, Fmin, Fmax, h)
   ystrike = (strike^(1-beta)-forward^(1-beta))/(1-beta);
   zstrike = -1/nu*log((sqrt(1-rho^2+(rho+nu*ystrike/alpha)^2)-rho-nu*ystrike/alpha
       )/(1-rho));
    if (zstrike < zmin)
      p = forward-strike;
    else
      if (zstrike > zmax)
        p = 0
      else
        k0 = ceil((zstrike-zmin)/h);
        ztilde = zmin+k0*h;
        ftilde = makeForward(alpha, beta, nu, rho, forward, ztilde);
        Fmax = makeForward(alpha, beta, nu, rho, forward, zmax);
        term = ftilde-strike;
        p = 0.5 * term * term *P(k0+1) + (Fmax - strike) * PR;
        k=k0+1:length(P)-2;
        zm = zmin + (k-0.5)*h;
        Fm = makeForward(alpha, beta, nu, rho, forward, zm);
        p = p + (Fm - strike) * h * P(k+1);
      end
    end
end
function F = makeForward(alpha, beta, nu, rho, forward, z)
  y = alpha/nu*(sinh(nu*z)+rho*(cosh(nu*z)-1));
  F = (forward^(1-beta)+(1-beta)*y).^(1/(1-beta));
end
```

Appendix B. Reference Implementation Values

Scheme	ATM price	$\theta(0)$	P_L	P_R
CN	0.156536999912	-75.391631075100	0.036151920718	0.000013551980
RAN	0.149164032279	0.486588975088	0.037035726447	0.000022398224
BDF2	0.149367146751	0.478480554725	0.036571170375	0.000034872631
RE	0.149620613592	0.482424676955	0.036971313630	-0.000001793511
LMG2	0.149447020543	0.486727660232	0.037356585469	-0.000003337903
LS	0.149699973996	0.482422521405	0.036472664324	0.000010671927
TRBDF2	0.149701545689	0.482401023656	0.036469263805	0.000010658861
Bathe	0.149629009401	0.486420051293	0.036725562889	0.000008443878

Table B1.: Sample values using 500 points and 5 time-steps,
$\alpha = 35\%, \beta = 0.25, \rho = -10\%, \nu = 100\%, T = 1, f = 1, n_{sd} = 4, \delta = 0.2, h = 0.012018637349$

Checking the moments at every step is very useful way to validate a scheme's implementation in code. To support further validation we recorded specific values from our implementations as reference. The TR-BDF2 is using a value of $\alpha = 2 - \sqrt{2}$ whilst the RAN scheme is using Rannacher time-stepping for the first two full time-steps. Extrapolation methods like RE and LMG2 produce here a very low negative accumulated probability at the higher boundary: there are very small oscillations in those schemes around the higher boundary. In practice it is unlikely to matter, furthermore, they quickly converge to the true value when the number of time-steps is increased.