Finite differences approximations for multidimensional models of pricing

Maya Briani Roberto Natalini Marco Papi
Cristiano Paris

1 Introduction

In the following we shall describe finite difference approximations for some multidimensional models of pricing. More preciley, we shall consider the Heston volatility model, a Two-factors stochastic volatility model, and a model for pricing swaptions in an affine framework. For each model we shall recall some financial backgrounds, we shall introduce the finite difference approximation and the setup for numerical tests. Some tests will be also described.

Premia 22

2 The Heston volatility model

Heston [1] proposed a stochastic model for asset volatility after the so-called *smile* effects was observed in the implied volatility of markets. Specifically, under the Heston model, the volatility and the asset price behave like the following SDE system:

$$\begin{aligned} dS &= (r-\delta)Sdt + \sqrt{V}SdW^1, \\ dV &= \kappa(\theta - V)dt + \sigma\sqrt{V}dW^2, \\ dW^1dW^2 &= \rho dt, \end{aligned}$$

where r is the spot interest rate, δ is the dividend paid by the asset S, V is the value of the spot volatility, θ is the long-run volatility, σ is the volatility of volatility (vol-vol) and W^1, W^2 two stochastic processes correlated by ρ.

Under the Feynman-Kac representation, the spot price U of the vanilla option satisfies the following PDE,

$$\begin{aligned} &\tfrac{1}{2}VS^2\tfrac{\partial^2 U}{\partial S^2} + \rho\sigma VS\tfrac{\partial^2 U}{\partial S\partial V} + \tfrac{1}{2}\sigma^2 V\tfrac{\partial^2 U}{\partial V^2} + (r-\delta)S\tfrac{\partial U}{\partial S} + \\ &+\{\kappa(\theta - V) - \Lambda(S,V,t)\sigma\sqrt{V}\}\tfrac{\partial U}{\partial V} - rU + \tfrac{\partial U}{\partial t} = 0. \end{aligned} \tag{1}$$

In this equation the term Λ represents the market price of volatility risk. For our implementation purposes, this term was removed from the model.

The PDE is usually given along with a proper payoff. In the case of vanilla european call option we have,

$$U(S,V) = \max\left[S - K, 0\right], \tag{2}$$

where K is the strike price. It should be noted that closed-form solutions of problem (1) for vanilla-option payoff do exist, methods for integrating such solutions can be found in [1, 2, 3]. Nevertheless, direct numerical integration of (1) is important when dealing with non-trivial payoff functions.

2.1 Numerical setup

A finite-differences scheme has been implemented to solve (1). Specifically, the following final-value problem with Dirichelet boundary conditions was considered,

$$\begin{cases} \frac{\partial U}{\partial t} + \mathbf{L}U - (r-\delta)U = 0, \\ U_B(S,V,t) = U_B(S,V,t), \quad (S,V) \in \partial D, \end{cases}$$

where $\mathbf{L}$ is the differential operator of (1), ∂D is the boundary of the cube $D = [S^{\mathrm{L}}, S^{\mathrm{R}}] \times [V^{\mathrm{L}}, V^{\mathrm{R}}]$ and U_B is a function over that boundary. In particular, the discounted payoff was used as the artificial boundary function,

$$U(S,V,t) = \max\left[S(t) - Ke^{-r(T-t)}, 0\right].$$

The intervals $[S^{\mathrm{L}}, S^{\mathrm{R}}]$ and $[V^{\mathrm{L}}, V^{\mathrm{R}}]$ were choosen wide enough so as to minimize the influence of the boundary conditions on the numerical solution. For example if we want to compute the solution at point $\left\{S^0, V^0\right\}$,

$$\begin{cases} S^{\mathrm{L}} = \max\left\{S^0 - \alpha_S S^0, \varepsilon\right\}, & S^{\mathrm{R}} = S^L + \alpha_S S^0, \\ V^{\mathrm{L}} = \max\left\{V^0 - \alpha_V V^0, \varepsilon\right\}, & V^{\mathrm{R}} = V^L + \alpha_V V^0. \end{cases}$$

where α_S and α_V are two constants and $0 < \varepsilon \approx 10^{-6}$.

Since the final problem runs backward in time, a simple change of time variable, $\tau = T - t$, was performed to solve an equivalent forward-time problem.

Further, we operated the following change of the U variable to eliminate the zero-order term,

$$\tilde{U}(S,V,\tau) = e^{r\tau}U(S,V,T-\tau)$$

The new Dirichelet problem then becomes,

$$\begin{cases} \frac{\partial \tilde{U}}{\partial \tau} = \mathbf{L}\tilde{U}, \\ \tilde{U}(S,V,\tau) = \tilde{U}_B(S,V,\tau), \quad (S,V) \in \partial D, \end{cases}$$

where $\tilde{U}(x,\tau) = U(x, T-\tau)$ and $\tilde{U}_B(x,\tau) = U_B(x, T-\tau)$.

After these changes of variables, the PDE was discretized using centered schemes for first and second order terms. For mixed derivatives we used the following expansion suggested by Bouchut [4]:

$$\frac{\partial^2 S}{\partial x_i \partial x_j} \approx \frac{1}{2}\frac{1}{\Delta x_i \Delta x_j}\left(\Delta^0_{x_i}\Delta^0_{x_j} + \Delta^+_{x_i}\Delta^-_{x_i}\Delta^+_{x_j}\Delta^-_{x_j}\right) S^n.$$

As shown by Bouchut, not only does this expansion provide second-order accuracy under Crank-Nicolson schemes, but also is monotone where the naive expansion is not.

The above expansions for the space variables, were used in an explicit scheme in time in order to smooth out the initial condition for the first 20 steps. After that, a Crank-Nicolson scheme was implemented. To keep the scheme stable we used a CFL condition for the first explicit steps as,

$$\Delta T_{\text{EX}} \approx \min\left\{\Delta x_1 V^R (S^R)^2, \Delta x_2 V^R \sigma^2\right\},$$

and, for the following time steps, using Crank-Nicolson approximation,

$$\Delta T_{\text{CN}} = \sqrt{\Delta T_{\text{EX}}}.$$

Under this setup, at each time step, the finite-differences approximation generate, both for the explicit and Crank-Nicolson part, the following linear system:

$$Ax_{n+1} + Bb_{n+1} = Cx_n + Db_n,$$

where A and C are $(N_1-1)(N_2-1)(N_3-1)$ generally non-symmetric banded square matrices, B and D are $[N_1N_2N_3 - (N_1-1)(N_2-1)(N_3-1)] \times (N_1-1)(N_2-1)(N_3-1)$ matrices. Vectors x_n, x_{n+1} are the solution at time n and $n+1$ respectively, while vectors b_n and b_{n+1} store the boundary conditions at time n and $n+1$ respectively. At time n, the x_n, b_n and b_{n+1} are all known, with x_0 coming from the problem's initial value.

For the explicit part $A \equiv I$ so the solution x_{n+1} can be directly calculated. For the Crank-Nicolson part, the solution x_{n+1} is calculated using a standard Stabilized BiConjugate Gradient (SBiCG) iterative algorithm with the Incomplete LU preconditioner. If the SBiCG algorithm doesn't converge to a solution with a specific accuracy after N steps, a Generalized Minimum Residuals (GMRES) algorithm is used.

2.2 Programming interface

The solver is implemented as a single function named `fn_hes()`. The function accepts the following parameters:

- `kappa`. Reversion rate.
- `sigma`. Volatility of volatility.
- `theta`. Lont-term volatility.
- `rho`. Asset-volatility correlation.
- `r`. Interest rate.
- `V0`. Current volatility.
- `S0`. Current asset price.
- `K`. Strike price.
- `T`. Exercise time (in years).

- `opttype`. Type of the option: 0 for European Call, 1 for European Put.
- `d`. Dividend rate.
- `N1`. Number of nodes along the asset price direction.
- `N2`. Number of nodes along the volatility direction.
- `V`. Pointer to a `double` that will contain the calculated option price.
- `delta`. Pointer to a `double` that will contain the calculated option price derivative.

The function returns an `integer`: 0 on success and 1 on failure. Compile the code with the `DEBUG` macro to get more info on errors.

2.3 Numerical Test

For the purpose of testing the implementation, we fixed the following parameters:

- `kappa`=2.
- `sigma`=0.2.
- `theta`=0.01.
- `rho`=0.5.
- `r`=0.1.
- `V0`=0.01.
- `K`=100.
- `T`=1.

In the table the results for a European Call for different asset spot prices on an Athlon AMD64 2.4Ghz and 2GB of RAM are shown: here N1 = 201 and N2 = 51. The results are compared with those calculated from the evaluation of the closed form formula as implemented in Premia:

S_0	P (\$)	Δ (\$)	time (s)	P_{CF} (\$)	Δ (\$)
90	3.020626	0.421302	3.919221	3.076410	0.422205
95	5.766727	0.689945	3.921054	5.792484	0.666280
100	9.829271	0.892125	4.261158	9.658327	0.863656
105	14.593351	0.971309	3.890816	14.253142	0.958117
110	19.533194	0.993057	4.361929	19.127745	0.988669

3 Two-factors stochastic volatility model: a finite differences approach

Following [5], we consider a two-factor stochastic volatility model (S_t, Y_t, Z_t), where S_t is the underlying price, and Y_t and Z_t are correlated diffusion processes. Under the risk-neutral probability measure, the model is described by the following equations:

$$\begin{aligned}
dS_t &= rS_t dt + \sigma_t S_t dW_t^S, \\
\sigma_t &= f(Y_t, Z_t), \\
dY_t &= \left(\alpha(m_f - Y_t) - \nu_f\sqrt{2\alpha}\text{ł}_f(Y_t, Z_t)\right) dt \\
&\quad + \nu_f\sqrt{2\alpha}\left(\rho_1 dW_t^S + \sqrt{1-\rho_1 2} dW_t^Y\right), \\
dZ_t &= \left(\delta(m_s - Z_t) - \nu_s\sqrt{2\delta}\text{ł}_s(Y_t, Z_t)\right) dt \\
&\quad + \nu_s\sqrt{2\delta}\left(\rho_2 dW_t^S + \rho_{12} dW_t^Y + \sqrt{1-\rho_2 2 - \rho_{12} 2} dW_t^Z\right).
\end{aligned}$$

Here (W_t^S, W_t^Y, W_t^Z) are independent standard Brownian motions, and the correlation coefficients ρ_1, ρ_2, and ρ_{12} satisfy $-1 < \rho_1 < 1$, $\rho_2 2 + \rho_{12} 2 < 1$ respectively.

The parameter $r > 0$ is the constant risk-free interest rate. It is also the rate of return of the stock S_t under the risk-neutral measure.
The risk-neutral probability is defined in terms of the market prices of volatility risk ł_f and ł_s, which we assume to be bounded and independent of the stock price S_t.
Moreover $\alpha > 0$, $\delta > 0$, $\nu_f > 0$, $\nu_s > 0$, and the volatility function $f(y, z)$ is smooth and bounded away from 0 (i.e. $0 < c_1 \leq f \leq c_2$, everywhere).

3.1 Price of a European Option

Given the payoff of un European option $H(S, y, z)$ of the stock price at maturity, using the Markov property, the no-arbitrage price of the option is obtained as the conditional expectation of the discounted payoff (given the current stock price driving volatility levels):

$$P(x, y, z, t) = E\left[e^{-r(T-t)}H(S_T)|S_t = x, Y_t = y, Z_t = z\right]. \qquad (\mathbf{P})$$

3.2 Pricing of a European Option

By applying the Feynman-Kac formula to $(\mathbf{P})$, the function P solves the three-dimensional partial differential equation

$$\frac{\partial P}{\partial t} + \mathbf{L}P - rP = 0, \qquad (3)$$

where P is the price of the option and

$$\mathbf{L} \equiv \alpha\mathbf{L}_0 + \sqrt{\alpha}\mathbf{L}_1 + \mathbf{L}_{\text{BS}} + \sqrt{\delta}\mathbf{M}_1 + \delta\mathbf{M}_2 + \sqrt{\alpha\delta}\mathbf{M}_3,$$

with,

- $\mathbf{L}_0 \equiv \nu_f^2 \frac{\partial^2}{\partial y^2} + (m_f - y)\frac{\partial}{\partial y}$,
- $\mathbf{L}_1 \equiv \nu_f^2 \sqrt{2}\left(\rho_1 S f(y,z)\frac{\partial^2}{\partial S \partial y} - \Lambda_f(y,z)\frac{\partial}{\partial y}\right)$,
- $\mathbf{L}_{\text{BS}} \equiv \frac{1}{2} f^2(y,z) S^2 \frac{\partial^2}{\partial S^2} + (r-d)S\frac{\partial}{\partial S}$,
- $\mathbf{M}_1 \equiv \nu_s^2 \sqrt{2}\left(\rho_2 S f(y,z)\frac{\partial^2}{\partial S \partial z} - \Lambda_f(y,z)\frac{\partial}{\partial z}\right)$,
- $\mathbf{M}_2 \equiv \nu_s^2 \frac{\partial^2}{\partial z^2} + (m_s - z)\frac{\partial}{\partial z}$,
- $\mathbf{M}_3 \equiv 2\nu_f \nu_s \left(\rho_1\rho_2 + \rho_{12}\sqrt{1-\rho_1^2}\right)\frac{\partial^2}{\partial y \partial z}$,

and with the payoff function:

$$H(S,y,z) = \max\left[S - K, 0\right]. \tag{4}$$

In the following, we considered Λ_f and Λ_s as equal to zero.

3.3 Numerical setup

A finite-differences scheme has been implemented to solve (3). Specifically, the following final-value problem with Dirichelet boundary conditions was considered,

$$\begin{cases} \frac{\partial P}{\partial t} + \mathbf{L}P - rP = 0, \\ P(S,y,z,t) = P_B(S,y,z,t), \quad (S,y,z) \in \partial D, \end{cases}$$

where $\mathbf{L}$ is the differential operator of equation (3), ∂D is the boundary of the cube $D = [S^{\text{L}}, S^{\text{R}}] \times [y^{\text{L}}, y^{\text{R}}] \times [z^{\text{L}}, z^{\text{R}}]$ and P_B is a function defined over that boundary. In particular, the following discounted payoff was used as the artificial boundary function,

$$P_B(S,y,z,t) = \max\left[S(t) - Ke^{-r(T-t)}, 0\right].$$

The intervals $[S^{\text{L}}, S^{\text{R}}]$, $[y^{\text{L}}, y^{\text{R}}]$ and $[z^{\text{L}}, z^{\text{R}}]$, were choosen wide enough so as to minimize the influence of the boundary conditions on the numerical solution. For example if we want to compute the solution at point $\left\{S^0, y^0, z^0\right\}$,

$$\begin{cases} S^{\text{L}} = \max\left\{S^0 - \alpha_S S^0, \varepsilon\right\}, & S^{\text{R}} = S^L + \alpha_S S^0, \\ y^{\text{L}} = \max\left\{y^0 - \alpha_y y^0, \varepsilon\right\}, & y^{\text{R}} = y^L + \alpha_y y^0. \\ z^{\text{L}} = \max\left\{z^0 - \alpha_z z^0, \varepsilon\right\}, & z^{\text{R}} = z^L + \alpha_z z^0. \end{cases}$$

where α_S, α_y and α_z are two constants and $0 < \varepsilon \approx 10^{-6}$.

Since the final problem runs backward in time, a simple change of time variable, $\tau = T - t$, was performed to solve an equivalent forward-time problem.

Further, we operated the following change of the P variable to eliminate the zero-order term,

$$\tilde{P}(S,y,z,\tau) = e^{r\tau} P(S,y,z,T-\tau)$$

The new Dirichelet problem then becomes

$$\begin{cases} \frac{\partial \tilde{P}}{\partial \tau} = \mathbf{L}\tilde{P}, \\ \tilde{P}(S,y,z,\tau) = \tilde{P}(S,y,z,\tau), \quad (S,y,z) \in \partial D \end{cases}$$

where $\tilde{P}(S,y,z,\tau) = S(S,y,z,T-\tau)$ and $\tilde{S}_B(S,y,z,\tau) = S_B(S,y,z,T-\tau)$.

After these changes of variables, the PDE was discretized using centered expansions for the first and second order terms. For mixed derivatives we used the following expansion suggested by Bouchut [4],

$$\frac{\partial^2 S}{\partial x_i \partial x_j} \approx \frac{1}{2}\frac{1}{\Delta x_i \Delta x_j}\left(\Delta^0_{x_i}\Delta^0_{x_j} + \Delta^+_{x_i}\Delta^-_{x_i}\Delta^+_{x_j}\Delta^-_{x_j}\right) S^n$$

As shown by Bouchut, not only does this expansion provide second-order accuracy under Crank-Nicolson schemes, but also is monotone where the naive expansion is not.

The above expansions for the space variables, were used in an explicit scheme in time in order to smooth out the initial condition for the first 20 steps. After that, a Crank-Nicolson scheme was implemented. To keep the scheme stable we used a CFL condition for the first explicit steps as,

$$\Delta T_{\text{EX}} \approx \min\left\{\Delta x_1 \frac{1}{2} f^2(y^{\text{R}}, z^{\text{R}}) S^{\text{R}}, \Delta x_2 \nu_f^2, \Delta x_3 \nu_s^2\right\}$$

under the assumption that f is increasing in its arguments. For the next Crank-Nicolson steps, we fix

$$\Delta T_{\text{CN}} = \sqrt{\Delta T_{\text{EX}}}$$

Under this setup, at each time step, the finite-differences approximation generate, both for the explicit and Crank-Nicolson part, the following linear system:

$$Ax_{n+1} + Bb_{n+1} = Cx_n + Db_n,$$

where A and C are $(N_1-1)(N_2-1)(N_3-1)$ generally non-symmetric banded square matrices, B and D are $[N_1N_2N_3 - (N_1-1)(N_2-1)(N_3-1)] \times (N_1-1)(N_2-1)(N_3-1)$ matrices. Vectors x_n, x_{n+1} are the solution at time n and $n+1$ respectively, while vectors b_n and b_{n+1} store the boundary conditions at time n and $n+1$ respectively. At time n, the x_n, b_n and b_{n+1} are all known, with x_0 coming from the problem's initial value.

For the explicit part $A \equiv I$ so the solution x_{n+1} can be directly calculated. For the Crank-Nicolson part, the solution x_{n+1} is calculated using a standard Stabilized BiConjugate Gradient (SBiCG) iterative algorithm with the Incomplete LU preconditioner. If the SBiCG algorithm doesn't converge to a solution with a specific accuracy after N steps, a Generalized Minimum Residuals (GMRES) algorithm is used.

3.4 Programming interface

The solver is implemented as a single function named `fn_stochvol2f()`. The function accepts the following parameters:

- `r`. Interest rate.

- `m_f`. Long-term fast-reverting process value.
- `m_s`. Long-term slow-reverting process value.
- `nu_s`. Volatility of the fast-reverting process value.
- `nu_f`. Volatility of the slow-reverting process value.
- `rho1`,`rho2`,`rho12`. Factor-asset correlations and factor-factor correlation.
- `alpha`. Fast reversion rate.
- `delta`. Slow reversion rate.
- `x0`. Current volatility.
- `y0`. Current fast-reverting process value.
- `z0`. Current slow-reverting process value.
- `K`. Strike price.
- `T`. Exercise time.
- `opttype`. Type of the option: 0 for European Call, 1 for European Put.
- `d`. Dividend rate.
- `N1`. Number of nodes along the asset price direction.
- `N2`. Number of nodes along the fast-reverting volatility direction.
- `N3`. Number of nodes along the slow-reverting volatility direction.
- `V`. Pointer to a `double` that will contain the calculated option price.
- `delta`. Pointer to a `double` that will contain the calculated option price derivative.

The function returns an `integer`: 0 on success and 1 on failure. Compile the code with the `DEBUG` macro to get more info on errors.

3.5 Results

For the purpose of testing the implementation, we fixed the following parameters:

- `r`=0.1.
- `mu_f`=-0.8.
- `mu_s`=-0.8.
- `nu_f`=0.5.
- `nu_s`=0.8.
- `rho1`=-0.2.

- `rho2`=-0.2.
- `rho12`=-0.0.
- `K`=50.
- `T`=1.
- `x0`=55.
- `y0`=-1.0.
- `z0`=-1.0.
- `d`=0.0.

In the table the results for a European Call for different values of δ and α on an Athlon AMD64 2.4Ghz and 2GB of RAM are shown: here N1 = 51, N2 = 51 amd N3 = 51. The results are compared with those found in the reference article [5]:

δ	α	P ($)	time (s)	P_{ref} ($)
100	0.01	9.758370	391.691974	11.03
50	0.05	9.761846	321.589108	10.99
20	0.1	9.809959	198.568684	11.00
5	1.0	10.077694	153.125092	11.60

4 Pricing of an option over a Bond in a Gaussian model with three factors.

An interest rate swaption is an option on an interest rate swap. It gives the holder the right but not the obligation to enter into an interest rate swap at a specific date (T_0 below) in the future, at a particular fixed rate and for a specified term. For an up-front fee (premium), the customer selects the strike rate (the level at which it enters the interest rate swap agreement), the length of the option period, the floating rate index (Prime, LIBOR), and tenor. Here we study and implement the model proposed in [6], see also [7].

4.1 The short rate model

The instantaneous short rate is defined as a linear combination of 3 factors, $r(t) = \delta + \sum_{j=1}^{3} x_j(t)$, described by Markov processes $x_j(t)$, $j = 1, 2, 3$, following a Gaussian model:

$$dx_j(t) = -k_j x_j(t)dt + \sigma_j dW_j(t), \qquad j = 1, 2, 3,$$

where:

- δ, k_j, σ_j, are constants for all the factors.
- $W_j(t)$, $j = 1, 2, 3$ are three Brownian motions (under the risk-neutral measure) which are dependent with each other, with instantaneous correlation coefficients ρ_{ij}, for $i, j = 1, 2, 3$.

4.2 The bond price

Let $P(x(t), t; T)$ denote the bond price on date t for an euro on date $T > t$. Then we will get:

$$P(x(t), t; T) = \mathbb{E}_t \left[e^{-\int_t^T r(s)ds} \right]$$

where $\mathbb{E}_t$ denotes the conditional expectation at time t under the risk-neutral measure.

For the 3-factor Guassian model described above, we have the following closed-form expression:

$$P(x(t), t; T) = \exp \left(B_0(T-t) - \sum_{j=1}^{3} B_j(T-t) x_j(t) \right),$$

where the deterministic functions B_0 and B_j satisfy a system of ordinary differential (Riccati) equations. In the case of the Gaussian model, there are explicit expressions for these functions. For all $k > 0$ and $s > 0$, define:

$$\varphi(s; k) \equiv \frac{1 - e^{-ks}}{k}.$$

Then, we have:

$$\begin{aligned} B_j(s) &= \varphi(s; k_j), \\ B_0(s) &= -\delta s + \frac{1}{2} \sum_{i,j=1} 3 \frac{\sigma_j \sigma_i \rho_{ij}}{k_i k_j} \left[s - \varphi(s; k_i) - \varphi(s; k_j) + \varphi(s; k_j + k_i) \right]. \end{aligned}$$

4.3 The swaption

Define the following quantities:

- T_0 denotes the exercise date of the swaption.
- $\{T_1, \ldots, T_N\}$ denote the dates the coupon payments are made on, with $T_i > T_0$, for any $i = 1, \ldots, N$.
- $\{C_1, \ldots, C_N\}$ denote the payments at dates T_i, $i = 1, \ldots, N$.
- $CB(T_0)$ denotes the price of the underlying coupon bond at the date T_0, and we have:
$$CB(T_0) = \sum_{i=1}^{N} C_i P(x(T_0), T_0; T_i).$$
- K is the strike price of the swaption.
- $S_{wn}(x(T_0), T_0)$ denotes the price of a swaption which is defined as:
$$S_{wn}(x(T_0), T_0) = \mathbb{E} \left[e^{-\int_t^{T_0} r(s)ds} \max \left(CB(T_0) - K, 0 \right) \right].$$

4.4 The PDE approach

The application of the Feynman-Kac formula gives a three-dimensional partial differential equation for the swaption price S:

$$\frac{\partial S}{\partial t} - \sum_{j=1}^{3} x_j k_j \frac{\partial S}{\partial x_j} + \frac{1}{2} \sum_{i,j=1}^{3} \rho_{ij} \sigma_i \sigma_j \frac{\partial^2 S}{\partial x_i \partial x_j} = \left(\delta + \sum_{j=1}^{3} x_j \right) S \tag{5}$$

where $x = (x_1, x_2, x_3) \in \mathbf{R}^3$. The payoff function is given by:

$$S_B(x, T_0) = \max\left(\mathrm{CB}(T_0) - K, 0\right) \tag{6}$$

4.5 Numerical setup

A finite-differences scheme has been implemented to solve eq. 5. Specifically, the following final-value problem with Dirichelet boundary conditions was considered:

$$\begin{cases} \frac{\partial S}{\partial t} + \mathbf{L}S = \left(\delta + \sum_{j=1}^{3} x_j \right) S \\ S(t, x) = S_B(t, x) & x \in \partial D \end{cases}$$

where $\mathbf{L} \equiv -\sum_{j=1}^{3} x_j k_j \frac{\partial}{\partial x_j} + \frac{1}{2} \sum_{i,j=1}^{3} \rho_{ij} \sigma_i \sigma_j \frac{\partial^2}{\partial x_i \partial x_j}$, ∂D is the boundary of the cube $D = [x_1^{\mathrm{L}}, x_1^{\mathrm{R}}] \times [x_2^{\mathrm{L}}, x_2^{\mathrm{R}}] \times [x_3^{\mathrm{L}}, x_3^{\mathrm{R}}]$ and S_B is a function over that boundary. In particular the discounted payoff was used as the artificial boundary function:

$$S(x, t) = \max\left(\mathrm{CB}(t) - KP(x, t; T_0), 0\right)$$

The intervals $[x_j^{\mathrm{L}}, x_j^{\mathrm{R}}]$ were choosen wide enough so as to minimize the influence of the boundary conditions on the numerical solution:

$$\begin{cases} x_j^{\mathrm{L}} = x_j^0 - 8 \left| x_j^0 \right| \\ x_j^{\mathrm{R}} = x_j^0 + 8 \left| x_j^0 \right| \end{cases}$$

where x_j^0 is the final condition of the j-th factor.

Moreover, a simple change of time variable was performed to solve an equivalent forward-time problem:

$$\tau = T - t$$

The new Dirichelet problem becomes:

$$\begin{cases} -\frac{\partial \tilde{S}}{\partial \tau} + \mathbf{L}\tilde{S} = \left(\delta + \sum_{j=1}^{3} x_j \right) \tilde{S} \\ \tilde{S}(\tau, x) = \tilde{S}_B(\tau, x) & x \in \partial D \end{cases}$$

where $\tilde{S}(x, \tau) = S(x, T - \tau)$ and $\tilde{S}_B(x, \tau) = S_B(x, T - \tau)$.

After this change of variable, the PDE was discretized using centered schemes for first and second order terms. For mixed derivatives we used the following expansion suggested by Bouchut [4]:

$$\frac{\partial^2 S}{\partial x_i \partial x_j} \to \frac{1}{2} \frac{1}{\Delta x_i \Delta x_j} \left(\Delta_{x_i}^0 \Delta_{x_j}^0 + \Delta_{x_i}^+ \Delta_{x_i}^- \Delta_{x_j}^+ \Delta_{x_j}^- \right) S^n$$

As shown by Bouchut, not only does this expansion provide second-order accuracy under Crank-Nicolson schemes, but also is monotone where the naive expansion is not.

The above expansions were used in an explicit scheme in order to smooth out the final condition for the first 20 steps. After that, a Crank-Nicolson scheme was implemented. With the aim of keeping the scheme stable we used a CFL condition for the explicit part of the scheme as:

$$\Delta T_{\text{EX}} \approx \min_j \left\{ \frac{1}{2} \Delta x_j \sigma_j^2 \right\}$$

and the following time step during the Crank-Nicolson part:

$$\Delta T_{\text{CN}} = \sqrt{\Delta T_{\text{EX}}}$$

Under this setup, the finite-differences expansions generate, both for the explicit and Crank-Nicolson part, the following linear system:

$$Ax_{n+1} + Bb_{n+1} = Cx_b + Db_n$$

where A and C are $(N_1-1)(N_2-1)(N_3-1)$ generally non-symmetric banded square matrices, B and D are $[N_1N_2N_3 - (N_1-1)(N_2-1)(N_3-1)]\times(N_1-1)(N_2-1)(N_3-1)$ matrices, all set up after the numerical scheme has been applied. x_i, b_i are vectors representing, for time n and $n+1$, the solution and the boundary conditions at a specific time. At time n, the x_n, b_n and b_{n+1} are all known, with x_0 coming from the problem's final value.

For the explicit part $A \equiv I$ so the solution x_{n+1} can be directly calculated. For the Crank-Nicolson part, the solution x_{n+1} is calculated using a standard Stabilized BiConjugate Gradient (SBiCG) iterative algorithm with the Incomplete LU preconditioner. If the SBiCG algorithm doesn't converge to a solution with a specific accuracy after N steps, a Generalized Minimum Residuals (GMRES) algorithm is used.

4.6 Programming interface

The solver is implemented as a single function named `fn_affineswaption`. The function accepts the following parameters:

- `rho12, rho13, rho23` Correlations between the factor processes.
- `kappa[]` Vector of `double`s, each for every reversion rate.
- `x0[]`. Vector of `double`s, each for every initial factor value.
- `sigma[]`. Vector of `double`s, each for every factor process volatility.
- `delta`. Intial interest rate displacement.
- `toption`. Exercise time (in years).
- `tenor`. Interval (in years) between payments.
- `tswap`. Last payment time (in years).
- `C`. Coupon payment amount.

- `K` Strike price.
- `N1`. Number of nodes along the x1 factor.
- `N2`. Number of nodes along the x2 factor.
- `N3`. Number of nodes along the x3 factor.
- `V`. Pointer to a `double` that will contain the calculated option price.
- `delta`. Pointer to a `double` that will contain the calculated option price derivative.

The function returns an `integer`: 0 on success and 1 on failure. Compile the code with the `DEBUG` macro to get more info on errors.

4.7 Numerical Test

For the purpose of testing the implementation, we fixed the following parameters:

- `rho12` = -0.2;
- `rho13` = -0.1;
- `rho23` = 0.3;
- `kappa[0]` = 1.0;
- `kappa[1]` = 0.2;
- `kappa[2]` = 0.5;
- `sigma[0]` = 0.01;
- `sigma[1]` = 0.005;
- `sigma[2]` = 0.002;
- `delta` = 0.06;
- `x0[0]` = 0.01;
- `x0[1]` = 0.005;
- `x0[2]` = -0.02;
- `toption` = 0.5;
- `tenor` = 0.25;
- `tswap` = 5.75;
- `C`=0.085;

In the table the results for a bond price for different strike prices on an Athlon AMD64 2.4Ghz and 2GB of RAM are shown: here N1 = N2 = N3 = 31. The results are compared with those calculated from a Monte Carlo simulation:

K	P	P_{MC}
0.5	0.993890	0.9495
0.75	0.750913	0.6979
1	0.507935	0.4471
1.15	0.362149	0.29661
1.25	0.264958	0.1961

References

[1] S.L. Heston, A closed-form solution for options with stochastic volatility with applications to bond and currency options, Review of Financial Studies Volume 6, Number 2 (1993), 327-343. 1, 2

[2] Heston, S. and S. Nandi (1997): A Closed Form GARCH Option Pricing Model, Federal Reserve Bank of Atlanta Working Paper 97-9, 1–34. 2

[3] S Mikhailov, U. Nögel. Heston's stochastic volatility model. Implementation, calibration and some extensions, Wilmott magazine, July, 2003. 2

[4] F. Bouchut, H. Frid, Finite difference schemes with cross derivatives correctors for multidimensional parabolic systems. J. Hyperbolic Differ. Equ. 3 (2006), no. 1, 27–52. 2, 7, 11

[5] Fouque, Jean-Pierre; Han, Chuan-Hsiang. Variance reduction for Monte Carlo methods to evaluate option prices under multi-factor stochastic volatility models. Quant. Finance 4 (2004), no. 5, 597–606. 5, 9

[6] X. Wang, Pricing Swaptions within the Affine Framework, working paper (Fall 2005). 9

[7] P Collin-Dufresne, RS Goldstein, Pricing Swaptions within the Affine Framework, Journal of Derivatives, Fall 2002. 9