ADI Methods for PIDEs

Ludovic Goudenège

Premia 22

1 Jump model and ADI schemes

We consider the following model for the stock price:

$$\left\{ \begin{array}{rcl} S_t & = & S_0 \exp(L_t) \\ dV_t & = & \alpha(\beta - V_t)dt + \omega\sqrt{V_t}dZ_t, \end{array} \right. \tag{1}$$

with α, β and $\omega \in \mathbb{R}_+$, and where Z_t is a Gaussian process and L_t is a Lévy process such that

$$L_t = \gamma t + \sigma W_t + Y_t$$

with Lévy triplet (γ, σ, ν). Under the pricing measure, we have to fix

$$\gamma = r - q - \frac{\sigma^2}{2} - \int_{\mathbb{R}} (\exp(x) - 1 - x - \chi_{|x|<1}\nu(dx)$$

where r is the interest rate, q is a continuous dividend, and under the condition

$$\int_{|x|>1} \exp(x)\nu(dx) < +\infty.$$

Moreover we have

$$\langle dW_t, dZ_t \rangle = \rho dt$$

as correlation between the two implied Gaussian processes.

Using a martingale approach for an european or an american option (call or put), we can prove that the price is given by the solution of the following partial integro-differential equation in the variable $x = \log(S/S_0)$.

$$\begin{aligned} \frac{\partial C(x,v,t)}{\partial t} \quad = \quad & \frac{1}{2} v \frac{\partial^2 C(x,v,t)}{\partial x^2} + \frac{1}{2}\omega^2 v \frac{\partial^2 C(x,v,t)}{\partial v^2} + \rho\sigma\omega\sqrt{v}\frac{\partial^2 C(x,v,t)}{\partial x \partial v} \\ & + \left(r - q - \frac{v}{2}\right) \frac{\partial C(x,v,t)}{\partial x} + \alpha(\beta - v)\frac{\partial C(x,v,t)}{\partial v} - rC(x,v,t) \\ & + \int_{\mathbb{R}} \left[C(x+y,v,t) - C(x,v,t) - (\exp(y) - 1)\frac{\partial C(x,v,t)}{\partial x} \right] \nu(dy) \end{aligned}$$

with the following boundary conditions for the call option

$$
\begin{aligned}
&\frac{\partial C}{\partial x}(x,v,t) = 0 && \text{whenever } x = X_{\min},\\
&\frac{\partial C}{\partial x}(x,v,t) = \exp(-q\ t) && \text{whenever } x = X_{\max},\\
&\frac{\partial C}{\partial v}(x,v,t) = 0 && \text{whenever } v = V_{\max},
\end{aligned}
$$

and the following boundary conditions for the put option

$$
\begin{aligned}
&\frac{\partial C}{\partial x}(x,v,t) = K\exp(-rt) && \text{whenever } x = X_{\min},\\
&\frac{\partial C}{\partial x}(x,v,t) = 0 && \text{whenever } x = X_{\max},\\
&\frac{\partial C}{\partial v}(x,v,t) = 0 && \text{whenever } v = V_{\max}.
\end{aligned}
$$

By default, the initial values are $S = S_0 = 100$ and $V = V_0 = 0.01$, the maturity is one year and the strike value is 100, such that $C(x,v,0) = (b(S_0\exp(x) - K))^+$ where $b = 1$ for the call and $b = -1$ for the put. In the case of the american options, we should add the possibility to exercice the option before the maturity which is easily implemented in the partial differential equation by taking the maximum compared to the pay-off at any time.

2 ADI finite difference scheme

We refer to [1] where a similar method is described to solve the partial differential equation without the jumps. We have used the same grids [1] whose sizes are given respectively for time, S-space, V-space by N_t, N_s and N_v. The default values are 40, 100 and 20. This choice ensures very good estimations for the prices of call or put options in a large variety of parameters in less than 1 second.

The Douglas scheme described in [1] has been implemented, but the methods for all the others schemes are potentially already in the code, since all the necessary functions are already implemented. See also the documentation for the Heston model.

3 Splitting method for the PIDE

The method developed in [2] is based on a splitting between the classical PDE part (with linear differential operator D) and the integral part (with linear integral operator J). It consist in applying the following scheme between time t and time $t + \Delta_t$.

$$
C(x,v,t+\Delta_t) \simeq \exp\left(\frac{\Delta_t}{2}D\right)\exp\left(\Delta_t J\right)\exp\left(\frac{\Delta_t}{2}D\right)C(x,v,t)
$$

[1] up to a logarithm transformation for the space variable

In the Merton model, the operator J is given by

$$J = \lambda \left(\exp\left(\lambda\mu_J + \frac{1}{2}\sigma_J^2 \nabla^2 \right) - \kappa\nabla - 1 \right), \qquad \kappa = \exp\left(\mu_J + \frac{\sigma_J^2}{2} \right) - 1,$$

where ∇ is the derivative in the space variable x. Thus we want to solve

$$\begin{aligned}
C^{n+\frac{1}{4}}(x,v) &= \exp\left(\frac{\Delta_t}{2} D \right) C^n(x,v), \text{ with ADI solver} \\
C^{n+\frac{3}{4}}(x,v) &= \exp\left(\lambda\Delta_t \left(\exp\left(\lambda\mu_J + \frac{1}{2}\sigma_J^2 \nabla^2 \right) - \kappa\nabla - 1 \right) \right) C^{n+\frac{1}{4}}(x,v) \\
C^{n+1}(x,v) &= \exp\left(\frac{\Delta_t}{2} D \right) C^{n+\frac{3}{4}}(x,v), \text{ with ADI solver.}
\end{aligned}$$

With a Páde approximation, taking out the $-\lambda\kappa\nabla$ part in the ADI solver, it is sufficient to solve

$$\frac{\partial z(x,s)}{\partial s} = \left(\mu_J \nabla + \frac{1}{2}\sigma_J \nabla^2 \right) z(x,s)$$

for $0 \leq s \leq 1$ and $z(x,0) = C(x,v)$, then denote

$$J^* : C \mapsto \lambda C + z(x,1)$$

and to solve a Cranck-Nicolson scheme

$$C^{n+\frac{3}{4}}(x,v,t) - C^{n+\frac{1}{4}}(x,v,t) = \frac{1}{2}\Delta_t \left[C^{n+\frac{3}{4}}(x,v,t) + C^{n+\frac{1}{4}}(x,v,t) \right]$$

4 Implementation

The main program fixes the variables and compute the grid in space and variance variables. It calls the function `compute_price_and_delta` which first makes allocation of all the necessary arrays and builds all the matrix involved in ADI solver. Next this function makes a loop over time making all the splitting scheme by calling `time_evolution`. This function makes a half ADI time step in order to compute $C^{n+\frac{1}{4}}$. Then it calls the function `compute_jumps_inline` which computes the integral part by solving equations described above. At this step, we have compute $C^{n+\frac{3}{4}}$ and finally it makes another half ADI time steps in order to compute C^{n+1}.

References

[1] Tinne Haentjens and Karel J. in't Hout. ADI finite difference schemes for the Heston-Hull-White PDE. J. Comp. Finan. 16, 83-110 (2012). 2

[2] Andrey Itkin High-Order Splitting Methods for Forward PDEs and PIDEs. International Journal of Theoretical and Applied Finance. 2

[3] Andrey Itkin Efficient Solution of Backward Jump-Diffusion PIDEs with Splitting and Matrix Exponentials. arXiv:1304.3159v3 (2013).