Method of lines for the evaluation of barrier options under Heston model

Ludovic Goudenège

Premia 22

1 Heston model

We consider the following model for the stock price:

$$\left\{\begin{array}{rcl} dS_t &=& rS_t + \sqrt{V_t} S_t dW_t^S, \\ dV_t &=& \kappa_v(\theta_v - V_t)dt + \sigma_v \sqrt{V_t} dW_t^V. \end{array}\right. \tag{1}$$

where r is the domestic rate (default value is 10% per year). Moreover there is dividend (variable *divid* whose default value is 0%) and correlation factors between the two brownian motion given by ρ_{sv} whose default value is 0.5.

The initial values of asset and variance are $S = S_0$ and $V = V_0$, whose default values are given respectively by 100 and 0.01. The maturity is one year, the strike value K is 100, the barrier H is 110 and the rebate is 10.

Under these assumptions, we solve the following PDE

$$\begin{array}{rcl} \dfrac{\partial C}{\partial t} &=& \dfrac{1}{2}s^2 v \dfrac{\partial^2 C}{\partial s^2} + \dfrac{1}{2}\sigma_v^2 v \dfrac{\partial^2 C}{\partial v^2} + \rho_{sv}\sigma_v s v \dfrac{\partial^2 C}{\partial s \partial v} \\ && + rs\dfrac{\partial C}{\partial s} + \kappa_v(\theta_v - v)\dfrac{\partial C}{\partial v} - rC, \end{array}$$

with the following boundary conditions

$$\begin{array}{rcll} C(s,v,t) &= 0 & \text{whenever } s = 0, \\ C(s,v,t) &= rebate & \text{whenever } s = H, \\ \dfrac{\partial C}{\partial v}(s,v,t) &= 0 & \text{whenever } v = V_{\max}, \end{array}$$

for the call option, and

$$\begin{array}{rcll} C(s,v,t) &= K\exp(-rt) & \text{whenever } s = 0, \\ C(s,v,t) &= rebate & \text{whenever } s = H, \\ \dfrac{\partial C}{\partial v}(s,v,t) &= 0 & \text{whenever } v = V_{\max}, \end{array}$$

for the put option.

We refer to [1] where the complete method is described. For the variance grid we have used an uniform grid between 0 and $MIN(MAX(100V_0, 1), 5)$. For the asset grid, we have used a refined grid whose refinement occurs near the strike and the barrier. Due to the method the asset grid does not start at $S = 0$ but only at $S = 0.6 * S_0$. For this reason, the boundary conditions have been slightly modified for the put option in $K\exp(-rt) - s\exp(-qt)$.

The sizes are given respectively for time, S-space and V-space by N_t, N_s and N_v whose default values are 50, 500 and 50. This choice ensures good estimations for the prices of call or put options in a large variety of parameters in approximately less than 3 seconds.

2 Method of line algorithm - System of ODE's

The method of line algorithm consists in solving a system of coupled ODE's given by

$$\begin{aligned} & \frac{v_m s^2}{2}\frac{d^2C_m^n}{ds^2} + \rho_{sv}\sigma_v v_m s\frac{V_{m+1}^n - V_{m-1}^n}{2\Delta v} + \frac{\sigma_v^2 v_m}{2}\frac{C_{m+1}^n - 2C_m^n + C_{m-1}^n}{(\Delta v)^2} \\ + & \frac{\kappa_v(\theta_v - v_m)}{2}\frac{C_{m+1}^n - C_{m-1}^n}{\Delta v} + \frac{|\kappa_v(\theta_v - v_m)|}{2}\frac{C_{m+1}^n - 2C_m^n + C_{m-1}^n}{\Delta v} \\ + & (r-q)s\frac{dC_m^n}{ds} - rC_m^n - \frac{C_m^n - C_m^{n-1}}{\Delta\tau} = 0, \end{aligned}$$

or equivalently

$$\left\{\begin{array}{rcl} \frac{dC_m^n}{ds} & = & V_m^n \\ \frac{dV_m^n}{ds} & = & \frac{-2}{v_m s^2}\left[\frac{\sigma_v^2 v_m}{2}\frac{C_{m+1}^n - 2C_m^n + C_{m-1}^n}{(\Delta v)^2}\right. \\ & + & \frac{\kappa_v(\theta_v - v_m)}{2}\frac{C_{m+1}^n - C_{m-1}^n}{\Delta v} + \frac{|\kappa_v(\theta_v - v_m)|}{2}\frac{C_{m+1}^n - 2C_m^n + C_{m-1}^n}{\Delta v} \\ & + & (r-q)sV_m^n - rC_m^n - \frac{C_m^n - C_m^{n-1}}{\Delta\tau} \\ & + & \left.\rho_{sv}\sigma_v v_m s\frac{V_{m+1}^n - V_{m-1}^n}{2\Delta v}\right], \end{array}\right.$$

for $m = 1, \ldots, N_v - 1$, $n = 1, \ldots, N_t$, on the domain $[S_{\min}, S_{\max}]$ discretized by N_s points. At the boundary $v_{N_v} = V_{\max}$ i.e. $m = N_v$ we use virtually $C_{N_v+1}^n = C_{N_v-1}^n$ and $V_{N_v+1}^n = V_{N_v-1}^n$ such that the ODE for $m = N_v$ has changed and has becomed

$$\begin{aligned} & \frac{v_{N_v} s^2}{2}\frac{d^2C_{N_v}^n}{ds^2} + \frac{\sigma_v^2 v_{N_v}}{2}\frac{2C_{N_v-1}^n - 2C_{N_v}^n}{(\Delta v)^2} \\ + & \frac{|\kappa_v(\theta_v - v_{N_v})|}{2}\frac{2C_{N_v-1}^n - 2C_{N_v}^n}{\Delta v} \\ + & (r-q)s\frac{dC_{N_v}^n}{ds} - rC_{N_v}^n - \frac{C_{N_v}^n - C_{N_v}^{n-1}}{\Delta\tau} = 0, \end{aligned}$$

or equivalently

$$\left\{\begin{array}{rcl} \frac{dC_{N_v}^n}{ds} & = & V_{N_v}^n \\ \frac{dV_{N_v}^n}{ds} & = & \frac{-2}{v_{N_v} s^2}\left[\sigma_v^2 v_{N_v}\frac{C_{N_v-1}^n - C_{N_v}^n}{(\Delta v)^2} + |\kappa_v(\theta_v - v_{N_v})|\frac{C_{N_v-1}^n - C_{N_v}^n}{\Delta v}\right. \\ & + & \left.(r-q)sV_{N_v}^n - rC_{N_v}^n - \frac{C_{N_v}^n - C_{N_v}^{n-1}}{\Delta\tau}\right]. \end{array}\right.$$

At the boundary $v = 0$, we have an upwinding scheme in volatility such that

$$\kappa_v\theta_v\frac{C_1^n - C_0^n}{\Delta v} + (r-q)s\frac{dC_0^n}{ds} - rC_0^n - \frac{C_0^n - C_0^{n-1}}{\Delta\tau} = 0,$$

or equivalently

$$\frac{dC_0^n}{ds} = \frac{1}{(r-q)s}\left[\kappa_v\theta_v\frac{C_0^n - C_1^n}{\Delta v} + rC_0^n + \frac{C_0^{n-1} - C_0^n}{\Delta\tau}\right].$$

But it is suggested in the article [1] that we do not use this equation. Instead we use a quadratic interpolation of the values in v_1, v_2 and v_3 to compute the value at $v_0 = 0$.

3 Riccati equations

For each system of ODE's we use a Riccati transformation given by

$$C_m^n(s) = R_m(s)V_m^n(s) + W_m^n(s)$$

where R and W are solutions to the following ODE's on $[S_{\min}, H]$

$$\begin{aligned}
\frac{dR_m}{ds} &= 1 - B_m(s)R_m(s) - A_m(s)R_m(s)^2, \qquad R_m(S_*) = \Gamma, \\
\frac{dW_m^n}{ds} &= -A_m(s)R_m(s)W_m^n(s) - R_m(s)P_m^n(s), \quad W_m^n(S_*) = \alpha^n.
\end{aligned}$$

To obtain V_m^n we finally solve the ODE on $[S_{\min}, H]$

$$\frac{dV_m^n}{ds} = A_m(s)(R_m(s)V_m^n(s) + W_m^n(s)) + B_m(s)V_m^n(s) + P_m^n(s),$$

with the boundary condition $V_m^n(S^*) = \nu_m^n$. For $m = 1, \ldots, N_v - 1$ we have the expressions

$$\begin{aligned}
A_m(s) &= \frac{2}{v_m s^2}\left[\frac{\sigma_v^2 v_m}{(\Delta v)^2} + \frac{|\kappa_v - \theta_v v_m|}{\Delta v} + r + \frac{1}{\Delta\tau}\right], \\
B_m(s) &= \frac{-2(r-q)s}{v_m s^2}, \\
P_m(s) &= \frac{-2}{v_m s^2}\left[\frac{\sigma_v^2 v_m}{2}\frac{C_{m+1}^n(s) + C_{m-1}^n(s)}{(\Delta v)^2} + \frac{C_m^{n-1}(s)}{\Delta\tau}\right. \\
&+ \left.\frac{\kappa_v - \theta_v v_m}{2}\frac{C_{m+1}^n(s) - C_{m-1}^n(s)}{\Delta v} + \frac{|\kappa_v - \theta_v v_m|}{2}\frac{C_{m+1}^n(s) + C_{m-1}^n(s)}{\Delta v}\right] \\
&- \frac{\rho_{sv}\sigma_v}{s}\frac{V_{m+1}^n(s) - V_{m-1}^n(s)}{\Delta v}.
\end{aligned}$$

For the boundary at $v = V_{N_v}$, we obtain the systems

$$\begin{aligned}
A_{N_v}(s) &= \frac{2}{v_{N_v} s^2}\left[\frac{\sigma_v^2 v_{N_v}}{(\Delta v)^2} + \frac{|\kappa_v - \theta_v v_{N_v}|}{\Delta v} + r + \frac{1}{\Delta\tau}\right], \\
B_{N_v}(s) &= \frac{-2(r-q)s}{v_{N_v} s^2}, \\
P_{N_v}(s) &= \frac{-2}{v_{N_v} s^2}\left[\sigma_v^2 v_{N_v}\frac{C_{N_v-1}^n(s)}{(\Delta v)^2} + \frac{C_{N_v}^{n-1}(s)}{\Delta\tau} + |\kappa_v - \theta_v v_{N_v}|\frac{C_{N_v-1}^n(s)}{\Delta v}\right].
\end{aligned}$$

The Riccati transformation is based on the choice for S_*, S^*, Γ, α^n and ν_m^n. They are given respectively by the following tables:

Variables	Call	Put
S_*	$S_{\min}$	$S_{\min}$
Γ	0	0
α^n	0	$K\exp(-rt) - s\exp(-qt)$
S^*	H	H
ν_m^n	$\dfrac{rebate - W_m^n(H)}{R_m(H)}$	$\dfrac{rebate - W_m^n(H)}{R_m(H)}$

Table 1: Variables for ODE's in European case.

where b_m^n is the early exercice border found with monitoring of the function $\phi(s) = R_m(s) + W_m^n(s) - (s - K)$ for the call option and $\phi(s) = -R_m(s) + W_m^n(s) - (K - s)$ for the put option. And where $\Theta = MAX(H - K, rebate)$ and $\Lambda = MAX(K - S_{\min}, K\exp(-rt) - S_{\min}\exp(-qt))$. It is clear that the ODE's are solved forward or backward with respect to the value of S_* and S^*, and obviously the ODE's for R and W are not solved in the same way than the ODE for V.

Variables	Call	Put
S_*	$S_{\min}$	H
Γ	0	0
α^n	0	$rebate$
S^*	b_m^n	b_m^n
ν_m^n	$\begin{cases} 1, & \text{if } b_m^n < H \\ \dfrac{\Theta - W_m^n(H)}{R_m(H)}, & \text{else.} \end{cases}$	$\begin{cases} -1, & \text{if } b_m^n > S_{\min} \\ \dfrac{\Lambda - W_m^n(S_{\min})}{R_m(S_{\min})}, & \text{else.} \end{cases}$

Table 2: Variables for ODE's in American case.

4 Implementation

The implementation of the method of lines for the evaluation of barrier option in Heston model is relatively easy. But the main difficulty is in the loop over all the variance lines for a fixed time. In fact, given a time t_n and a variance v_m, solving the ODE in C_m^n is quite easy since the ODE's are not singular. But since the coefficient of the ODE's depends on values C_{m-1}^n, C_{m+1}^n and C_m^{n-1}, we have an implicit system of ODE's.

The idea is to start with given values (those at time $n-1$) for all the variables, solve the system for $m = 1$ using these given values, update the value C_1^n, solve the system for $m = 2$ using again given values for C_3^n but this new value for C_1^n, etc. At $m = N_v$ the value C_{N_v+1} is not needed and the loop finishes. Finally compute the value of C_0^n with quadratic interpolation.

At this point, restart the same algorithm for the same time $n-1$ at $m = 1$ but use the values found during the preceding algorithm for all the variables. Loop again while the difference between two iterations are greater than 10^{-8}. Actually in the american put option case, there are multiple iteration where there is no convergence of this algorithm and we have fixed to 50 the maximum number of iteration. This lack of convergence is due to the early exercice border which can jump from one point to another and reversely between two iterations.

Most of the functions are very short. I detail here the principal functions.

`static int find_early_exercice(double K, double *sgrid, int Ns, int call_or_put, double *sol_R, double *sol_W)` This function monitors the value of $R + W$ or $-R + W$ with respect to the payoff. It finds the index where there will be a early exercice.

```
static void solve_R_edo(double coeff_A, double coeff_B, double *sgrid, int Ns,
double initial, double *sol_R)
static void solve_R_edo_backward(double coeff_A, double coeff_B, double *sgrid,
int Ns, double terminal, double *sol_R)
static void solve_W_edo(double coeff_A, double *coeff_P, double *coeff_Q,
double *sgrid, int Ns, double initial, double *sol_R, double *sol_W)
static void solve_W_edo_backward(double coeff_A, double *coeff_P, double *coeff_Q,
double *sgrid, int Ns, double terminal, double *sol_R, double *sol_W)
static void solve_V_edo_backward(double coeff_A, double coeff_B, double *coeff_P,
double *coeff_Q, double *sgrid, int Ns, double terminal, double *sol_R,
double *sol_W, double *sol_V)
static void solve_V_edo_call_american(double coeff_A, double coeff_B,
double *coeff_P, double *coeff_Q, double *sgrid, int Ns, double terminal,
int index, double *sol_R, double *sol_W, double *sol_V)
static void solve_V_edo_put_american(double coeff_A, double coeff_B,
double *coeff_P, double *coeff_Q, double *sgrid, int Ns, double initial,
int index, double *sol_R, double *sol_W, double *sol_V)
```

All these functions have explicit names and they solve the ODE's with trapezoidal rules.

```
static void solve_lines_europ(...)
static void solve_lines_american(...)
```
These two functions solve the ODE's over one variance line from $m = 1$ to $m = N_v$.

```
static void compute_price_new(..., double *ptprice, double *ptdelta)
```
This function makes the loop over time iterations and the loop over one variance line until convergence. It makes also all memory allocations, initializations and memory deallocations. It returns the value of the price and the delta of the product.

References

[1] Carl Chiarella, Boda Kang, Gunter H. Meyer The evaluation of barrier option prices under stochastic volatility. Computers and Mathematics with Applications 64 (2012) 2034-2048. 1, 2