C++ Project

I. Expression of the linear system

We want to solve the more general equat°:

$$\frac{\partial f}{\partial t}(x,t) = a(x,t) \cdot \frac{\partial^2 f}{\partial x^2}(x,t) + b(x,t) \cdot \frac{\partial f}{\partial x}(x,t) + c(x,t) \cdot f(x,t) + d(x,t)$$

Let's define $L_i^n = a_i^n \cdot \frac{\partial^2 f}{\partial x^2}(x,t) + b_i^n \cdot \frac{\partial f}{\partial x}(x,t) + c_i^n \cdot f_i^n + d_i^n$

- $\forall i \in \{1, 2, \dots, N-1\}$:

$$- \frac{\partial^2 f}{\partial x^2}(x_i, t_n) = \frac{f_{i+1}^n - 2f_i^n + f_{i-1}^n}{dx^2}$$

$$- \frac{\partial f}{\partial x}(x_i, t_n) = \frac{f_{i+1}^n - f_{i-1}^n}{2\,dx}$$

$$L_i^n = a_i^n \cdot \frac{f_{i+1}^n - 2f_i^n + f_{i-1}^n}{dx^2} + b_i^n \cdot \frac{f_{i+1}^n - f_{i-1}^n}{2dx} + c_i^n \cdot f_i^n + d_i^n$$

$$= \left[\frac{a_i^n}{dx^2} - \frac{b_i^n}{2dx}\right] f_{i-1}^n + \left[c_i^n - \frac{2a_i^n}{dx^2}\right] f_i^n + \left[\frac{a_i^n}{dx^2} + \frac{b_i^n}{2dx}\right] \cdot f_{i+1}^n + d_i^n$$

$$= \alpha_i^n \cdot f_{i-1}^n + \beta_i^n \cdot f_i^n + \gamma_i^n \cdot f_{i+1}^n + \varepsilon_i^n$$

With: $\alpha_i^n = \frac{a_i^n}{dx^2} - \frac{b_i^n}{2dx}$ $\quad \beta_i^n = c_i^n - \frac{2a_i^n}{dx^2}$

$\gamma_i^n = \frac{a_i^n}{dx^2} + \frac{b_i^n}{2dx}$ $\quad \varepsilon_i^n = d_i^n$

- $i = 0$:

$$-\frac{\partial^2 f}{\partial x^2}(x_0, t_n) = \frac{f_2^n - 2f_1^n + f_0^n}{dx^2}$$

$$-\frac{\partial^2 f}{\partial x}(x_0, t_n) = \frac{f_1^n - f_0^n}{dx}$$

$$L_0^n = a_0^n \cdot \frac{f_2^n - 2f_1^n + f_0^n}{dx^2} + b_0^n \frac{f_1^n - f_0^n}{dx} + c_0^n \cdot f_0^n + d_0^n$$

$$= \left[\frac{a_0^n}{dx^2} - \frac{b_0^n}{dx} + c_0^n\right] \cdot f_0^n + \left[\frac{b_0^n}{dx} - 2\frac{a_0^n}{dx^2}\right] \cdot f_1^n$$

$$+ \left[\frac{a_0^n}{dx^2}\right] f_2^n + d_0^n$$

$$= \alpha_0^n \cdot f_0^n + \beta_0^n \cdot f_1^n + \gamma_0^n \cdot f_2^n + \varepsilon_0^n$$

With : $\alpha_0^n = \frac{a_0^n}{dx^2} - \frac{b_0^n}{dx} + c_0^n$ $\quad \beta_0^n = \frac{b_0^n}{dx} - 2\frac{a_0^n}{dx^2}$

$\gamma_0^n = \frac{a_0^n}{dx^2}$ $\quad \varepsilon_0^n = d_0^n$

- $i = N$:

$$-\frac{\partial^2 f}{\partial x^2}(x_N, t_n) = \frac{f_N^n - 2f_{N-1}^n + f_{N-2}^n}{dx^2}$$

$$-\frac{\partial f}{\partial x}(x_N, t_n) = \frac{f_N^n - f_{N-1}^n}{dx}$$

$$L_N^n = a_N^n \cdot \frac{f_N^n - 2f_{N-1}^n + f_{N-2}^n}{dx^2} + b_N^n \cdot \frac{f_N^n - f_{N-1}^n}{dx} + c_N^n \cdot f_N^n + d_N^n$$

$$= \left[\frac{a_N^n}{dx^2}\right] \cdot f_{N-2}^n + \left[-2\frac{a_N^n}{dx^2} - \frac{b_N^n}{dx}\right] \cdot f_{N-1}^n$$

$$+ \left[\frac{a_N^n}{dx^2} + \frac{b_N^n}{dx} + c_N^n\right] \cdot f_N^n + d_N^n$$

$$= \alpha_N^n \cdot f_{N-2}^n + \beta_N^n \cdot f_{N-1}^n + \gamma_N^n \cdot f_N^n + \varepsilon_N^n$$

With: $\alpha_N^n = \frac{a_N^n}{dx^2}$ $\quad \beta_N^n = -2\frac{a_N^n}{dx^2} - \frac{b_N^n}{dx}$

$\gamma_N^n = \frac{a_N^n}{dx^2} + \frac{b_N^n}{dx} + c_N^n \quad \varepsilon_N^n = d_N^n$

Let $X^n = \begin{pmatrix} f_0^n \\ f_1^n \\ \vdots \\ f_{N-1}^n \\ f_N^n \end{pmatrix}$ and $L^n = \begin{pmatrix} L_1^n \\ L_2^n \\ \vdots \\ L_{N-1}^n \\ L_N^n \end{pmatrix}$

Using our previous equations we can express L^n as:

$L^n = A^n \cdot X^n + D^n$

With:

$$A^n = \begin{pmatrix} \alpha_0^n & \beta_0^n & \gamma_0^n & 0 & & \\ \alpha_1^n & \beta_1^n & \gamma_1^n & 0 & & (0) \\ 0 & \alpha_2^n & \beta_2^n & \gamma_2^n & & \\ & & & & & \\ & (0) & & & \alpha_{N-1}^n & \beta_{N-1}^n & \gamma_{N-1}^n \\ & & & & \alpha_N^n & \beta_N^n & \gamma_N^n \end{pmatrix}$$

$$D^n = \begin{pmatrix} \varepsilon_0^n \\ \varepsilon_1^n \\ \vdots \\ \varepsilon_{N-1}^n \\ \varepsilon_N^n \end{pmatrix}$$

We can define the theta scheme:

$$\frac{\delta_i^{n+1} - \delta_i^n}{dt} = \theta . L_i^n + (1-\theta) . L_i^{n+1}$$

As: $\frac{X^{n+1} - X^n}{dt} = \theta . L^n + (1-\theta) . L^{n+1}$

$$\Leftrightarrow X^n = X^{n+1} - [\theta . L^n + (1-\theta) . L^{n+1}] . dt$$

$$= X^{n+1} - [\theta . A^n . X^n + \theta . D^n + (1-\theta) . A^{n+1} X^n + (1-\theta) . D^{n+1}] dt$$

$$= [\bar{\bar{I}}_{N+1} - (1-\theta) . A^{n+1} . dt] . X^{n+1} - \theta . A^n . X^n . dt$$

$$- \theta . D^n dt - (1-\theta) . D^{n+1} dt$$

$$\Leftrightarrow [\bar{\bar{I}}_{N+1} + \theta . A^n . dt] . X^n = [\bar{\bar{I}}_{N+1} - (1-\theta) . A^{n+1} . dt] . X^{n+1}$$

$$+ [- \theta . D^n . dt - (1-\theta) . D^{n+1} . dt]$$

$$\Leftrightarrow A'^n . X^n = B^n . X^{n+1} + E^n = F^n$$

With:
$$\begin{cases} A'^n = \bar{\bar{I}}_{N+1} + \theta . A^n . dt \\ B^n = \bar{\bar{I}}_{N+1} - (1-\theta) . A^{n+1} . dt \\ E^n = - \theta . D^n . dt - (1-\theta) . D^{n+1} . dt \\ F^n = B^n . X^{n+1} + E^n \end{cases}$$

II Solve the system

We want to solve the system $AX = B$ where:

$$A = \begin{pmatrix} d_0 & u_0 & e & 0 & & \\ l_0 & d_1 & u_1 & 0 & & \\ 0 & l_1 & d_2 & u_2 & & (0) \\ & & & & & \\ & & & & & \\ & (0) & & & l_{n-3} & d_{n-2} & u_{n-2} \\ & & & & f & l_{n-2} & d_{n-1} \end{pmatrix}$$

To solve the system we need to make a LU decomposition of A. ($\Leftrightarrow A = LU$)

$$L = \begin{pmatrix} 1 & & & & & \\ l_0 & 1 & & & & (0) \\ 0 & l_1 & 1 & & & \\ & & & & & \\ & & & & & \\ (0) & & & l_{n-3} & 1 & \\ & & & w & l_{n-2} & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} i_0 & j_0 & e & 0 & & \\ 0 & i_1 & j_1 & 0 & & (0) \\ 0 & 0 & i_2 & j_2 & & \\ & & & & & \\ & & & & & \\ & (0) & & & i_{n-2} & j_{n-2} \\ & & & & 0 & i_{n-1} \end{pmatrix}$$

$$LU = \begin{pmatrix} i_0 & j_0 & e & 0 \\ h_0 i_0 & h_0 j_0 + i_1 & h_0 e + j_1 & 0 \\ 0 & h_1 \cdot i_1 & h_1 \cdot j_1 + i_2 & j_3 \\ & & & (0) \\ & (0) & & \\ & & h_{n-3} \cdot i_{n-3} & h_{n-3} \cdot j_{n-3} + i_{n-2} & j_{n-2} \\ & & w \cdot i_{n-3} & w \cdot j_{n-3} + h_{n-2} \cdot i_{n-2} & h_{n-2} \cdot j_{n-2} + i_{n-1} \end{pmatrix}$$

We can compute the coefficients of matrices L and U as:

$$\begin{cases} i_0 = d_0 \\ j_0 = u_0 \\ h_0 = \frac{l_0}{i_0} \end{cases} \qquad \begin{cases} i_1 = d_1 - h_0 \cdot j_0 \\ j_1 = u_1 - h_0 \cdot e \end{cases}$$

$\forall m \in \{1, 2, \dots, N-2\}$:

$$\begin{cases} h_m = l_m / i_m \\ i_{m+1} = d_m - h_m \cdot j_m \\ j_{m+1} = u_m \end{cases}$$

$$\begin{cases} w = f / i_{n-3} \\ h_{n-2} = \frac{1}{i_{n-2}} [l_{n-2} - w \cdot j_{n-3}] \\ i_{n-1} = d_{n-1} - h_{n-2} \cdot j_{n-2} \end{cases}$$

Once matrices L and U are determined, we can solve $AX = B$ by solving $LY = B$ and $UX = Y$.

$LY = B \iff$

$$\begin{pmatrix} 1 & & & & \\ h_0 & 1 & & & \\ 0 & h_1 & 1 & & \\ & & \ddots & \ddots & \ddots \\ (0) & & & h_{n-3} & 1 & 0 \\ & & & w & h_{n-2} & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix}$$

$$\begin{cases} y_0 = b_0 \\ h_0 y_0 + y_1 = b_1 \iff y_1 = b_1 - h_0 y_0 \\ \vdots \\ h_{n-3} y_{n-3} + y_{n-2} = b_{n-2} \iff y_{n-2} = b_{n-2} - h_{n-3} y_{n-3} \\ w y_{n-3} + h_{n-2} y_{n-2} + y_{n-1} = b_{n-1} \\ \iff y_{n-1} = b_{n-1} - h_{n-2} y_{n-2} - w y_{n-3} \end{cases}$$

$UX = Y \iff$

$$\begin{pmatrix} i_0 & j_0 & e & & 0 \\ 0 & i_1 & j_1 & & \\ & \ddots & \ddots & \ddots & \\ & & & i_{n-2} & j_{n-2} \\ (0) & & & 0 & i_{n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix}$$

$$\begin{cases} i_{n-1} x_{n-1} = y_{n-1} \iff x_{n-1} = y_{n-1} / i_{n-1} \\ i_{n-2} x_{n-2} + j_{n-2} x_{n-1} = y_{n-2} \iff x_{n-2} = \dfrac{y_{n-2} - x_{n-1} j_{n-2}}{i_{n-2}} \\ \vdots \\ i_1 x_1 + j_1 x_2 = y_1 \iff x_1 = \dfrac{y_1 - j_1 x_2}{i_1} \\ i_0 x_0 + j_0 x_1 + e x_2 = y_0 \iff x_0 = \dfrac{y_0 - j_0 x_1 - e x_2}{i_0} \end{cases}$$

III Boundary Conditions

We can impose the value of $\frac{\partial^2 f}{\partial x^2}(x_0, t_n)$, $\frac{\partial f}{\partial x}(x_0, t_n)$ or $f(x_0, t_n)$ when solving our system (and we can d the same for x_N).

These will slightly change the system:

- First derivative set (lower)

$$\frac{\partial f}{\partial x}(x_0, t_n) = f'^n_0 \quad \text{fixed}$$

$$\rightarrow \alpha_0^n = \frac{a_0^n}{dx^2} + c_0^n \qquad \beta_0^n = -2\frac{a_0^n}{dx^2}$$

$$\gamma_0^n = \frac{a_0^n}{dx^2} \qquad \varepsilon_0^n = d_0^n + b_0^n \cdot f'^n_0$$

- First derivative set (upper)

$$\frac{\partial f}{\partial x}(x_N, t_n) = f'^n_N \quad \text{fixed}$$

$$\alpha_N^n = \frac{a_N^n}{dx^2} \qquad \beta_N^n = -\frac{b_N^n}{dx}$$

$$\gamma_N^n = \frac{a_N^n}{dx^2} + c_N^n \qquad \varepsilon_N^n = d_N^n + b_N^n \cdot f'^n_N$$

- Second derivative set (lower)

$$\frac{\partial^2 f}{\partial x^2}(x_0, t_n) = f''^n_0 \quad \text{fixed}$$

$$\alpha_0^n = -\frac{b_0^n}{dx} + c_0^n \qquad \beta_0^n = \frac{b_0^n}{dx}$$

$$\gamma_0^n = 0 \qquad \varepsilon_0^n = d_0^n + a_0^n \cdot f''^n_0$$

- Second derivative set (upper)

$$\frac{\partial^2 f}{\partial x^2}(x_N, t_n) = f''^n_N \quad \text{fixed}$$

$$\alpha_N^n = 0 \qquad \beta_N^n = -\frac{b_N^n}{dx}$$

$$\gamma_N^n = \frac{b_N^n}{dx} + c_N^n \qquad \varepsilon_N^n = d_N^n + a_N^n \cdot f''^n_N$$

- Both set (lower)

$$\frac{\partial^2 f}{\partial x^2}(x_0, t_n) = f''^n_0 \qquad \frac{\partial f}{\partial x}(x_0, t_n) = f'^n_0 \quad \text{fixed}$$

$$\alpha_0^n = c_0^n \qquad \beta_0^n = 0$$

$$\gamma_0^n = 0 \qquad \varepsilon_0^n = a_0^n \cdot f''^n_0 + b_0^n \cdot f'^n_0 + d_0^n$$

- Both set (upper)

$$\frac{\partial^2 f}{\partial x^2}(x_N, t_n) = f''^n_N \qquad \frac{\partial f}{\partial x}(x_N, t_n) = f'^n_N \quad \text{fixed}$$

$$\alpha_N^n = 0 \qquad \beta_N^n = 0$$

$$\gamma_N^n = c_N^n \qquad \varepsilon_N^n = a_N^n \cdot f''^n_N + b_N^n \cdot f'^n_N + d_N^n$$

If we set the value of f_0^n (or f_N^n) directly, we need to adjust the size of our system accordingly.

- Value set (lower)

$$A^n = \begin{pmatrix} \beta_1^n & \gamma_1^n & 0 & & \\ \alpha_2^n & \beta_2^n & \gamma_2^n & & (0) \\ & & & & \\ & (0) & & \alpha_{N-1}^n & \beta_{N-1}^n & \gamma_{N-1}^n \\ & & & \alpha_N & \beta_N & \gamma_N \end{pmatrix}$$

$$X^n = \begin{pmatrix} f_1^n \\ f_2^n \\ \vdots \\ f_{N-1}^n \\ f_N \end{pmatrix} \qquad D^n = \begin{pmatrix} \alpha_1^n f_0^n + d_1^n \\ \varepsilon_2^n \\ \\ \varepsilon_{N-1}^n \\ \varepsilon_N \end{pmatrix}$$

- Value set (upper)

$$A^n = \begin{pmatrix} \alpha_0^n & \beta_0^n & \gamma_0^n & & \\ \alpha_1^n & \beta_1^n & \gamma_1^n & & (0) \\ & & & & \\ & (0) & & \alpha_{N-2}^n & \beta_{N-2}^n & \gamma_{N-2}^n \\ & & & 0 & \alpha_{N-1}^n & \beta_{N-1}^n \end{pmatrix}$$

$$X^n = \begin{pmatrix} f_0^n \\ f_1^n \\ \vdots \\ f_{N-2}^n \\ f_{N-1}^n \end{pmatrix} \qquad D^n = \begin{pmatrix} \varepsilon_0^n \\ \varepsilon_1^n \\ \vdots \\ \varepsilon_{N-2}^n \\ \gamma_{N-1}^n f_N + d_{N-1}^n \end{pmatrix}$$

IV. Specific models

A. The Black - Scholes model

$$a_i^n = -\frac{1}{2}\,\sigma_i^{n\,2} \qquad b_i^n = \left(\frac{1}{2}\,\sigma_i^{n\,2} - r_i^n\right)$$

$$c_i^n = r_i^n \qquad d_i^n = 0$$

B. Coefficients constant in space

$\forall i \in \{1, \dots, N-1\}$: $a_i^n = a^n$ $\quad b_i^n = b^n$

$c_i^n = c^n$ $\quad d_i^n = d^n$

$\alpha_i^n = \alpha^n$ $\quad \beta_i^n = \beta^n$ $\quad \gamma_i^n = \gamma^n$

$$A^n = \begin{pmatrix} \alpha_0 & \beta_0 & \gamma_0 & 0 & & & \\ \alpha & \beta & \gamma & 0 & & (0) & \\ 0 & \alpha & \beta & \gamma & & & \\ & & & & \ddots & & \\ & (0) & & & \alpha & \beta & \gamma \\ & & & & \alpha_N & \beta_N & \gamma_N \end{pmatrix}$$

C. Coefficients constant in time

$a_i^n = a_i$ $\quad b_i^n = b_i$ $\quad c_i^n = c_i$ $\quad d_i^n = d_i$

$A^n = A$

$A'^n = A' = \bar{\bar{I}}_{N+1} + A.\theta.dt$

$B^n = B = \bar{\bar{I}}_{N+1} - (1-\theta).A.dt$

$D^n = D$

$E^n = -D.dt = E$

$F^n = B.X^{n+1} + E = F$

V. C++ implementation

First we implemented a class tridiagExtended to represent the matrices of our system.

All matrices A, B, A', F have the same shape as A, which is tridiagonal with 2 added coefficients.

$$M_{(N\times N)} = \begin{pmatrix} d_0 & u_0 & e & 0 & & & \\ l_0 & d_1 & u_1 & 0 & & (0) & \\ 0 & l_1 & d_2 & u_2 & & & \\ & & & \ddots & & & \\ & (0) & & & & & \\ & & & & l_{n-3} & d_{n-2} & u_{n-2} \\ & & & & f & l_{n-2} & d_{n-1} \end{pmatrix}$$

compared to a traditional matrix

Using this representation allows us to avoid storing a lot of empty elements, and to more efficiently do a product between a matrix and a vector (complexity $o(N)$ with this rep

We also implement the method to solve the system $MX=B$ found in section II (complexity $o(N)$, no matrix inversion needed).

We then implemented a PDE_solver class taking as inputs all of the variables needed to solve the equation (a, b, c, d, dt, T, theta, payoff, upper and lower bounds).

From this we generate a space/time grid initiated with the payoff as its values on the last time step.

To fill the grid we go backward in time starting at maturity T and computing the values of A' and F and solving the system at each time step.

We use various functions to generate the coefficients of A and the different matrices and vectors used in the solver.

Finally we implemented a BS-solver class that directly inherits from PDE_solver that takes S, σ and r as inputs and initiates a PDE-solver with the log of the price as the space axis and computes the coefficients a, b, c, d.
We also implement specific financial functions in this class to recover the greeks.

VI Possible improvements

The first improvement to make to the solver would be to add the possibility to have space and time varying input coefficients a, b, c and d.
They could take the form of a matrix for coefficients varying in both space and time, a column vector for inputs varying in space only, a row vector for coefficients varying in time only, and a float for constant coefficients.
We would then need to compute coefficients α_i^n, β_i^n, γ_i^n at each point in time and space.
Lastly instead of computing A', B and E once, we would compute them at each time step and use them to find the value of our function at each associated time step.

We focused a lot on improving the speed of the solver and are happy with our implementation of the LU decomposition algorithm, reducing the complexity of the problem to $O(N)$ and reducing the runtime of the algorithm compared to the first version of the solver where we inverted A'.
We could save some memory by not storing coefficients $\alpha_i, \beta_i, \gamma_i$ $N-1$ times when they are constant in space, and save some computing power when making operation on matrix A.