# Transparent boundary conditions for solving numerically the Black-Scholes equation

Sébastien Tordeux Ekaterina Voltchkova

February 18, 2016

# Premia 18

## 1 Introduction

Numerical solution of the Black-Scholes type pricing equation requires a localization on a bounded computational domain and, in consequence, a choice of boundary conditions. The standard approach usually proposed in the textbooks on financial engineering and academic literature is to take Dirichlet or Neumann boundary conditions deduced from the shape of the payoff function. Equivalently, one first subtracts the payoff from the solution, and then takes zero boundary conditions on the excess to payoff. It can be shown that the localization error in these cases converges to zero when the computational domain increases. Note that these conditions are only asymptotic, and the localization error may be large for a fixed finite domaine.

We propose a different approach using the so-called transparent or absorbing boundary conditions (see e.g. [**?**],[**?**], [**?**] [**?**]). It is well known that the Black-Scholes PDE may be transformed by a change of variables to the following parabolic equation with constant coefficients:

$$u_t = a u_{xx} + \mu u_x \tag{1}$$

where $a > 0$, $\mu \in \mathbb{R}$. We first derive the *exact* condition on the boundary for this PDE given by a Dirichlet-to-Neumann operator in the form

$$\frac{\partial u}{\partial n} + \mathcal{S}_- u = g_- \quad \text{at } x_{min}, \qquad \frac{\partial u}{\partial n} + \mathcal{S}_+ u = g_+ \quad \text{at } x_{max} \tag{2}$$

where $\mathcal{S}_-$ and $\mathcal{S}_+$ are non-local in time operators. One may discretize these conditions directly and incorporate them in the standard finite difference (or finite element) scheme. The localization error is then only due to the discretization of (2) and does not depend on the size of the domain. The drawback of

this method is that the boundary conditions at $t_{n+1}$ involve all previous values of the solution on the boundary, and not only $u^n$ and $u^{n+1}$.

To avoid this, we approximate integral operators $\mathcal{S}_-$ and $\mathcal{S}_+$ by local differential operators involving time derivatives. After discretization, we obtain a finite difference scheme with a small localization error even on a small computational domain.

We have implemented transparent boundary conditions in Premia for European and American calls and puts. This method may also be used for barrier options, as well as for other standard payoffs.

## 2 Black-Scholes equation for European option prices

We are interested in the numerical computation of the solution of the Black-Scholes equation

$$\partial_\tau V(S,\tau) + \frac{\sigma^2 S^2}{2}\partial_S^2 V(S,\tau) + rS\,\partial_S V(S,\tau) - r\,V(S,\tau) = 0$$

defined on one of the following domains in variable $S$:

$$0 < S < +\infty \tag{3}$$

$$0 < S < U \tag{4}$$

$$L < S < +\infty \tag{5}$$

The first case corresponds to a European vanilla option, while the second and the third cases correspond to up-and-out and down-and-out barrier options. The domain of definition in time is $[0, T)$ where $T$ is the maturity of the option. At $\tau = T$, we have a terminal condition given by the payoff function

$$V(T, S) = \Phi(S).$$

The localization technique described in this note applies to all standard payoff functions.

## 3 Toward advection diffusion equation

We perform a standard change of variables

$$x = \ln(S/S_0), \quad t = T - \tau, \quad u(x,t) = e^{rt}V(S,\tau)$$

which leads to a forward PDE with constant coefficients

$$\partial_t u(x,t) = \frac{\sigma^2}{2}\partial_x^2 u(x,t) + \mu\,\partial_x u(x,t) \quad \text{with } \mu = r - \frac{\sigma^2}{2}. \tag{6}$$

After this change of variables, the terminal condition becomes an initial condition at $t = 0$

$$u(0, x) = u_0(x)$$

and the domain of definition in $x$ is transformed to an unbounded or semi-bounded domain:

$$-\infty < x < +\infty \tag{7}$$
$$-\infty < x < \ln(U/S_0) \tag{8}$$
$$\ln(L/S_0) < x < +\infty \tag{9}$$

# 4 Localization on a bounded computational domain

In order to apply standard methods of numerical solution of PDEs, such as finite difference or finite element schemes, we need to localize the domain of definition in $x$ to a bounded interval

$$x \in (x_-, x_+).$$

The standard approach consists in choosing the computational domain $(x_-, x_+)$ "sufficiently large" and imposing Dirichlet or Neumann boundary conditions based on the asymptotics of the solution. For instance, in the case of a call option, we have the following asymptotic behavior of the solution:

$$V(S, \tau) \simeq S - K\, e^{-r(T-\tau)}, \qquad S \longrightarrow +\infty,$$
$$V(S, \tau) \simeq 0, \qquad S \longrightarrow 0^+.$$

This motivates the following choice of boundary conditions:

$$\textbf{Dirichlet:} \qquad V(S_-, \tau) = 0 \quad \Leftrightarrow \quad u(x_-, t) = 0 \tag{10}$$
$$V(S_+, \tau) = S_+ - Ke^{-r(T-\tau)} \quad \Leftrightarrow \quad u(x_+, t) = S_0\, e^{x_+ + rt} - K. \tag{11}$$

or

$$\textbf{Neumann:} \qquad \partial_S V(S_-, \tau) = 0 \quad \Leftrightarrow \quad \partial_x u(x_-, t) = 0 \tag{12}$$
$$\partial_S V(S_+, \tau) = 1 \quad \Leftrightarrow \quad \partial_x u(x_+, t) = S_0\, e^{x_+ + rt}. \tag{13}$$

It was proved that the localization error due to these boundary conditions goes to zero exponentially when the size of the computational domain goes to infinity. However, these results do not indicate whether it is better to take Dirichlet or Neumann conditions nor how to choose $(x_-, x_+)$ "sufficiently large" in practice. One often suggests to take this interval in relation with the standard deviation of the Brownian motion for the period $[0, T]$. That is, $x_\pm = \pm k\sigma\sqrt{T}$ with $k = 3$, for example.

In the next section, we present a different approach to the choice of the artificial boundary conditions. This approach uses the so-called transparent boundary conditions which have the following advantages.

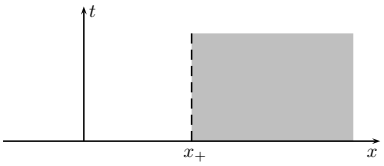


Figure 1:

- Transparent boundary conditions may be used for any interval $(x_-, x_+)$, even small, provided it contains the singularities of the payoff.
- They provide *exact* boundary conditions for the localized PDE (and not only asymptotically exact). There is still a small numerical error due to the discretization of these conditions.
- Finally, transparent boundary conditions are almost as easy to implement as Neumann or Dirichlet conditions.

## 5 Transparent boundary conditions

The idea of transparent boundary conditions is the following. Take an arbitrary $x_+ \in \mathbb{R}$ and consider equation (6) on $\{t > 0\} \times \{x > x_+\}$ (the gray domain on Fig. 1). It is well known that the initial condition on $\{t = 0\} \times \{x > x_+\}$ and boundary values on $\{t > 0\} \times \{x = x_+\}$ define completely the solution of the PDE in this domain. In particular, they define the normal derivative at the boundary: $\partial_x u(x_+, t)$. So, at least formally, there exists an operator which relates the initial and boundary values of the solution to its normal derivative on the boundary:

$$\partial_x u(x_+, t) = \mathcal{S}_+[u(x_+, t)_{t>0},\ u(x, 0)_{x>x_+}] \tag{14}$$

In fact, due to the constant coefficients of the PDE and simple form of initial data, the operator $\mathcal{S}_+$ may be found explicitly. The idea is then to use (14) as a boundary condition when solving for $x \le x_+$. Of course, the boundary values $u(x_+, t)_{t>0}$ are not known but we have a kind of mixed boundary condition relating Dirichlet and Neumann values of the solution on the boundary. For this reason, the operator $\mathcal{S}_+$ is also called Dirichlet-to-Neumann operator.

Note that (14) is an exact relation satisfied by the true solution of the PDE, and not by its asymptotics as it is the case for (10)–(13).

Boundary condition on $\{x = x_-\}$ is obtained by the same reasoning applied to the domain $\{x < x_-\}$. In the next section, we derive the explicit form of the Dirichlet-to-Neumann operator.

### 5.1 Dirichlet-to-Neumann operator

We start by the case of a right-open domain. The idea is to apply the Laplace transform with respect to the time variable, defined as

$$\widehat{u}(x,p) = \int_0^{+\infty} u(x,t)\ \exp(-pt)dt,$$

to the PDE (6). Note that the time derivative is transformed to

$$\widehat{\partial_t u}(x,p) = p\widehat{u}(x,p) + u(x,0).$$

To get rid of the boundary term $u(x,0)$, we introduce

$$u_H(x,t) = u(x,t) - \varphi(x,t)$$

where $\varphi(x,t)$ is a solution of (6) with $\varphi(x,0) = u(x,0)$ on $\{x > x_+\}$. Here $x_+$ is arbitrary but such that all singularities of the payoff $u_0(x)$ are on the left of $x_+$ (in the case of calls and puts it simply means that $x_+ > \ln(K/S_0)$). It is important that $u(x,0)$ has no singularity on $\{x > x_+\}$ because, in this case, $\varphi(x,t)$ has a simple explicit form. For example,

$$\text{call:}\quad \varphi(x,t) = S_0 e^{x+rt} - K, \qquad \text{put:}\quad \varphi(x,t) = 0.$$

By construction, $u_H(x,t)$ satisfies, for $x > x_+$,

$$\begin{aligned} \partial_t u_H(x,t) &= \frac{\sigma^2}{2}\partial^2 u_H(x,t) + \mu\ \partial_x u_H(x,t) \\ u_H(x,0) &= 0 \end{aligned} \tag{15}$$

Applying the Laplace transform to (15), we obtain

$$p\ \widehat{u}_H(x,p) = \frac{\sigma^2}{2}\ \partial_x^2 \widehat{u}_H(x,p) + \mu\ \partial_x \widehat{u}_H(x,p).$$

This is a linear ordinary differential equation with constant coefficients with respect to the $x$ variable. Its solution which does not explode when $x$ goes to infinity has the form

$$\widehat{u}_H(x,p)\ =\ A\ \exp\Big(-\Big(\frac{\mu}{\sigma^2} + \frac{|\mu|}{\sigma^2}\sqrt{1+\frac{2p\sigma^2}{\mu^2}}\Big)x\Big).$$

Therefore, in the Laplace domain, we have the following relation between $u_H$ and $\partial_x u_H$:

$$\partial_x \widehat{u}_H(x_+,p) = -\Big(\frac{\mu}{\sigma^2} + \frac{|\mu|}{\sigma^2}\sqrt{1+\frac{2p\sigma^2}{\mu^2}}\Big)\widehat{u}_H(x_+,p). \tag{16}$$

| function | Laplace transform |
|---|---|
| $\frac{1}{\sqrt{\pi t}}$ | $\frac{1}{\sqrt{p}}$ |
| $\frac{1}{\sqrt{\pi}}\int_0^t \frac{u(s)}{\sqrt{t-s}}ds$ | $\frac{\widehat{u}(p)}{\sqrt{p}}$ |
| $\frac{1}{\sqrt{\pi}}\int_0^t \frac{e^{-a(t-s)}}{\sqrt{t-s}}u(s)ds$ | $\frac{\widehat{u}(p)}{\sqrt{p+a}}$ |
| $\frac{1}{\sqrt{\pi}}\Big(\partial_t + a\Big)\int_0^t \frac{e^{-a(t-s)}}{\sqrt{t-s}}u(s)ds$ | $\sqrt{p+a}\,\widehat{u}(p)$ |

Table 1: Some useful Laplace transforms.

Going back to the time variable, we obtain[1]

$$\begin{aligned}\partial_x u_H(x_+,t) &= -\frac{\mu}{\sigma^2}\,u_H(x_+,t) - \Big(\frac{2}{\sigma^2\pi}\Big)^{\frac{1}{2}}\Big(\partial_t + \frac{\mu^2}{2\sigma^2}\Big)\int_0^t \frac{e^{-\frac{\mu^2}{2\sigma^2}(t-s)}u_H(x_+,s)}{\sqrt{t-s}}ds\\ &\equiv S_+ u_H(x_+,t)\end{aligned}$$

That is, the Dirichlet-to-Neumann operator is a multiplication operator in the Laplace domain and an integro-differential operator in time domain. In terms of the non-homogeneous solution, we have

$$\partial_x u(x_+,t) = S_+ u(x_+,t) + (\partial_x\varphi(x_+,t) - S_+\varphi(x_+,t)) \tag{17}$$

Using the same arguments for $\{x < x_-\}$, we obtain

$$\partial_x u(x_-,t) = S_- u(x_-,t) + (\partial_x\psi(x_-,t) - S_-\psi(x_-,t)) \tag{18}$$

where

$$S_- u(t) := -\frac{\mu}{\sigma^2}\,u(t) + \Big(\frac{2}{\sigma^2\pi}\Big)^{\frac{1}{2}}\Big(\partial_t + \frac{\mu^2}{2\sigma^2}\Big)\int_0^t \frac{e^{-\frac{\mu^2}{2\sigma^2}(t-s)}u(s)}{\sqrt{t-s}}ds$$

and $\psi(x,t)$ is a solution of (6) with $\psi(x,0) = u(x,0)$ on $\{x < x_-\}$. For example, if $x_- < \ln(K/S_0)$,

$$\text{call:}\quad \psi(x,t) = 0, \qquad \text{put:}\quad \psi(x,t) = K - S_0 e^{x+rt}.$$

## 6 A local transparent boundary condition

It is possible to use exact boundary conditions (17) and (18) directly by discretizing the integro-differential operators $S_+$ and $S_-$. This gives a very good precision of the solution on the boundary. Indeed, the error is only due to the discretization (e.g., trapezoidal rule).

[1] This may be checked directly. To help the reader, we recall the Laplace transforms of some useful functions in Table 1.

However, the formulas to implement are rather messy, and, more importantly, at each time iteration, boundary conditions (17) and (18) involve *all* previous values of the solution on the boundary, and not only $u^n$ and $u^{n-1}$. The obtained discretization scheme is not local in time.

In the next sections, we propose an approximation of the exact boundary conditions based on approximation of the Laplace symbol of the Dirichlet-to-Neumann operator by rational functions.

## 6.1 Approximating the Laplace symbol by a rational function

Recall that the Laplace symbol of the Dirichlet-to-Neumann operator (that is, the multiplication factor in (16)) involves the square root function which corresponds to an integro-differential operator in the time domain. To approximate this operator by a local one, the idea is to approximate its Laplace symbol by simpler functions corresponding to differential operators in time variable.

The simplest choice would be a polynomial approximation. However, it does not work well in practice.[2] Instead, we propose to approximate the symbol by a rational function.

We start by approximating $\sqrt{z}$ by a rational function $Q_n(z)$ of the form

$$Q_n(z) = \frac{q_1 z}{z + z_1} + \frac{q_2 z}{z + z_2} + \cdots + \frac{q_n z}{z + z_n}$$

where the coefficients $z_i$ and $q_i$ have to be strictly positive in order $Q_n$ to be bounded on $\mathbb{R}_+$, the domain of definition of $\sqrt{z}$. We first fix interpolation points $z_j$. The choice is rather arbitrary and is based on numerical tests of the quality of the approximation that we have performed. We take $z_i = 1, 2, 1/2, 4, 1/4, \ldots$. The number of these interpolation points may be chosen by the user and can be different on the right and on the left boundary (see parameters $m_r$ and $m_l$ below, equations (19) and (20)). In practice, $n$ will usually be less than 10.

Then we compute coefficients $q_j$ so that $Q_n(z)$ coincides with $\sqrt{z}$ at $z = z_j$:

$$Q_n(z_j) = \sqrt{z_j}, \qquad j = 1, \ldots, n.$$

(Note that, by construction, we also have $Q_n(z) = \sqrt{z}$ at $z = 0$). This leads to a linear system that can be easily solved:

$$\text{Find } q \in \mathbb{R}^n : Mq = F \quad \text{ with } M_{i,j} = \frac{z_i}{z_i + z_j} \text{ and } F_i = \sqrt{z_i}.$$

The coefficients $M_{i,j}$ and $F_i$ are computed only once for $i, j = 1, \ldots, N_{max}$ where we put $N_{max} = 30$ (more interpolation points are hardly useful). They

[2] Briefly speaking, the reason is that the radius of convergence of polynomial approximations of the square root (we tried Taylor expansions at different points) is small. This implies a poor approximation of the symbol for high frequencies (large $p$). In the original variable, this corresponds to small $t$, that is, precisely the region of interest. Numerical experiments showed that polynomial approximation does not perform better than Dirichlet or Neumann boundary conditions. Increasing the degree of the approximation does not improve the error at the boundary.

are written in the file "InterpolationParameters.txt" which remains constant.[3] On the contrary, vector $q$ is computed each time we use transparent boundary conditions because it depends on the user-chosen parameter $n$. To compute $q$, we use a direct LU-solver for the system $Mq = F$ (recall that the size $n$ of this system is small, so it is not time consuming).

The Laplace symbol of the Dirichlet-to-Neumann operator at $x = x_+$ is now approximated in the following way:

$$\begin{aligned} -\Big(\frac{\mu}{\sigma^2} + \frac{|\mu|}{\sigma^2}\sqrt{1+\frac{2p\sigma^2}{\mu^2}}\Big) &\approx -\Big(\frac{\mu}{\sigma^2} + \frac{\sqrt{2}}{\sigma} Q_{m_r}\Big(p + \frac{\mu^2}{2\sigma^2}\Big)\Big) \\ &= \eta_0 + \frac{\beta_1}{p+\delta_1} + \cdots + \frac{\beta_{m_r}}{p+\delta_{m_r}}. \end{aligned} \tag{19}$$

A straightforward computation yields

$$\eta_0 = -\frac{\mu}{\sigma^2} - \frac{\sqrt{2}}{\sigma}\sum_{j=1}^{m_r} q_j, \qquad \beta_j = \frac{\sqrt{2}}{\sigma} q_j z_j, \qquad \delta_j = \frac{\mu^2}{2\sigma^2} + z_j.$$

Similarly, at $x = x_-$, we get

$$\begin{aligned} -\Big(\frac{\mu}{\sigma^2} - \frac{|\mu|}{\sigma^2}\sqrt{1+\frac{2p\sigma^2}{\mu^2}}\Big) &\approx -\Big(\frac{\mu}{\sigma^2} - \frac{\sqrt{2}}{\sigma} Q_{m_l}\Big(p + \frac{\mu^2}{2\sigma^2}\Big)\Big) \\ &= \xi_0 + \frac{\alpha_1}{p+\gamma_1} + \cdots + \frac{\alpha_{m_l}}{p+\gamma_{m_l}} \end{aligned} \tag{20}$$

with

$$\xi_0 = -\frac{\mu}{\sigma^2} + \frac{\sqrt{2}}{\sigma}\sum_{j=1}^{m_r} q_j, \qquad \alpha_j = -\frac{\sqrt{2}}{\sigma} q_j z_j, \qquad \gamma_j = \frac{\mu^2}{2\sigma^2} + z_j.$$

## 6.2 Approximate transparent boundary conditions

Using the rational approximation of the Laplace symbol described above, we obtain the following relation at $x = x_+$ in Laplace domain:

$$\partial_x \widehat{u}_H(x_+, p) = \Big(\eta_0 + \frac{\beta_1}{p+\delta_1} + \cdots + \frac{\beta_{m_r}}{p+\delta_{m_r}}\Big)\widehat{u}_H(x_+, p).$$

Substituting $u_H = u - \varphi$ and going back to the time variable, we obtain the following approximate boundary condition:

$$\begin{aligned} \partial_x u(x_+,t) &= \eta_0 u(x_+,t) + \beta_1\rho_1(t) + \cdots + \beta_{m_r}\rho_{m_r}(t) \\ &+ (\partial_x\varphi(x_+,t) - \eta_0\varphi(x_+,t) - \beta_1\omega_1(t) - \cdots - \beta_{m_r}\omega_{m_r}(t)) \end{aligned} \tag{21}$$

[3] These coefficients should be recomputed only if we want to change interpolation points $z_j$.

where the auxiliary functions $\rho_i$ and $\omega_i$ satisfy, respectively, the following ODEs:

$$\begin{aligned} \partial_t \rho_j(t) + \delta_j \rho_j(t) &= u(x_+,t), \quad \rho_j(0) = 0, \qquad j = 1,\dots,m_r \qquad (22) \\ \partial_t \omega_j(t) + \delta_j \omega_j(t) &= \varphi(x_+,t), \quad \omega_j(0) = 0, \qquad j = 1,\dots,m_r \qquad (23) \end{aligned}$$

Indeed, applying the Laplace transform to (22), we get

$$(p+\delta_j)\widehat{\rho}_j(p) = \widehat{u}(x_+,p) \qquad \text{or} \qquad \widehat{\rho}_j(p) = \frac{1}{p+\delta_j}\widehat{u}(x_+,p).$$

Since the Laplace transform is one-to-one, $\rho_j$ is the inverse of $\frac{1}{p+\delta_j}\widehat{u}$. The same reasoning applies to $\omega_j$.

To write (21)–(23) in a short way, we may introduce the notation

$$\tilde{S}_+ v(t) \equiv \eta_0 v(t) + \beta_1 f_1^v(t) + \cdots + \beta_{m_r} f_{m_r}^v(t)$$

with the auxiliary functions $f_j^v$ satisfying

$$\partial_t f_j^v(t) + \delta_j f_j^v(t) = v(t), \quad f_j^v(0) = 0, \qquad j = 1,\dots,m_r$$

Then (21) becomes

$$\partial_x u(x_+,t) = \tilde{S}_+ u(x_+,t) + (\partial_x \varphi(x_+,t) - \tilde{S}_+ \varphi(x_+,t)) \qquad (24)$$

with $f_j^u \equiv \rho_j$ and $f_j^\varphi \equiv \omega_j$. Relation (24) is the approximate transparent boundary condition at $x = x_+$ that we will use instead of usual Dirichlet or Neumann conditions.

Similarly, the approximate transparent boundary condition at $x = x_-$ takes the form

$$\partial_x u(x_-,t) = \tilde{S}_- u(x_-,t) + (\partial_x \psi(x_-,t) - \tilde{S}_- \psi(x_-,t)) \qquad (25)$$

with

$$\tilde{S}_- v(t) \equiv \xi_0 v(t) + \alpha_1 g_1^v(t) + \cdots + \alpha_{m_l} g_{m_l}^v(t)$$

where the auxiliary functions $g_j^v$ are solutions of the ODEs

$$\partial_t g_j^v(t) + \gamma_j g_j^v(t) = v(t), \quad g_j^v(0) = 0, \qquad j = 1,\dots,m_l.$$

For simplicity, we denote $g_j^v \equiv \lambda_j$ and $g_j^\psi \equiv \mu_j$, so that (25) may also be expanded as follows:

$$\begin{aligned} \partial_x u(x_-,t) &= \xi_0 u(x_-,t) + \alpha_1 \lambda_1(t) + \cdots + \alpha_{m_l} \lambda_{m_l}(t) \\ &+ (\partial_x \psi(x_-,t) - \xi_0 \psi(x_-,t) - \alpha_1 \mu_1(t) - \cdots - \alpha_{m_l} \mu_{m_l}(t)) \qquad (26) \end{aligned}$$

where

$$\begin{aligned} \partial_t \lambda_i(t) + \gamma_i \lambda_i(t) &= u(x_-,t), \quad \lambda_i(0) = 0, \qquad i = 1,\dots,m_l \qquad (27) \\ \partial_t \mu_i(t) + \gamma_i \mu_i(t) &= \psi(x_-,t), \quad \mu_i(0) = 0, \qquad i = 1,\dots,m_l \qquad (28) \end{aligned}$$

# 7 Discretization of the approximate transparent boundary conditions

In this section, we explain how to incorporate boundary conditions (24) and (25) into a standard finite difference $\theta$-scheme for equation (6).

We introduce a regular grid on $[x_-, x_+] \times [0, T]$:

$$x_i = x_- + i\Delta x, \qquad \Delta x = (x_+ - x_-)/(N-1),$$
$$t_n = n\Delta t, \qquad \Delta t = T/N_{\text{time}}.$$

We denote by $u_i^n = u(x_i, t_n)$ the unknown values of the solution of (6) on the grid. To take into account boundary conditions, we will also need $u_{-1}^n$ and $u_N^n$, as well as auxiliary unknowns $\lambda_1^n, \dots, \lambda_{m_l}^n$ and $\rho_1^n, \dots, \rho_{m_r}^n$ corresponding to the functions $\lambda_i(t)$ and $\rho_i(t)$ in (27) and (22).

Recall that values $u_i^0$ are given by the initial condition: $u_i^0 = u_0(x_i)$. For any $n > 0$, we have a usual three-point finite difference scheme:

$$a_l u_{i-1}^{n+1} + a_d u_i^{n+1} + a_u u_{i+1}^{n+1} = b_l u_{i-1}^n + b_d u_i^n + b_u u_{i+1}^n, \quad i = 0, \dots, N-1. \tag{29}$$

In the case of the $\theta$-scheme, the coefficients are given by

$$\begin{aligned}
a_l &= \theta\Delta t \left( -\frac{\sigma^2}{2\Delta x^2} + \frac{\mu}{2\Delta x} \right), \\
a_d &= 1 + \theta\Delta t \frac{\sigma^2}{\Delta x^2}, \\
a_u &= \theta\Delta t \left( -\frac{\sigma^2}{2\Delta x^2} - \frac{\mu}{2\Delta x} \right), \\
b_l &= -(1-\theta)\Delta t \left( -\frac{\sigma^2}{2\Delta x^2} + \frac{\mu}{2\Delta x} \right), \\
b_d &= 1 - (1-\theta)\Delta t \frac{\sigma^2}{\Delta x^2}, \\
b_u &= -(1-\theta)\Delta t \left( -\frac{\sigma^2}{2\Delta x^2} - \frac{\mu}{2\Delta x} \right).
\end{aligned}$$

At the boundary points $x_0 = x_-$ and $x_{N-1} = x_+$ we use boundary conditions (25) and (24) also discretized using standard finite difference schemes. Let us explain this in more details for the left boundary. We discretize (26) as follows

$$\begin{aligned}
\frac{u_1^{n+1} - u_{-1}^{n+1}}{2\Delta x} &= \xi_0 u_0^{n+1} + \alpha_1 \lambda_1^{n+1} + \dots + \alpha_{m_l} \lambda_{m_l}^{n+1} \\
&+ \left( \frac{\psi_1^{n+1} - \psi_{-1}^{n+1}}{2\Delta x} - \xi_0 \psi_0^{n+1} - \alpha_1 \mu_1^{n+1} - \dots - \alpha_{m_l} \mu_{m_l}^{n+1} \right).
\end{aligned} \tag{30}$$

Note that, while it is possible to compute $\partial_x \psi(x_-, t) - \tilde{S}_- \psi(x_-, t)$ analytically, we don't need to do it. We just discretize this expression in the same way as

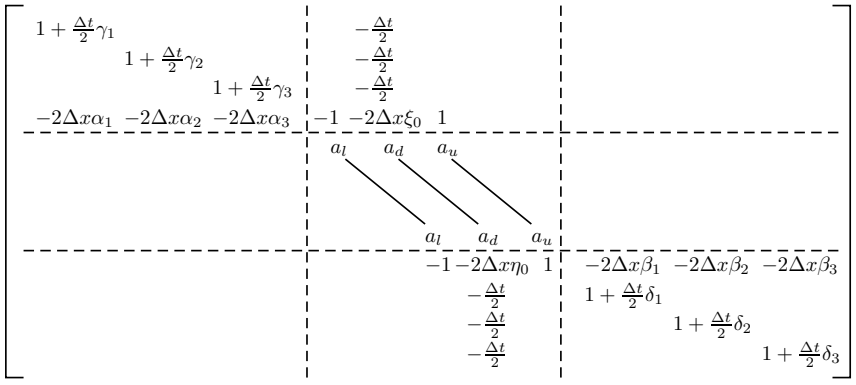


Figure 2: Extended matrix $\tilde{A}$ of the scheme (35).

$\partial_x u(x_-,t) - \tilde{S}_- u(x_-,t)$. Moreover, it is necessary to proceed so if we want to avoid numerical instability.

We also discretize ODEs (27) and (28):

$$\frac{\lambda_j^{n+1}-\lambda_j^n}{\Delta t} + \gamma_j \frac{\lambda_j^{n+1}+\lambda_j^n}{2} = \frac{u_0^{n+1}+u_0^n}{2}, \quad \lambda_j^0 = 0, \quad j = 1,\dots,m_l \tag{31}$$

$$\frac{\mu_j^{n+1}-\mu_j^n}{\Delta t} + \gamma_j \frac{\mu_j^{n+1}+\mu_j^n}{2} = \frac{\psi_0^{n+1}+\psi_0^n}{2}, \quad \mu_j^0 = 0, \quad j = 1,\dots,m_l \tag{32}$$

One way to implement these boundary conditions is to introduce an extended vector of unknowns at $t = t_n$

$$\tilde{u}^n = (\lambda_{m_l}^n,\dots,\lambda_1^n, u_{-1}^n, u_0^n,\dots,u_{N-1}^n, u_N^n, \rho_1^n,\dots,\rho_{m_r}^n)'. \tag{33}$$

It is initialized at

$$\tilde{u}^0 = (0,\dots,0,u_0(x_{-1}),u_0(x_0),\dots,u_0(x_{N-1}),u_0(x_N),0,\dots,0)' \tag{34}$$

and the time iterations take the form

$$\tilde{A}\tilde{u}^{n+1} = \tilde{b}^n \tag{35}$$

where matrix $\tilde{A}$ is mainly tri-diagonal with a few additional rows which are also sparse (see Figure 2 where we take for simplicity $m_l = m_r = 3$). The first $m_l$ rows come from the discretized ODEs on $\lambda_j$ (31), the next row corresponds to the boundary condition (30), the central $N \times N$ tri-diagonal part is the matrix of the $\theta$-scheme (29). The last $1 + m_r$ rows come from a similar treatment of the right boundary condition.

Note that the number of additional rows is small with respect to $N$ and does not depend on $N$. The numerical complexity of the algorithm is the same as for Dirichlet or Neumann boundary conditions. In the environments supporting

sparse matrices, such as Matlab, it is very easy to implement scheme (35) with such an extended matrix. However, if only routines for inverting tri-diagonal matrices are available, it is possible to come back to a tri-diagonal system by eliminating by hands the auxiliary variables $\lambda_{m_l}, \ldots, \lambda_1, u_{-1}, u_N, \rho_1, \ldots, \rho_{m_r}$ from (35). This is the way we implemented the algorithm in Premia. We give below the coefficients of the system after elimination.

A time iteration consists in solving the following linear system

$$Au^{n+1} = b^n \tag{36}$$

where $u^{n+1} = (u_0^{n+1}, \ldots, u_{N-1}^{n+1})'$ is the vector of unknowns, $A$ the following constant tri-diagonal matrix

$$A = \begin{pmatrix} a_d - c_l a_l & a_l + a_u & & & \\ a_l & a_d & a_u & & \\ & \ddots & \ddots & \ddots & \\ & & a_l & a_d & a_u \\ & & & a_l + a_u & a_d + c_r a_u \end{pmatrix} \tag{37}$$

and $b^n = (b_0^n, \ldots, b_{N-1}^n)'$ the vector of the right-hand side. Here

$$c_l = 2\Delta x \left[ \xi_0 + \frac{\Delta t}{2} \sum_{j=1}^{m_l} \frac{\alpha_j}{1 + \frac{\Delta t}{2}\gamma_j} \right],$$

$$c_r = 2\Delta x \left[ \eta_0 + \frac{\Delta t}{2} \sum_{j=1}^{m_r} \frac{\beta_j}{1 + \frac{\Delta t}{2}\delta_j} \right].$$

The computation of the coefficients $b_i^n$ is described in the algorithm below.

**ALGORITHM**

- **Initialization**

$$\begin{aligned} u^0 &= (u_0(x_0, \ldots, u_0(x_{N-1})))' \\ u_{-1}^0 &= u_0(x_{-1}) \\ u_N^0 &= u_0(x_N) \\ \lambda_j^0 &= \mu_j^0 = 0, \quad j = 1, \ldots, m_l \\ \rho_j^0 &= \omega_j^0 = 0, \quad j = 1, \ldots, m_r \end{aligned}$$

- **Iteration** $n \to n+1$**,** $n = 0, \ldots, N_{\text{time}} - 1$
  - *Update coefficients depending on the analytically known functions*

$\psi(x,t)$ *and* $\varphi(x,t)$*:*

$$\mu_j^{n+1} = \frac{1}{1+\frac{\Delta t}{2}\gamma_j}\left[\mu_j^n + \frac{\Delta t}{2}(\psi_0^n - \gamma_j\mu_j^n + \psi_0^{n+1})\right], \quad j = 1,\ldots,m_l$$

$$D\psi^{n+1} = \frac{\psi_1^{n+1} - \psi_{-1}^{n+1}}{2\Delta x} - \xi_0\psi_0^{n+1} - \sum_{j=1}^{m_l}\alpha_j\mu_j^{n+1}$$

$$\omega_j^{n+1} = \frac{1}{1+\frac{\Delta t}{2}\delta_j}\left[\omega_j^n + \frac{\Delta t}{2}(\varphi_{N-1}^n - \delta_j\omega_j^n + \varphi_{N-1}^{n+1})\right], \quad j = 1,\ldots,m_r$$

$$D\varphi^{n+1} = \frac{\varphi_N^{n+1} - \varphi_{N-2}^{n+1}}{2\Delta x} - \eta_0\varphi_{N-1}^{n+1} - \sum_{j=1}^{m_r}\beta_j\omega_j^{n+1}$$

– *Compute the right-hand side:*

$$b_0^n = b_l u_{-1}^n + b_d u_0^n + b_u u_1^n$$
$$+2\Delta x a_l\left[\sum_{j=1}^{m_l}\frac{\alpha_j}{1+\frac{\Delta t}{2}\gamma_j}(\lambda_j^n + \frac{\Delta t}{2}(u_0^n - \gamma_j\lambda_j^n)) + D\psi^{n+1}\right]$$

$$b_i^n = b_l u_{i-1}^n + b_d u_i^n + b_u u_{i+1}^n, \quad i = 1,\ldots,N-2$$

$$b_{N-1}^n = b_l u_{N-2}^n + b_d u_{N-1}^n + b_u u_N^n$$
$$-2\Delta x a_u\left[\sum_{j=1}^{m_r}\frac{\beta_j}{1+\frac{\Delta t}{2}\delta_j}(\rho_j^n + \frac{\Delta t}{2}(u_{N-1}^n - \delta_j\rho_j^n)) + D\varphi^{n+1}\right]$$

– *Solve* $Au^{n+1} = b^n$*.*

– *Update auxiliary unknowns:*

$$\lambda_j^{n+1} = \frac{1}{1+\frac{\Delta t}{2}\gamma_j}\left[\lambda_j^n + \frac{\Delta t}{2}(u_0^n - \gamma_j\lambda_j^n + u_0^{n+1})\right], \quad j = 1,\ldots,m_l$$

$$u_{-1}^{n+1} = u_1^{n+1} - 2\Delta x\left[\xi_0 u_0^{n+1} + \sum_{j=1}^{m_l}\alpha_j\lambda_j^{n+1} + D\psi^{n+1}\right]$$

$$\rho_j^{n+1} = \frac{1}{1+\frac{\Delta t}{2}\delta_j}\left[\rho_j^n + \frac{\Delta t}{2}(u_{N-1}^n - \delta_j\rho_j^n + u_{N-1}^{n+1})\right], \quad j = 1,\ldots,m_r$$

$$u_N^{n+1} = u_{N-2}^{n+1} + 2\Delta x\left[\eta_0 u_{N-1}^{n+1} + \sum_{j=1}^{m_r}\beta_j\rho_j^{n+1} + D\varphi^{n+1}\right]$$

To treat European and American options in a unified manner, we solve $Au^{n+1} = b^n$ by the Brennan-Schwartz algorithm where, of course, we don't take the maximum between the intermediary solution and the payoff in the European case. For European options, the Brennan-Schwartz algorithm is just a variant of the LU-solver for linear systems of algebraic equations.

## References