Inriamericwaves
Report of activities - March-May/16

Joao Guilherme Caldas Steinstraesser
José Galaz

June 10, 2016

Contents

1 Study of Transparent Boundary Conditions in splitting methods

As an introduction for the future application of the Transparent Boundary Conditions in splitting methods, for the resolution of the wave propagation models studied in this project, we will present in this section a simple one-dimensional case, for which we will implement and qualitatively validate our proposed method. The idea is to verify if imposing independent boundary conditions for each step of the splitting method provides a good approximation for the TBCs.

After this initial propose, we will work in the numerical experiment presented in [2]. In this paper, approximate TBCs are implemented for a linear advection-diffusion equation with one time-dependent boundary condition, solved with a full method. Therefore, we will implement the tests described in the paper, in order to obtain the same results and compare with the solutions of the methods that we will propose.

1.1 Linear advection-diffusion equation

We seek to solve here the problem

$$\begin{cases} u_t + au_x - \varepsilon u_{xx} = 0, \quad x \in \Omega = [0,1], \quad t \geq 0, \quad a \in \mathbb{R}, \quad \varepsilon > 0 \\ u(0,x) = u_0(x) = e^{-\frac{(x-0.5)^2}{0.01}} \\ + \text{ boundary conditions} \end{cases} \tag{1}$$

In order to focus our analysis in only one of the two boundaries, we will consider $a = 1$ (so the wave travels to the right), and choose $\varepsilon =$ such that the arrival of the wave to the right boundary occurs before the arrival of the diffused solution to the left one. We will compare the results obtained with different methods :

Splitting method : The splitting method proposed here leads, in each time step $[t_n, t_{n+1}]$, to the resolution of the linear advection equation and the heat equation :

$$\begin{cases} L_a(v) = v_t + av_x = 0, \quad v^n = u^n, \quad t \in [t_n, t_{n+1}] \\ L_d(w) = w_t - \varepsilon w_{xx} = 0, \quad w^n = v^{n+1}, \quad t \in [t_n, t_{n+1}] \\ u^{n+1} = w^{n+1} \end{cases} \tag{2}$$

Both equations will be solved with Finite Difference methods. In the case of the linear advection, it will be an explicit backward method, with an homogeneous Dirichlet condition on the left boundary :

$$\begin{cases} u_i^{n+1} = u_i^n - a\Delta t \frac{u_i^n - u_{i-1}^n}{\Delta x}, \quad i = 1, ..., N \\ u_0^{n+1} = 0 \end{cases} \tag{3}$$

For the heat equation, we will use the Crank-Nicolson method (semi-implicit), with homogeneous Neumann conditions in both boundaries :

$$\begin{cases} \frac{u_i^{n+1} - u_i^n}{\Delta t} - \varepsilon \frac{1}{2} \left(\frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{\Delta x^2} + \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} \right) = 0 \\ u_1^{n+1} = u_0^{n+1} \\ u_N^{n+1} = u_{N-1}^{n-1} \end{cases} \tag{4}$$

Full equation In order to compare with the results of the splitting method, we implemented a resolution of the full linear advection-diffusion equation, also with the Crank-Nicolson method, with an explicit discretization of the advection term :

$$\begin{cases} \frac{u_i^{n+1} - u_i^n}{\Delta t} + a \frac{u_i^n - u_{i-1}^n}{\Delta x} - \varepsilon \frac{1}{2} \left(\frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{\Delta x^2} + \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} \right) = 0 \\ u_0^{n+1} = 0 \\ B_j(u_n^{n+1}) = 0 \end{cases} \tag{5}$$

The boundary conditions B_j are the approximate boundary conditions proposed in [2], where j indicates the order of a Taylor expansion of the solutions of the characteristic equation obtained when solving 1 in the Fourier space. We implemented here the zero and first order approximations:

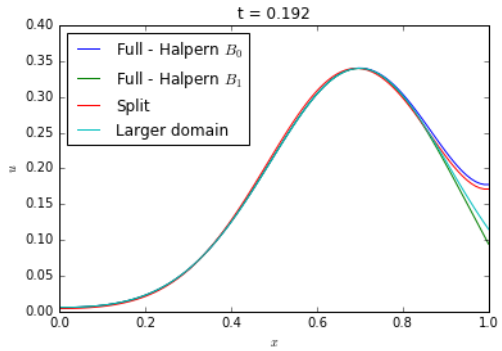
$$B_0 = u_x = 0 \implies u_N^{n+1} - u_{N-1}^{n+1} = 0 \tag{6}$$

$$B_1 = u_t + u_x = 0 \implies \frac{u_N^{n+1} - u_N^n}{\Delta t} - \frac{u_N^{n+1} - u_{N-1}^{n+1}}{\Delta x} = 0 \tag{7}$$

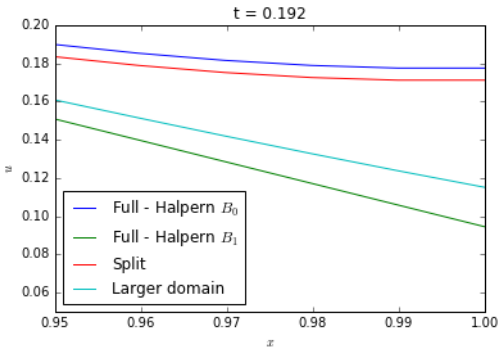
We notice that B_0 corresponds to an homogeneous Neumann boundary condition.

We show in the figure 1 some snapshots of the computed solutions. We compare them with a referential solution, which was computed with the full equation, using the TBC B_1 and in a larger domain ($\Omega_{ref} = [0, 2]$). The choice of the boundary condition may not have an influence over the comparison that we intend to do here, because we will study the solution near

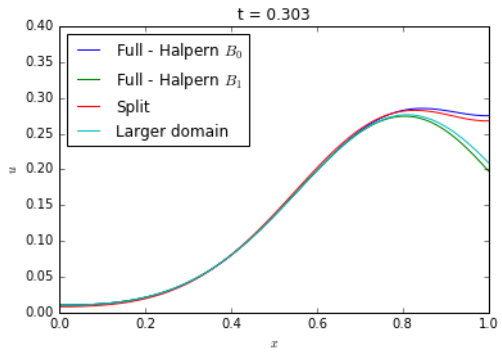
the boundary $x = 1$, which is far enough of the boundary in the referential domain.



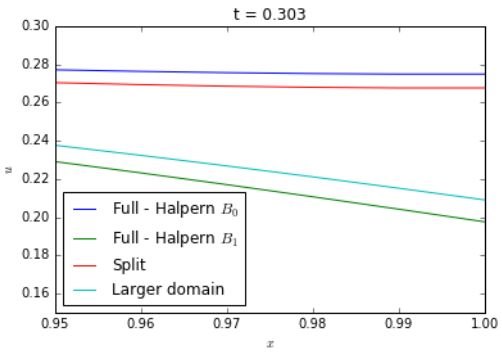
(a) Solution in the whole domain



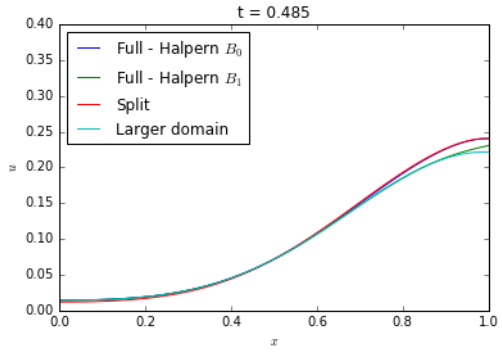
(b) Detail - right boundary



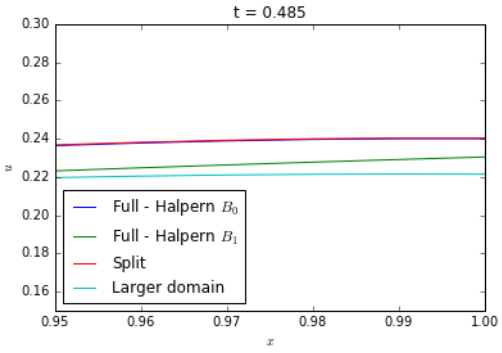
(c) Solution in the whole domain



(d) Detail - right boundary



(e) Solution in the whole domain



(f) Detail - right boundary

Figure 1: Results in different instants for the linear advection-diffusion equation, computed with a splitting method (in red) or full methods (in dark blue and green, with different approximate TBCs); also compared with a referential solution.

The results of the figure 1 shows that, compared with the referential solution, the condition B_1 provides a better approximation for the TBC.

The full equation with B_0 and the splitting method provides similar but worse result, even though they qualitatively allows the solution to exit the domain without notable influence of the boundary.

1.2 Numerical experiment realized in [2]

In order to make a deeper study of the boundary conditions in splitting methods, we will implement the numerical experiment presented in [2] and compare it with the solution of the splitted equation.

This numerical experiment consists in the same linear advection-diffusion equation, but with zero initial conditions and a time-dependent boundary condition imposed on the left boundary :

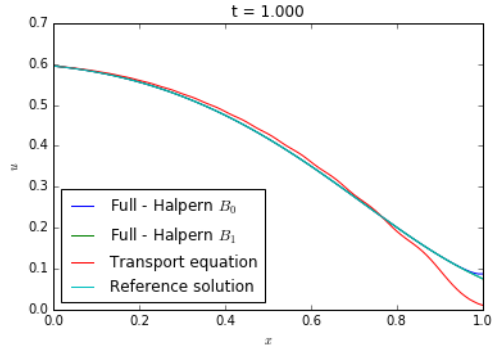
$$\begin{cases} u_t + u_x - u_{xx} = 0, \quad x \in \Omega = [0,1], \quad t \geq 0 \\ u(0,x) = 0, \quad x \in \Omega \\ u(t,0) = \frac{sint}{\sqrt{t^2+1}} \\ + \text{ right boundary conditions} \end{cases} \tag{8}$$

Similarly to what we did in the previous section, [2] considers a referential solution computed in the domain $\Omega_{ref} = [0,2]$. In order to validate our implementation, we will repeat some tests presented in the paper:

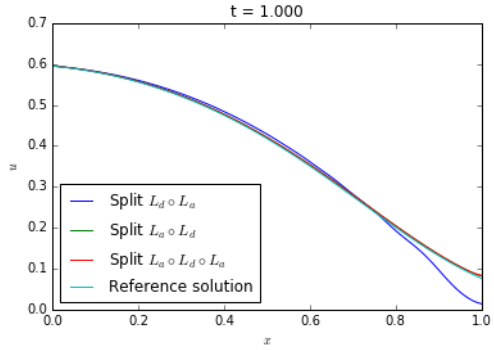
- Full equation with right boundary condition B_0;
- Full equation with right boundary condition B_1;
- Transport equation (i.e, ignoring the diffusive term).

We will also solve the equation using several splitting methods. Recalling the operators defined in (2), and introducing a superscript denoting the time interval in which the operator is solved, we will solve, for each time step

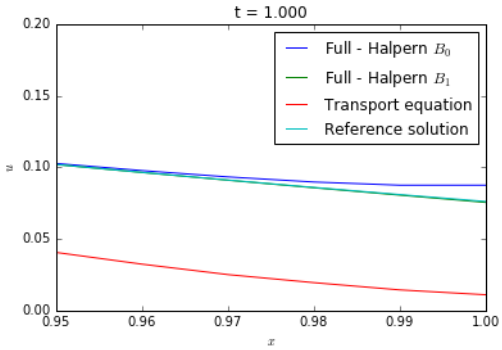
- $L_d^{\Delta t}(L_a^{\Delta t}(u)) = 0$;
- $L_a^{\Delta t}(L_d^{\Delta t}(u)) = 0$;
- $L_a^{\Delta t/2}(L_d^{\Delta t}(L_a^{\Delta t/2}(u))) = 0$;



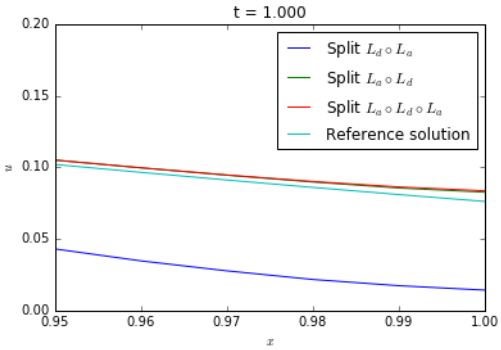
(a) Solution in the whole domain



(b) Solution in the whole domain



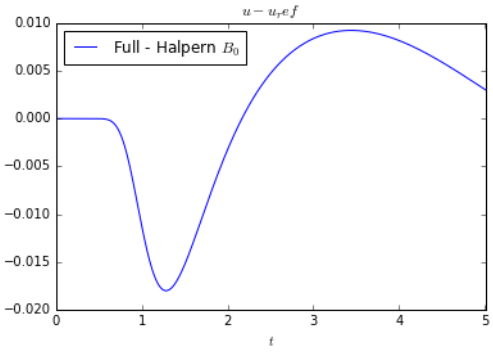
(c) Detail - right boundary



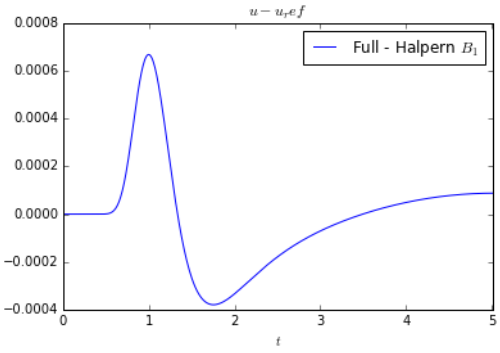
(d) Detail - right boundary

Figure 2: Results in $t = 1$ for the numerical experiment in [2], computed with several methods and boundary conditions.

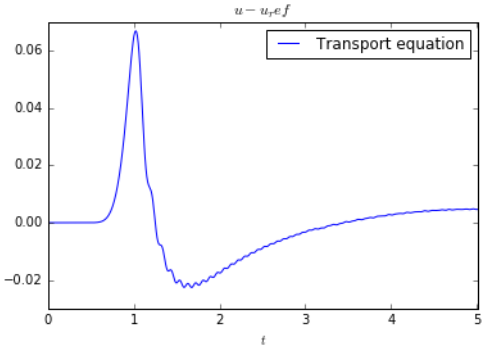
Also as done in [2], we compute for each time step the error on the boundary, $e^n = u_N^n - (u_{ref})_N^n$, as shown in the figure 3



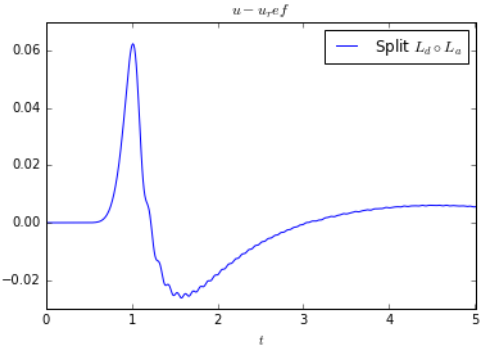
(a) Full equation with boundary condition B_0



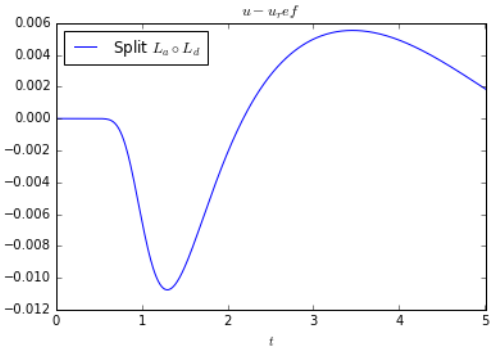
(b) Full equation with boundary condition B_1



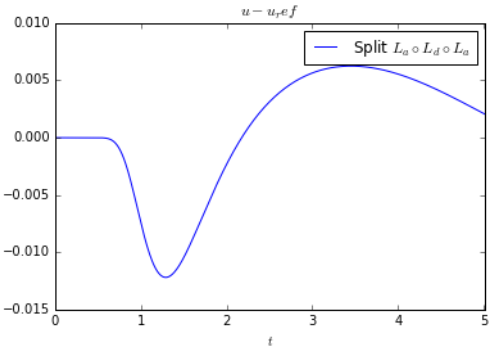
(c) Transport equation



(d) Splitting $L_d \circ L_a$



(e) Splitting $L_a \circ L_d$



(f) Splitting $L_a \circ L_d \circ L_a$

Figure 3: Evolution of the error $e^n = u_N^n - (u_{ref})_N^n$ for each method implemented

2 Derivation and approximation of Transparent Boundary Conditions for the linearized KdV equation

Two main objectives motivate the work developed in this section : firstly, we want to present the usual derivation of Transparent Boundary Conditions in the case of linearized problems; and secondly, after obtaining analytical expressions for the TBCs (which in general have a too much expensive application), we will propose, optimize and test approximations for them.

2.1 Description of the TBCs

The derivation of the TBCs is based on [1] . We will consider the formulation for the continuous version of the homogeneous and linearized KdV equation :

$$u_t + U_1 u_x + U_2 u_{xxx} = 0, \quad U_1 \in \mathbb{R}, \quad U_2 > 0 \tag{9}$$

Denoting by $\mathcal{L}^{-1}$ the inverse Laplace transform, the TBCs are

$$\begin{cases} u(t,a) - U_2 \mathcal{L}^{-1}\left(\frac{\lambda_1(s)^2}{s}\right) * u_x(t,a) - U_2 \mathcal{L}^{-1}\left(\frac{\lambda_1(s)}{s}\right) * u_{xx}(t,a) = 0 \\ u(t,b) - \mathcal{L}^{-1}\left(\frac{1}{\lambda_1(s)^2}\right) * u_{xx}(t,b) = 0 \\ u_x(t,b) - \mathcal{L}^{-1}\left(\frac{1}{\lambda_1(s)}\right) * u_{xx}(t,b) = 0 \end{cases} \tag{10}$$

where $[a,b]$ is the computational physical domain, $s \in \mathbb{C}$ is the Laplace frequency and λ_1 is one of the roots of the cubic characteristic equation obtained when solving (9) in the Laplace space.

We firstly notice that we can rewrite (10) as

$$\begin{cases} u(t,a) - U_2 \mathcal{L}^{-1}\left(\frac{\lambda_1(s)^2}{s}\hat{u}_x(t,s)\right) - U_2 \mathcal{L}^{-1}\left(\frac{\lambda_1(s)}{s}\hat{u}_{xx}(t,s)\right) = 0 \\ u(t,b) - \mathcal{L}^{-1}\left(\frac{1}{\lambda_1(s)^2}\hat{u}_{xx}(t,s)\right) = 0 \\ u_x(t,b) - \mathcal{L}^{-1}\left(\frac{1}{\lambda_1(s)}\hat{u}_{xx}(t,s)\right) = 0 \end{cases} \tag{11}$$

where $\hat{u}$ is the Laplace transform of u.

In the case of pure dispersion ($U_1 = 0$ and $U_2 = 1$), λ is written as

$$\lambda(s) = -\sqrt[3]{s}$$

So, from (2.1) we can write

$$\frac{\lambda}{s} = \frac{\lambda}{-\lambda^3} = -\frac{1}{\lambda^2}\frac{\lambda^2}{s} = \frac{\lambda^2}{-\lambda^3} = -\frac{1}{\lambda} \tag{12}$$

In the sequence, we will always consider this case ($U_1 = 0$ and $U_2 = 1$).

2.2 Approximation of the TBCs

2.2.1 Approximation using a constant polynomial

In a first moment, we will approximate $\frac{\lambda^2}{s}$ by a constant polynomial $P_0(s) = c$. Therefore, from (12) we get

$$\frac{\lambda^2}{s} = c\frac{\lambda}{s} = -c^2\frac{1}{\lambda_1(s)^2} = c^2\frac{1}{\lambda_1(s)} = -c \tag{13}$$

and replacing in (11) and considering the linearity of the Laplace transform :

$$\begin{cases} u(t,a) - cU_2u_x(t,a) + c^2U_2u_{xx}(t,s) = 0 \\ u(t,b) - c^2\hat{u}_{xx}(t,s) = 0 \\ u_x(t,b) + cu_{xx}(t,s) = 0 \end{cases} \tag{14}$$

Using finite difference approximations and considering different constants c_Landc_R for the left and the right boundaries, (14) is discretized as

$$\begin{cases} u_0 - c_LU_2\frac{u_1-u_0}{\Delta x} + c_L^2U_2\frac{u_0-2u_1+u_2}{\Delta x^2} = 0 \\ u_N - c_R^2\frac{u_N-2u_{N-1}+u_{N-2}}{\Delta x^2} = 0 \\ \frac{u_N-u_{N-1}}{\Delta x} + c_R^2\frac{u_N-2u_{N-1}+u_{N-2}}{\Delta x^2} = 0 \end{cases} \tag{15}$$

2.2.2 Approximation using a linear polynomial

Similarly to the case presented above, we will approximate $\frac{\lambda^2}{s}$ by a linear polynomial $P_1(s) = ds + c$. Therefore,

$$\frac{\lambda^2}{s} = ds + c\frac{\lambda}{s} = -(ds+c)^2\frac{1}{\lambda_1(s)^2} = (ds+c)^2\frac{1}{\lambda_1(s)} = -(ds+c) \tag{16}$$

Thus, the inverse Laplace transforms in (11) reads :

$$
\begin{aligned}
\mathcal{L}^{-1}\left(\frac{\lambda^2}{s}\hat{u}_x(t,s)\right) =& \mathcal{L}^{-1}\left[(ds+c)\hat{u}_x(t,s)\right] = \\
& d\mathcal{L}^{-1}\left[\hat{u}_{xt}(t,s)\right] + c\mathcal{L}^{-1}\left[\hat{u}_x(t,s)\right] = du_{xt}(x,t) + cu_x(x,t) \\
\mathcal{L}^{-1}\left(\frac{\lambda}{s}\hat{u}_{xx}(t,s)\right) =& \mathcal{L}^{-1}\left[-(ds+c)^2\hat{u}_{xx}(t,s)\right] = -d^2u_{xxtt}(x,t) - 2dcu_{xxt}(x,t) - c^2u_{xx}(x,t) \\
\mathcal{L}^{-1}\left(\frac{1}{\lambda^2}\hat{u}_{xx}(t,s)\right) =& \mathcal{L}^{-1}\left[(ds+c)^2\hat{u}_{xx}(t,s)\right] = d^2u_{xxtt}(x,t) + 2dcu_{xxt}(x,t) + c^2u_{xx}(x,t) \\
\mathcal{L}^{-1}\left(\frac{1}{\lambda}\hat{u}_{xx}(t,s)\right) =& \mathcal{L}^{-1}\left[-(ds+c)\hat{u}_{xx}(t,s)\right] = -du_{xxt}(x,t) - cu_{xx}(x,t)
\end{aligned}
\tag{17}
$$

Using finite differences and different coefficients c_L, c_R and d_L, d_R for the left and the right boundaries, these expressions can be approximated by

$$
\begin{aligned}
\mathcal{L}^{-1}\left(\frac{\lambda^2}{s}\hat{u}_x(t,s)\right) =& d_L\frac{(u_x)_0^{n+1} - (u_x)_0^n}{\Delta t} + c_L(u_x)_0^{n+1} = \\
& \left(\frac{d_L}{\Delta t} + c_L\right)\left(\frac{u_1^{n+1} - u_0^{n+1}}{\Delta x}\right) - \frac{d_L}{\Delta t}\left(\frac{u_1^n - u_0^n}{\Delta x}\right)
\end{aligned}
\tag{18}
$$

$$
\begin{aligned}
\mathcal{L}^{-1}\left(\frac{\lambda}{s}\hat{u}_{xx}(t,s)\right) =& -d_L^2\left(\frac{(u_{xx})_0^{n+1} - 2(u_{xx})_0^n + (u_{xx})_0^{n-1}}{\Delta t^2}\right) + \\
& - 2d_Lc_L\left(\frac{(u_{xx})_0^{n+1} - (u_{xx})_0^n}{\Delta t}\right) - c_L^2(u_{xx})_0^{n+1} = \\
& -\left(\frac{d_L^2}{\Delta t^2} + \frac{2d_Lc_L}{\Delta t} + c_L^2\right)\left(\frac{u_0^{n+1} - 2u_1^{n+1} + u_2^{n+1}}{\Delta x^2}\right) + \\
& \left(2\frac{d_L^2}{\Delta t^2} + \frac{2d_Lc_L}{\Delta t}\right)\left(\frac{u_0^n - 2u_1^n + u_2^n}{\Delta x^2}\right) - \frac{d_L^2}{\Delta t^2}\left(\frac{u_0^{n-1} - 2u_1^{n-1} + u_2^{n-1}}{\Delta x^2}\right)
\end{aligned}
\tag{19}
$$

$$
\begin{aligned}
\mathcal{L}^{-1}\left(\frac{1}{\lambda^2}\hat{u}_{xx}(t,s)\right) =& \left(\frac{d_R^2}{\Delta t^2} + \frac{2d_Rc_R}{\Delta t} + c_R^2\right)\left(\frac{u_N^{n+1} - 2u_{N-1}^{n+1} + u_{N-2}^{n+1}}{\Delta x^2}\right) + \\
& - \left(2\frac{d_R^2}{\Delta t^2} + \frac{2d_Rc_R}{\Delta t}\right)\left(\frac{u_N^n - 2u_{N-1}^n + u_{N-2}^n}{\Delta x^2}\right) + \\
& \frac{d_R^2}{\Delta t^2}\left(\frac{u_N^{n-1} - 2u_{N-1}^{n-1} + u_{N-2}^{n-1}}{\Delta x^2}\right)
\end{aligned}
\tag{20}
$$

$$\mathcal{L}^{-1}\left(\frac{1}{\lambda}\hat{u}_{xx}(t,s)\right) = -d_R\frac{(u_{xx})_0^{n+1}-(u_{xx})_0^n}{\Delta t} - c_R(u_{xx})_0^{n+1} = \\ -\left(\frac{d_R}{\Delta t}+c_R\right)\left(\frac{u_N^{n+1}-2u_{N-1}^{n+1}+u_{N-2}^{n+1}}{\Delta x^2}\right)+\frac{d_R}{\Delta t}\left(\frac{u_N^n-2u_{N-1}^n+u_{N-2}^n}{\Delta x^2}\right) \tag{21}$$

Then we use (18) - (21) in (11) to obtain the discrete TBCs :

$$\begin{aligned} u_0^{n+1} - U_2\left(\frac{d_L}{\Delta t}+c_L\right)\left(\frac{u_1^{n+1}-u_0^{n+1}}{\Delta x}\right)+ \\ U_2\left(\frac{d_L^2}{\Delta t^2}+\frac{2d_Lc_L}{\Delta t}+c_L^2\right)\left(\frac{u_0^{n+1}-2u_1^{n+1}+u_2^{n+1}}{\Delta x^2}\right) = \\ -U_2\frac{d_L}{\Delta t}\left(\frac{u_1^n-u_0^n}{\Delta x}\right)+U_2\left(2\frac{d_L^2}{\Delta t^2}+\frac{2d_Lc_L}{\Delta t}\right)\left(\frac{u_0^n-2u_1^n+u_2^n}{\Delta x^2}\right)+ \\ -U_2\frac{d_L^2}{\Delta t^2}\left(\frac{u_0^{n-1}-2u_1^{n-1}+u_2^{n-1}}{\Delta x^2}\right) \end{aligned} \tag{22}$$

$$u_N^{n+1} - \left(\frac{d_R^2}{\Delta t^2}+\frac{2d_Rc_R}{\Delta t}+c_R^2\right)\left(\frac{u_N^{n+1}-2u_{N-1}^{n+1}+u_{N-2}^{n+1}}{\Delta x^2}\right) = \\ -\left(2\frac{d_R^2}{\Delta t^2}+\frac{2d_Rc_R}{\Delta t}\right)\left(\frac{u_N^n-2u_{N-1}^n+u_{N-2}^n}{\Delta x^2}\right)+\frac{d_R^2}{\Delta t^2}\left(\frac{u_N^{n-1}-2u_{N-1}^{n-1}+u_{N-2}^{n-1}}{\Delta x^2}\right) \tag{23}$$

$$\begin{aligned} \frac{u_N^{n+1}-u_{N-1}^{n+1}}{\Delta x}+\left(\frac{d_R}{\Delta t}+c_R\right)\left(\frac{u_N^{n+1}-2u_{N-1}^{n+1}+u_{N-2}^{n+1}}{\Delta x^2}\right) = \\ \frac{d_R}{\Delta t}\left(\frac{u_N^n-2u_{N-1}^n+u_{N-2}^n}{\Delta x^2}\right) \end{aligned} \tag{24}$$

2.3 Numerical tests

In order to validate compare our approximation with the results obtained by Besse, we will solve the same numerical test presented in his paper :

$$u_t + u_{xxx} = 0, \quad x \in \mathbb{R} \tag{25}$$

$$u(0,x) = e^{-x^2}, \quad x \in \mathbb{R} \tag{26}$$

$$u \to 0, \quad |x| \to \infty \tag{27}$$

The fundamental solution of (25) is

$$E(t,x) = \frac{1}{\sqrt[3]{3t}} Ai\left(\frac{x}{\sqrt[3]{3t}}\right) \tag{28}$$

where Ai is the Airy function, and the exact solution for the problem (25) - (27) is

$$u_{exact}(t,x) = E(t,x) * e^{-x^2} \tag{29}$$

The problem will be solved in the spatial domain $[-6,-6]$

For a quantitative evaluation of the results, we computed the same errors defined in the paper of Besse et al. For each time step, we compute the relative error

$$e^n = \frac{\left\| u^n_{exact} - u^n_{computed} \right\|_2}{\left\| u^n_{exact} \right\|_2}$$

and, in the whole time interval :

$$e_{Tm} = \max_{0<n<T_{max}} (e^n)$$

$$e_{L2} = \sqrt{\Delta t \sum_{n=1}^{T_{max}} (e^n)^2}$$

We also generate plottings and animations for the best and the worst solutions. In order to make a better comparison, in the definition of "worst solution" we ignored the ones for which the numerical computations diverged (following the arbitrary criteria $e_{L2} > 10$.

Several tests will be made with different combinations of the coefficients in the polynomial approximation of $\frac{\lambda^2}{s}$. For the constant polynomial $P_0(s) = c$, we will optimize the parameters (c_L, c_R). For the linear polynomial $P_1(s) = cs + d$, we will optimize the four parameters (c_L, d_L, c_R, d_R), but in a first moment we will consider $c_L = c_R$ and $d_L = d_R$ and study the behavior of the approximation in only one of the boundaries.

2.4 Introduction tests

In a first moment, we will execute the tests with the proposed TBCs for a small range of parameters (c_L, c_R) (for the constant polynomial approximation) and (c_L, c_R, d_L, d_R) (for the linear polynomial approximation). We tested all the possible combinations between the values in $[-10, -1, -0.1, 0, 0.1, 1, 10]$. In the case of the linear polynomial approximation, in order to avoid large computations, we considered only the cases where $c_L = c_R$ and $d_L = d_R$.

The objective of these initial tests is to see the general behavior of the proposed approximations and guide our next steps in this study. The tables 1 and 2 show the best results (concerning the error e_{L2}) for each one of the approximations; and the figures 4 and 5 show some snapshots comparing the best, the worst and the analytical solution. The choice of the worst solution does not consider the ones that "exploded".

c_L	c_R	e_{L2}
1.0	1.0	0.0947
1.0	10.0	0.0973
1.0	0.1	0.0984
1.0	0.0	0.0992
1.0	-10.0	0.0994
1.0	-0.1	0.1000
1.0	-1.	0.1016
10.0	1.0	0.3470
10.0	0.1	0.3474
10.0	0.0	0.3475

Table 1: Best results (smallest e_{L2}) for the constant polynomial approximation

$d_L = d_R$	$c_L = c_R$	e_{L2}
0.	1.0	0.0947
0.1	1.0	0.1234
1.0	1.0	0.2003
10.0	0.1	0.2204
-10.0	0.1	0.2398
10.0	1.0	0.2716
-10.0	0.0	0.2480
-10.0	1.0	0.3004
10.0	0.0	0.2721
0.0	0.1	0.3674

Table 2: Best results (smallest e_{L2}) for the linear polynomial approximation

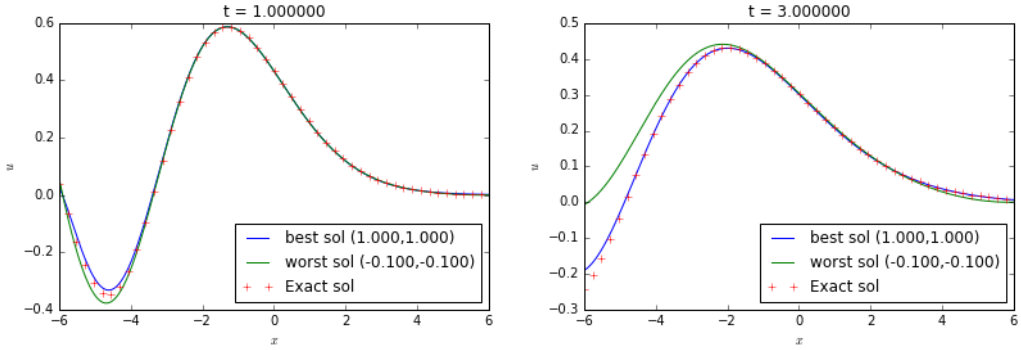


Figure 4: Snapshots with the best and the worst solution in the case of the constant polynomial approximation

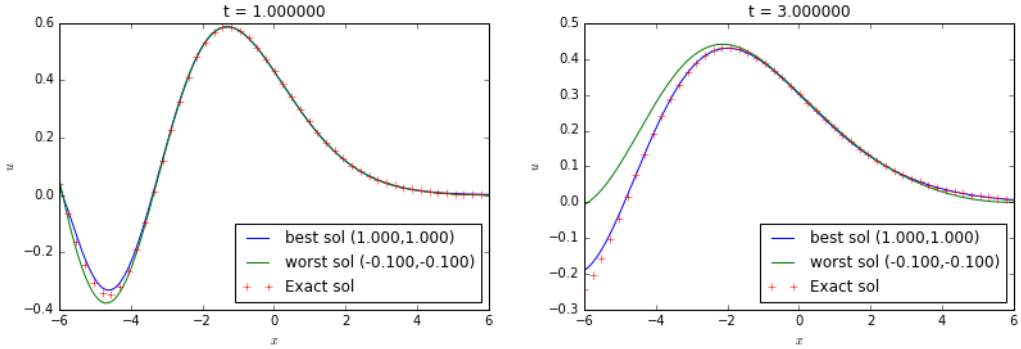


Figure 5: Snapshots with the best and the worst solution in the case of the linear polynomial approximation

The results of the tables 1 and 2 and the figures 4 and 5 shows that using a higher-order polynomial approximation in the TBCs (which implies the existence of time derivative terms) does not improve the error of the numerical solution. In fact, the best solution was the one with $d_L = d_R = 0$, which leads to constant polynomial case. Additionally, The tests with P_0 shows that the solution is much more sensitive to the left than the right boundary, which is a consequence of the fact that the solution is practically constant and equal to zero in $L = 6$, but shows great variations in $L = -6$.

Therefore, in the following we will work only with the constant polynomial approximation, due to the better results that it provides and its simpler implementation.

3 Application of the approximate TBCs to a Domain Decomposition Method

The constant polynomial approximation of the Transparent Boundary Conditions, expressed by 14, will be applied to the implementation of a Domain Decomposition Method (DDM). Firstly, we will briefly describe the DDM that we will consider here, and after we will describe and test the incorporation of the proposed TBCs.

3.1 The Schwarz Methods

The following description is based on [**?**]. Domain Decomposition Methods allow to decompose a domain Ω in multiple subdomains Ω_i (that can possibly overlap) and solve the problem in each one of them. Therefore, one must find functions that satisfies the PDE in each subdomain and that match on the interfaces.

The first DDM developed was the Schwarz method, which consists on an iterative method : in the case of a evolution problem, the solution $u_i^{n,\infty}$, in each time step t_n and each subdomain Ω_i, is computed as the convergence of the solution obtained in each iteration, $u_i^{n,k}$, $\quad k \geq 0$. There are two types of Schwarz methods, depending on the way that the boundary conditions on the interfaces are constructed for computing $u_i^{n,k}$

In the additive Schwarz method (ASM), the boundary conditions are always constructed using the solution $u_j^{n,k-1}$, $\quad j \neq i$ of the previous iteration in the other partitions of the domain. Therefore, in each interface between the domains Ω_i and Ω_j, the boundary condition for the problem in Ω_i is

$$\mathcal{B}(u_i^{n,k+1}) = \mathcal{B}(u_j^{n,k})$$

where $\mathcal{B}$ denotes the operator of the TBC.

In the order hand, the multiplicative Schwarz method (MSM) uses always the most recent information for computing the interface boundary conditions. Therefore, if we consider a DDM with two subomains, Ω_1 and Ω_2, they TBCs would be written (for example) as

$$\mathcal{B}(u_1^{n,k+1}) = \mathcal{B}(u_2^{n,k})\mathcal{B}(u_2^{n,k+1}) = \mathcal{B}(u_1^{n,k+1})$$

for solving the problem in Ω_1 and Ω_2, respectively.

We will consider here only the ASM.In the following description, without lost of generality, we will consider a domain decomposed in two non-overlapping subdomains.

Evidently, the biggest challenge of the Schwarz methods is to define appropriate operators such that :

- The method shows a fast convergence
- The solution u_i in each subdomain Ω_i converges to $u|_{\Omega_1}$, i.e, the solution u of the monodomain Ω restricted to Ω_1

In fact, accordingly to [**?**], the optimal additive Schwarz method is the one which uses the TBCs **??** as interface boundary conditions: with them, the method converges in two iterations, and no ASM can converge in less than two iterations.

Nevertheless, as discussed previously in this report, the numerical implementation of the exact TBCs **??** are generally impractical, because they are non local in time. Therefore, one should use approximate TBCs, what will be done here in the sequence using the approximations proposed n the previous section.

3.2 ASM with the approximate TBCs for the dispersive equation

The resolution of the dispersive equation with the Additive Schwarz method, using the constant polynomial approximation for the TBCs, is written as

$$\begin{cases} (u_1^{n,k+1})_t + (u_1^{n,k+1})_{xxx} = 0, \quad x \in \Omega_1, \quad t \leq 0 \\ u_1^{n,0} = u_1^{n-1,\infty}, \quad x \in \Omega_1 \\ \Upsilon_1^{c_L^*}(u_1^{n+1,k+1}, -L) = 0, \\ \Theta_2^{c_R^*}(u_1^{n+1,k+1}, 0) = \Theta_2^{c_R^*}(u_2^{n,k}, 0), \\ \Theta_3^{c_R^*}(u_1^{n+1,k+1}, 0) = \Theta_3^{c_R^*}(u_2^{n,k}, 0) \end{cases} \tag{30}$$

$$\begin{cases} (u_2^{n,k+1})_t + (u_2^{n,k+1})_{xxx} = 0, \quad x \in \Omega_2, \quad t \leq 0 \\ u_2^{n,0} = u_2^{n-1,\infty}, \quad x \in \Omega_2 \\ \Theta_1^{c_L^*}(u_2^{n+1,k+1}, 0) = \Theta_1^{c_L^*}(u_1^{n,k}, 0) \\ \Upsilon_2^{c_R^*}(u_2^{n+1,k+1}, L) = 0 \\ \Upsilon_3^{c_R^*}(u_2^{n+1,k+1}, L) = 0 \end{cases} \tag{31}$$

where Υ_i, $i = 1, 2, 3$, are the boundary conditions on the external boundaries (i.e, in the intersection of the monodomain boundaries and the subdomain boundaries). These external BCs are independent of the interface BCs. Here, we will consider $\Upsilon_1 = \Theta_1^{1.0}$, $\Upsilon_2 = \Theta_2^{0.0}$ and $\Upsilon_3 = \Theta_3^{0.0}$, which gives

$$\Upsilon_1(u,x) = u - u_x + u_{xx} \tag{32}$$
$$\Upsilon_2(u,x) = 0 \tag{33}$$
$$\Upsilon_3(u,x) = 0 \tag{34}$$
$$\tag{35}$$

This choice was made based on the easy implementation and the good results provided by the coefficients $c_L = 1.0$ and $c_R = 0.0$ in approximating the analytical solution in Ω (as shown in the table 1). Nevertheless,it does not have much importance in the study that we will done in the following paragraphs. In fact, our purpose is to study exclusively the behavior of the DDM implemented here; therefore, all the results must be compared to a referential solution u_{ref}, that can be simply the numerical solution of the monodomain problem. The only restriction for an appropriate study is that the external BCs for computing u_{ref} must be same as $\Upsilon_i, \quad i = 1, 2, 3.$

3.3 Error in the converged solution

When using approximate TBCs in the ASM, when should guarantee that the converged solutions u_1, u_2 satisfies the same equation as the solution u_{ref} of the monodomain problem. Firstly, we will show a very simple example in which this property is not verified, and after we will prove and verify numerically that the the method (30) - (31) proposed here has the same problem. Based on that, we will be able to propose corrections for it.

3.3.1 Motivational example :

Consider the 1D problem

$$\Big\{ -\Delta u + \lambda u = f, \quad x \in \Omega = [-L, L], \quad t \geq 0 u(-L) = u(L) = 0 \tag{36}$$

solved with the ASM, using a Neumann boundary condition on the interface :

$$\begin{cases} -\Delta(u_1^{k+1}) + \lambda(u_1^{k+1}) = f, \quad x \in \Omega_1 = [-L, 0], \quad t \geq 0 \\ (u_1^{k+1})_x(0) = (u_2^k)_x(0) \end{cases} \tag{37}$$

$$\begin{cases} -\Delta(u_2^{k+1}) + \lambda(u_2^{k+1}) = f, \quad x \in \Omega_2 = [0, L], \quad t \geq 0 \\ (u_2^{k+1})_x(0) = (u_1^k)_x(0) \end{cases} \tag{38}$$

The problem can be solved with the following Finite Difference Discretization

$$-\frac{u_{i,j-1} - 2_{i,j} + u_{i,j+1}}{\Delta x^2} + \lambda u_{i,j} = f_j \quad i = 1, 2 \tag{39}$$

which is valid for $j = 1, ..., N-1$ in the case $i = 1$; for $j = N+1, ..., 2N-1$ in the case $i = 2$; and $j = 1, ..., 2N-1$ in the case $i = ref$.

Suppose the method 37 - 38 converges to u^*, i.e. :

$$\begin{cases} u_j^* = u_{1,j}^{\infty}, & i = 0, .., N-1 \\ u_j^* = u_{2,j}^{\infty}, & i = N+1, .., 2N \\ u_N^* = u_{1,N}^{\infty} = u_{2,j}^{\infty} \end{cases} \tag{40}$$

Therefor, in the convergence, the boundary condition on the interface $j = N$ reads (using a first order finite difference approximation, for example)

$$\frac{u_N^* - u_{N-1}^*}{\Delta x} = \frac{u_{N+1}^* - u_N^*}{\Delta x} \implies -\frac{u_{N-1}^* - 2u_N^* + u_{N-1}^*}{\Delta x} \tag{41}$$

while the same point, in the modomain problem satisfies the discrete equation (**??**) :

$$-\frac{u_{N-1}^* - 2_N^* + u_{N+1}^*}{\Delta x^2} + \lambda u_N^* = f_N \quad i = 1, 2 \tag{42}$$

References

[1] C. Besse, M. Ehrhardt, and I. Lacroix-Violet. Discrete Artificial Boundary Conditions for the Korteweg-de Vries Equation. working paper or preprint, Jan. 2015.

[2] L. Halpern. Artificial boundary conditions for the linear advection diffusion equation. *"Mathematics of Computation"*, 46(174):425–438, April 1986.