Schwarz waveform relaxation method for one dimensional Schrödinger equation with general potential

Christophe Besse, Feng Xing

Schwarz waveform relaxation method for one dimensional Schrödinger equation with general potential

C. Besse[a], F. Xing[b]

[a] *Institut de Mathématiques de Toulouse UMR5219, Université de Toulouse; CNRS, UPS IMT, F-31062 Toulouse Cedex 9, France.*
[b] *Maison de la Simulation, CEA Saclay France & Laboratoire Paul Painlevé, Université Lille Nord de France.*

Abstract

In this paper, we apply the Schwarz Waveform Relaxation (SWR) method to the one dimensional Schrödinger equation with a general linear or a nonlinear potential. We propose a new algorithm for the Schrödinger equation with time independent linear potential, which is robust and scalable up to 500 subdomains. It reduces significantly computation time compared with the classical algorithms. Concerning the case of time dependent linear potential or the nonlinear potential, we use a preprocessed linear operator for the zero potential case as preconditioner which lead to a preconditioned algorithm. This ensures high scalability. Besides, some newly constructed absorbing boundary conditions are used as the transmission condition and compared numerically.

Keywords: Schrödinger equation, Schwarz Waveform Relaxation method, Absorbing boundary conditions, Parallel algorithms.

1. Introduction

Schwarz waveform relaxation method (SWR) is one class of the domain decomposition methods for time dependent partial differential equations. The time-space domain is decomposed into subdomains. The solution is computed on each subdomain for whole time interval and exchange the time-space boundary value. Some articles are devoted to this method for linear Schrödinger equation [1, 2], advection reaction diffusion equations [3, 4, 5], wave equations [6, 7] and Maxwell's equation [8].

This paper deals with the SWR method without overlap for the one dimensional Schrödinger equation defined on a bounded spatial domain (a_0, b_0), $a_0, b_0 \in \mathbb{R}$ and $t \in (0, T)$. The Schrödinger equation with homogeneous Neu-

Email addresses: christophe.besse@math.univ-toulouse.fr (C. Besse), feng.xing@unice.fr (F. Xing)

mann boundary condition reads

$$\begin{cases} \mathscr{L}u := (i\partial_t + \partial_{xx} + \mathscr{V})u = 0, \ (t,x) \in (0,T) \times (a_0, b_0), \\ u(0,x) = u_0(x), \ x \in (a_0, b_0), \\ \partial_{\mathbf{n}} u(t,x) = 0, \ x = a_0, b_0, \end{cases} \tag{1}$$

where $\mathscr{L}$ is the Schrödinger operator, $\partial_{\mathbf{n}}$ is the normal directive, the initial value $u_0 \in L^2(\mathbb{R})$ and $\mathscr{V}$ is a real potential. We consider both linear and nonlinear potentials:

1. $\mathscr{V} = V(t,x)$,
2. $\mathscr{V} = f(u)$, ex. $\mathscr{V} = |u|^2$.

In order to perform domain decomposition method, the time-space domain $(0,T)\times(a_0,b_0)$ is decomposed into N subdomains $\Theta_j = (0,T)\times\Omega_j$, $\Omega_j = (a_j, b_j)$ without overlap as shown in Figure 1 for $N = 3$.

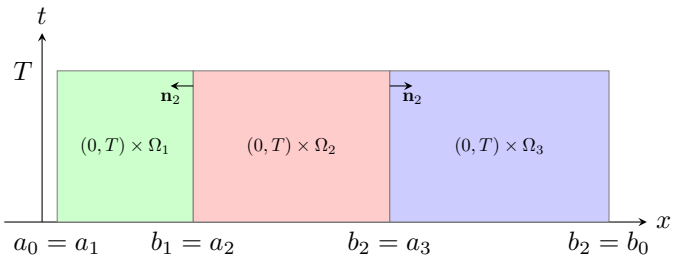

Figure 1: Domain decomposition without overlap, $N = 3$.

The classical SWR algorithm consists in applying the sequence of iterations for $j = 2, 3, ..., N-1$

$$\begin{cases} \mathscr{L}u_j^{k+1} = 0, & (t,x) \in \Theta_j, \\ u_j^{k+1}(0,x) = u_0(x), & x \in \Omega_j, \\ B_j u_j^{k+1} = B_j u_{j-1}^k, & x = a_j, \\ B_j u_j^{k+1} = B_j u_{j+1}^k, & x = b_j. \end{cases} \tag{2}$$

The two extremal subdomains require special treatment since the Neumann boundary condition is imposed in (1) at the points a_0 and b_0.

$$\begin{cases} \mathscr{L}u_1^{k+1} = 0, (t,x) \in \Theta_1, \\ u_1^{k+1}(0,x) = u_0(x), x \in \Omega_1, \\ \partial_{\mathbf{n}_1} u_1^{k+1} = 0, x = a_1, \\ B_1 u_1^{k+1} = B_1 u_2^k, x = b_1, \end{cases} \qquad \begin{cases} \mathscr{L}u_N^{k+1} = 0, (t,x) \in \Theta_N, \\ u_N^{k+1}(0,x) = u_0(x), x \in \Omega_N, \\ B_N u_N^{k+1} = B_N u_{N-1}^k, x = a_N, \\ \partial_{\mathbf{n}_N} u_N^{k+1} = 0, x = b_N. \end{cases}$$

The notation u_j^k denotes the solution on subdomain $\Theta_j = (0,T) \times (a_j, b_j)$ at iteration $k = 0, 1, 2, ...$ of the SWR algorithm. The boundary information is

transmitted with adjacent subdomains Θ_{j-1} and Θ_{j+1} through the transmission operators B_j.

The transmission condition is one of the key issues for this method. For the linear Schrödinger equation, the SWR method with or without overlap is introduced and analyzed by Halpern and Szeftel in [1]. For the decomposition without overlap, if $\mathscr{V}$ is a constant, they use an optimal transmission condition given by the underlying transparent boundary condition. However, the transparent boundary condition is not always available for a variable potential. Robin transmission condition and quasi-optimal transmission condition are therefore used and are named as optimized Schwarz waveform relaxation algorithm and quasi-optimal Schwarz waveform relaxation algorithm respectively. In both cases, the transmission operator is written as

$$B_j = \partial_{\mathbf{n}_j} + S_j, \tag{3}$$

where the operator S_j is

$$\text{Robin}: \ S_j = -ip, \ p \in \mathbb{R}^+, \quad \text{Quasi-optimal}: \ S_j = \sqrt{-i\partial_t - V|_{a_j,b_j}},$$

and $\mathbf{n}_j$ denotes the outwardly unit normal vector at a_j or b_j. Recently, Antoine, Lorin and Bandrauk [9] consider the general Schrödinger equation. On the interface between subdomains, they propose to use recent absorbing conditions as transmission condition, which is also an idea that we follow in this paper.

In recent years, some absorbing operators for one dimensional Schrödinger equation have been constructed by using some adaptations of pseudo-differential techniques [10, 11, 12, 13]. We use them here as the transmission operators in (3) and expect to get good convergence properties.

We are also interested in this article about the effectivness of the method on parallel computers. Another import issue for the method is therefore the scalability. As we know, without additional considerations, the more subdomains are used to decomposed (a_0, b_0), the more iterations are required for SWR algorithm to reach convergence. Thus, the total computation time could hardly decrease significantly. In this paper, we propose two solutions: a new scalable algorithm if the potential is independent of time and a preconditioned algorithm for general potentials.

This paper is organized as follows. In section 2, we present the transmission conditions which are used in this paper for the classical SWR algorithm, and the discretization that plays an important role for the analyses of the interface problem in Section 3. In Section 4 and 5, we present the new algorithm for time independent linear potential and the preconditioned algorithm for general potentials. Some numerical results are shown in Section 6. Finally, we draw a conclusion in the last section.

2. SWR algorithm and discretization

2.1. Transmission conditions

The transmission conditions on boundary points a_j and b_j are given thanks to the relation

$$B_j = \partial_{\mathbf{n}_j} + S_j, \tag{4}$$

where the operators S_j could take different forms. Besides the Robin transmission condition, we propose in this paper to use the operators S_j coming from the artificial boundary conditions for (1) defined in [13, 11, 12, 14] for a linear or nonlinear potential $\mathscr{V}(t,x,u)$. The authors propose three families of conditions written as

$$\partial_{\mathbf{n}} u + S_l^M u = 0,$$

on the boundary of considered computation domain, M denotes the order of the artificial boundary conditions. We index by l these families of boundary conditions: $l = 0$ for potential strategy, $l = 1$ for gauge change strategy and $l = 2$ for Padé approximation strategy. We recall here the definition of operators S_l^M for the different strategies.

Potential strategy $l = 0$ ([13])

$$\begin{aligned}
&\text{Order 2}: \quad S_0^2 = e^{-i\frac{\pi}{4}}\partial_t^{1/2},\\
&\text{Order 3}: \quad S_0^3 = S_0^2 - e^{i\frac{\pi}{4}}\frac{\mathscr{V}}{2}I_t^{1/2},\\
&\text{Order 4}: \quad S_0^4 = S_0^3 - i\frac{\partial_{\mathbf{n}}\mathscr{V}}{4}I_t,
\end{aligned}$$

where the fractional half-order derivative operator $\partial_t^{1/2}$ applied to a function h is defined by

$$\partial_t^{1/2} h(t) = \frac{1}{\sqrt{\pi}}\partial_t \int_0^t \frac{h(s)}{\sqrt{t-s}} ds,$$

the half-order integration operator $I_t^{1/2}$ and the integration operator are given by

$$I_t^{1/2} h(t) = \frac{1}{\sqrt{\pi}} \int_0^t \frac{h(s)}{\sqrt{t-s}} ds, \quad I_t h(t) = \int_0^t h(s) ds.$$

Gauge change strategy $l = 1$ ([11, 12])

$$\begin{aligned}
&\text{Order 2}: \quad S_1^2 = e^{-i\frac{\pi}{4}} e^{i\mathcal{V}(t,x)} \partial_t^{1/2}(e^{-i\mathcal{V}(t,x)}\cdot),\\
&\text{Order 4}: \quad S_1^4 = S_1^2 - i\mathrm{sgn}(\partial_{\mathbf{n}}\mathscr{V})\frac{\sqrt{|\partial_{\mathbf{n}}\mathscr{V}|}}{2} e^{i\mathcal{V}(t,x)} I_t\Big(\frac{\sqrt{|\partial_{\mathbf{n}}\mathscr{V}|}}{2} e^{-i\mathcal{V}(t,x)}\cdot\Big),
\end{aligned}$$

where $sgn(\cdot)$ is the sign function and

$$\mathcal{V}(t,x) = \int_0^t \mathscr{V}(s,x,u(s,x)) ds.$$

Padé approximation strategy $l = 2$ ([11, 12])

$$\text{Order 2}: \quad S_2^2 = -i\sqrt{i\partial_t + \mathscr{V}},$$

$$\text{Order 4}: \quad S_2^4 = S_2^2 + \operatorname{sgn}(\partial_{\mathbf{n}}\mathscr{V})\frac{\sqrt{|\partial_{\mathbf{n}}\mathscr{V}|}}{2}\big(i\partial_t + \mathscr{V}\big)^{-1}\Big(\frac{\sqrt{|\partial_{\mathbf{n}}\mathscr{V}|}}{2}\cdot\Big).$$

2.2. Discretization

The aim of this subsection is to present the discretization of the Schrödinger equation with a linear potential $\mathscr{V} = V(t,x)$ or a nonlinear potential $\mathscr{V} = f(u)$.

2.2.1. Case of linear potential

First, we describe the discretization of the linear Schrödinger equation. We discretize the time interval $(0,T)$ uniformly with N_T intervals and define $\Delta t = T/N_T$ to be the time step. A semi-discrete approximation adapted to the Schrödinger equation on $(0,T)\times(a_j,b_j), j = 1,2,...,N$ is given by the semi-discrete Crank-Nicolson scheme

$$i\frac{u_{j,n}^k - u_{j,n-1}^k}{\Delta t} + \partial_{xx}\frac{u_{j,n}^k + u_{j,n-1}^k}{2} + \frac{V_n + V_{n-1}}{2}\frac{u_{j,n}^k + u_{j,n-1}^k}{2} = 0,\ 1 \leqslant n \leqslant N_T,$$

and $u_{j,0}^k = u(0,x)$ for $x \in (a_j,b_j)$. The unknown function $u_{j,n}^k(x)$ is an approximation of the solution $u_j^k(n\Delta t,x)$ to the Schrödinger equation at time $t_n = n\Delta t$ on subdomain Ω_j and at iteration k. We define the approximation of the potential $V_n(x) = V(t_n,x)$.

For implementation issue, it is useful to introduce new variables $v_{j,n}^k = (u_{j,n}^k + u_{j,n-1}^k)/2$ with $v_{j,0}^k = u_{j,0}^k$. The scheme could be written as

$$2i\frac{v_{j,n}^k}{\Delta t} + \partial_{xx}v_{j,n}^k + W_n v_{j,n}^k = 2i\frac{u_{j,n-1}^k}{\Delta t}, \tag{5}$$

with $W_n = (V_n + V_{n-1})/2$. The spatial approximation is realized thanks to a classical P_1 finite element method. The use of transmission condition gives the following boundary conditions for each subdomain

$$\begin{cases} \partial_{\mathbf{n}_j}v_{j,n}^k + \overline{S}v_{j,n}^k = \partial_{\mathbf{n}_j}v_{j-1,n}^{k-1} + \overline{S}v_{j-1,n}^{k-1},\ x = a_j, \\ \partial_{\mathbf{n}_j}v_{j,n}^k + \overline{S}v_{j,n}^k = \partial_{\mathbf{n}_j}v_{j+1,n}^{k-1} + \overline{S}v_{j+1,n}^{k-1},\ x = b_j, \end{cases} \tag{6}$$

with special treatments for the two extreme subdomains

$$\partial_{\mathbf{n}_1}v_{1,n}^k = 0,\ x = a_1, \quad \partial_{\mathbf{n}_N}v_{N,n}^k = 0,\ x = b_N,$$

where $\overline{S}$ is a semi-discretization of S. For each strategy, $\overline{S}$ is given by

Potential strategy $l = 0$

$$\begin{aligned}
&\text{Order} \quad 2: \quad \overline{S}_0^2 v_{j,n}^k = e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}\sum_{s=0}^{n}\beta_{n-s}v_{j,s}^k,\\
&\text{Order} \quad 3: \quad \overline{S}_0^3 v_{j,n}^k = \overline{S}_0^2 v_{j,n}^k - e^{i\pi/4}\sqrt{\frac{\Delta t}{2}}\frac{W_n}{2}\sum_{s=0}^{n}\alpha_{n-s}v_{j,s}^k,\\
&\text{Order} \quad 4: \quad \overline{S}_0^4 v_{j,n}^k = \overline{S}_0^3 v_{j,n}^k - i\frac{\partial_{\mathbf{n}_j}W_n}{4}\frac{\Delta t}{2}\sum_{s=0}^{n}\gamma_{n-s}v_{j,s}^k,
\end{aligned}$$

where

$$(\alpha_0,\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,...) = (1,1,\frac{1}{2},\frac{1}{2},\frac{3}{8},\frac{3}{8},\frac{3\cdot 5}{2\cdot 4\cdot 6},...),\ \beta_s = (-1)^s\alpha_s, \forall s \geqslant 0,$$
$$(\gamma_0,\gamma_1,\gamma_2,\gamma_3,...) = (1,2,2,2,...).$$

Gauge change strategy $l = 1$

$$\begin{aligned}
&\text{Order} \quad 2: \quad \overline{S}_1^2 v_{j,n}^k = e^{-i\pi/4}e^{i\mathcal{W}_n}\sqrt{\frac{2}{\Delta t}}\sum_{s=0}^{n}\beta_{n-s}e^{-i\mathcal{W}_s}v_{j,s}^k,\\
&\text{Order} \quad 4: \quad \overline{S}_1^4 v_{j,n}^k = \overline{S}_1^2 v_{j,n}^k\\
&\qquad - isgn(\partial_{\mathbf{n}_j}W_n)\frac{\sqrt{|\partial_{\mathbf{n}_j}W_n|}}{2}e^{i\mathcal{W}_n}\frac{\Delta t}{2}\sum_{s=0}^{n}\gamma_{n-s}\frac{\sqrt{|\partial_{\mathbf{n}_j}W_s|}}{2}e^{-i\mathcal{W}_s}v_{j,s}^k,
\end{aligned}$$

where $\mathcal{W}_n = \frac{\mathcal{V}_n+\mathcal{V}_{n-1}}{2}$ and $\mathcal{V}_n(x) = \int_0^{t_n} V(s,x)ds$.

Padé approximation strategy $l = 2$

$$\begin{aligned}
\overline{S}_2^2 v_{j,n}^k =& -i\big(\sum_{s=0}^{m}a_s^m\big)v_{j,n}^k + i\sum_{s=1}^{m}a_s^m d_s^m\frac{1}{\frac{2i}{\Delta t}+W_n+d_s^m}v_{j,n}^k\\
&+ i\sum_{s=1}^{m}a_s^m d_s^m\frac{\frac{2i}{\Delta t}}{\frac{2i}{\Delta t}+W_n+d_k^m}\varphi_{j,n-1}^s,\\
\overline{S}_2^4 v_{j,n}^k =& \overline{S}_2^2 v_{j,n}^k + \frac{\partial_{\mathbf{n}_j}W_n}{4}\frac{1}{\frac{2i}{\Delta t}+W_n}v_{j,n}^k\\
&+ sgn(\partial_{\mathbf{n}_j}W_n)\frac{\sqrt{|\partial_{\mathbf{n}_j}W_n|}}{2}\frac{\frac{2i}{\Delta t}}{\frac{2i}{\Delta t}+W_n}\psi_{j,n-1},
\end{aligned}$$

where $\varphi_{j,n}^s, \phi_{j,n}, s = 1,2,...,m$ are introduced as auxiliary functions

$$\begin{cases}
\varphi_{j,n-\frac{1}{2}}^s = \frac{1}{\frac{2i}{\Delta t}+W_n+d_s^m}v_{j,n}^k + \frac{\frac{2i}{\Delta t}}{\frac{2i}{\Delta t}+W_n+d_s^m}\varphi_{j,n-1}^s,\ s = 1,2,...,m\\
\varphi_{j,n}^s = 2\varphi_{j,n-\frac{1}{2}}^s - \varphi_{j,n-1}^s,\\
\varphi_{j,0}^s = 0,
\end{cases}$$

and

$$\begin{cases} \psi_{n-\frac{1}{2}} = \frac{\sqrt{|\partial_{\mathbf{n}_j} W_n|}}{2} \frac{1}{\frac{2i}{\Delta t}+W_n} v^k_{j,n} + \frac{\frac{2i}{\Delta t}}{\frac{2i}{\Delta t}+W_n} \psi_{j,n-1}, \\ \psi_{j,n} = 2\psi_{j,n-\frac{1}{2}} - \psi_{j,n-1}, \\ \psi_{j,0} = 0. \end{cases}$$

We also recall here the Robin transmission condition and its approximation

$$S = S_p = -ip, \quad \overline{S} v^k_{j,n} = \overline{S}_p v^k_{j,n} = -ip \cdot v^k_{j,n}, \quad p \in \mathbb{R}^+.$$

We propose below to rewrite (6) by using fluxes, which are defined at interfaces by

$$l^k_{j,n} = \partial_{\mathbf{n}_j} v^k_{j,n}(a_j) + \overline{S} v^k_{j,n}(a_j),\ r^k_{j,n} = \partial_{\mathbf{n}_j} v^k_{j,n}(b_j) + \overline{S} v^k_{j,n}(b_j),\ j = 1, 2, ..., N,$$

with the exception for $l^k_{1,n} = r^k_{N,n} = 0$. It is obvious that, on each subdomain, the boundary conditions are

$$\begin{cases} \partial_{\mathbf{n}_j} v^k_{j,n} + \overline{S} v^k_{j,n} = l^k_{j,n},\ x = a_j, \\ \partial_{\mathbf{n}_j} v^k_{j,n} + \overline{S} v^k_{j,n} = r^k_{j,n},\ x = b_j. \end{cases} \tag{7}$$

For the transmission condition S^2_0, S^3_0, S^2_1 and S^2_2 which do not contain the normal derivative of potential W_n, using (6), we have

$$\begin{aligned} r^k_{1,n} &= \partial_{\mathbf{n}_1} v^k_{1,n}(b_1) + \overline{S} v^k_{j,n}(b_1) = \partial_{\mathbf{n}_1} v^{k-1}_{2,n}(a_2) + \overline{S} v^{k-1}_{2,n}(a_2) \\ &\quad - \left(\partial_{\mathbf{n}_2} v^{k-1}_{2,n}(a_2) + \overline{S} v^{k-1}_{2,n}(a_2) \right) + 2\overline{S} v^{k-1}_{2,n}(a_2) = -l^{k-1}_{2,n} + 2\overline{S} v^{k-1}_{2,n}(a_2). \end{aligned}$$

The transmission conditions could therefore be rewritten as

$$\begin{cases} l^k_{1,n} = 0, \quad l^k_{j,n} = -r^{k-1}_{j-1,n} + 2\overline{S} v^{k-1}_{j-1,n}(b_{j-1}),\ j = 2, ..., N, \\ r^k_{1,N} = 0,\ r^k_{j,n} = -l^{k-1}_{j+1,n} + 2\overline{S} v^{k-1}_{j+1,n}(a_{j+1}),\ j = 1, 2, ..., N-1. \end{cases} \tag{8}$$

Dealing with the transmission conditions S^4_0, S^4_1 and S^4_2, we could also obtain similar formulas to (8). We can therefore replace the boundary conditions (6) for the N local problems (5) by (7) and fluxes definition (8).

Let us denote by $\mathbf{v}^k_{j,n}$ (resp. $\mathbf{u}^k_{j,n}$) the nodal P_1 interpolation vector of $v^k_{j,n}$ (resp. $u^k_{j,n}$) with N_j nodes, $\mathbb{M}_j$ the mass matrix, $\mathbb{S}_j$ the stiffness matrix and $\mathbb{M}_{j,W_n}$ the generalized mass matrix with respect to $\int_{a_j}^{b_j} W_n v \phi dx$, $j = 1, 2, ..., N$. Thus, the matrix formulation of the N local problems is given by

$$(\mathbb{A}_{j,n} - \mathbb{B}_{j,n}) \mathbf{v}^k_{j,n} = \frac{2i}{\Delta t} \mathbb{M}_j \mathbf{u}^k_{j,n-1} + \mathbf{b}^k_{j,n} - Q^T_j (l^k_n, r^k_n)^T, \tag{9}$$

where $\mathbb{A}_{j,n} = \frac{2i}{\Delta t} \mathbb{M}_j - \mathbb{S}_j + \mathbb{M}_{j,W_n}$ and "$.^T$" is the standard notation of the transpose of a matrix or a vector. The restriction matrix Q_j is defined by

$$Q_j = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \in \mathbb{C}^{2 \times N_j}.$$

$\mathbb{B}_{j,n} \in \mathbb{C}^{N_j \times N_j}$ (resp. $\mathbf{b}^k_{j,n} \in \mathbb{C}^{N_j}$) represent the boundary matrix (resp. vector) associated with the boundary condition at time step n, which depends on the transmission condition. The discrete form of the transmission condition (8) is given by

$$\begin{cases} l^k_{j,n} = -r^k_{j-1,n} + 2\widetilde{S}(Q_{j-1,r}\mathbf{v}^k_{j-1,n}),\ j = 1, 2, ..., N-1, \\ r^k_{j,n} = -l^k_{j+1,n} + 2\widetilde{S}(Q_{j+1,l}\mathbf{v}^k_{j+1,n}),\ j = 2, 3, ..., N. \end{cases} \tag{10}$$

where $Q_{j,l} = (1, 0, \cdots, 0, 0) \in \mathbb{C}^{N_j}$, $Q_{j,r} = (0, 0, \cdots, 0, 1) \in \mathbb{C}^{N_j}$. $\widetilde{S}$ is the fully discrete version of $\overline{S}$. For example the transmission condition S_0^2 leads to

$$\widetilde{S}_0^2(Q_{j,l}\mathbf{v}^k_{j,n}) = e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}\sum_{s=0}^{n}\beta_{n-s}(Q_{j,l}\mathbf{v}^k_{j,s}),$$

$$\widetilde{S}_0^2(Q_{j,r}\mathbf{v}^k_{j,n}) = e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}\sum_{s=0}^{n}\beta_{n-s}(Q_{j,r}\mathbf{v}^k_{j,s}).$$

2.2.2. Case of nonlinear potential

If the potential is nonlinear $\mathscr{V} = f(u)$, we propose to use the usual scheme developed by Durán- Sanz Serna [15]

$$i\frac{u^k_{j,n} - u^k_{j,n-1}}{\Delta t} + \partial_{xx}\frac{u^k_{j,n} + u^k_{j,n-1}}{2} + f(\frac{u^k_{j,n} + u^k_{j,n-1}}{2})\frac{u^k_{j,n} + u^k_{j,n-1}}{2} = 0,\ 1 \leqslant n \leqslant N_T,$$

By using the notations defined in the previous subsection, this schema reads as

$$2i\frac{v^k_{j,n}}{\Delta t} + \partial_{xx}v^k_{j,n} + f(v^k_{j,n})v^k_{j,n} = 2i\frac{u^k_{j,n-1}}{\Delta t}. \tag{11}$$

As in the previous subsection, we use a P_1 finite element method to deal with the space variable approximation. Since the problem is nonlinear, the computation of $v^k_{j,n}$ is made by a fixed point procedure. At a given time $t = t_n$, we take $\boldsymbol{\zeta}^0_j = \mathbf{v}^k_{j,n-1}$ and compute the solution $\mathbf{v}^k_{j,n}$ as the limit of the iterative scheme with respect to s:

$$\left(\frac{2i}{\Delta t}\mathbb{M}_j - \mathbb{S}_j - \mathbb{B}_{j,n}\right)\boldsymbol{\zeta}^{s+1}_j = \frac{2i}{\Delta t}\mathbb{M}_j\mathbf{u}^k_{j,n-1} - \mathbf{b}_{j,f(\zeta^s_j)} + \mathbf{b}^k_{j,n} - Q_j^T(l^k_n, r^k_n)^T, \tag{12}$$

where $\mathbf{b}_{j,f(v)}$ is the vector associated with $\int_{a_j}^{b_j} f(v)v\phi dx$. The matrix $\mathbb{B}_{j,n}$ and the vector $\mathbf{b}^k_{j,n}$ depend on the transmission operator. The discrete form of the transmission conditions is similar to (10) obtained for linear potential.

3. Interface problem

The N problems (9) and (12) on each subdomain could be written globally. Let us define the global interface vector g^k at iteration k by

$$g^k = \big(\underbrace{r^k_{1,1}, r^k_{1,2}, ..., r^k_{1,N_T}}_{j=1}, \cdots, \underbrace{l^k_{j,1}, ..., l^k_{j,N_T}, r^k_{j,1}, ..., r^k_{j,N_T}}_{j}, \cdots, \underbrace{l^k_{N,1}, l^k_{N,2}, ..., l^k_{N,N_T}}_{j=N}\big)^T.$$

Considering the transmission conditions with flux (10), it is not hard to see that there exist an operator $\mathcal{R}$ such that

$$g^{k+1} = \mathcal{R} g^k. \tag{13}$$

The interface operator $\mathcal{R}$ is linear or nonlinear depending on the linearity of $\mathscr{V}$. We focus below on the interface problem for the linear potential $\mathscr{V} = V(t,x)$, especially for $\mathscr{V} = V(x)$.

For the transmission conditions presented in Section 2, we are going to show that if $\mathscr{V} = V(t,x)$, then

$$g^{k+1} = \mathcal{R} g^k = \mathcal{L} g^k + d, \tag{14}$$

where $\mathcal{L}$ is a block matrix

$$\mathcal{L} = \begin{pmatrix} & X^{2,1} & X^{2,2} & & & & & & \\ X^{1,4} & & & & & & & & \\ & & & X^{3,1} & X^{3,2} & & & & \\ & X^{2,3} & X^{2,4} & & & & & & \\ & & & & & \cdots & & & \\ & & & X^{3,3} & X^{3,4} & & & & \\ & & & & & & X^{N-1,1} & X^{N-1,2} & \\ & & & & & \cdots & & & \\ & & & & & & & & X^{N,1} \\ & & & & & & X^{N-1,3} & X^{N-1,4} & \end{pmatrix}, \tag{15}$$

with $X^{j,p} \in \mathbb{C}^{N_T \times N_T}$, $j = 1,2,...,N$, $p = 1,2,3,4$ and d is a vector

$$d = \left(d_{1,r}^T, d_{2,l}^T, d_{2,r}^T, \cdots, d_{N,l}^T\right)^T \in \mathbb{C}^{(2N-2)\times N_T},\ d_{j,l},\ d_{j,r} \in \mathbb{C}^{N_T}. \tag{16}$$

It is easy to see that the formula (14) is equivalent to

1. for $j = 1$,

$$\begin{pmatrix} l_{2,1}^{k+1} \\ l_{2,2}^{k+1} \\ \vdots \\ l_{2,N_T}^{k+1} \end{pmatrix} = X^{1,4} \begin{pmatrix} r_{1,1}^{k} \\ r_{1,2}^{k} \\ \vdots \\ r_{1,N_T}^{k} \end{pmatrix} + d_{2,l},$$

2. for $j = 2,...,N-1$,

$$\begin{aligned} \begin{pmatrix} r_{j-1,1}^{k+1} \\ r_{j-1,2}^{k+1} \\ \vdots \\ r_{j-1,N_T}^{k+1} \end{pmatrix} &= X^{j,1} \begin{pmatrix} l_{j,1}^{k} \\ l_{j,2}^{k} \\ \vdots \\ l_{j,N_T}^{k} \end{pmatrix} + X^{j,2} \begin{pmatrix} r_{j,1}^{k} \\ r_{j,2}^{k} \\ \vdots \\ r_{j,N_T}^{k} \end{pmatrix} + d_{j-1,r}, \\ \begin{pmatrix} l_{j+1,1}^{k+1} \\ l_{j+1,2}^{k+1} \\ \vdots \\ l_{j+1,N_T}^{k+1} \end{pmatrix} &= X^{j,3} \begin{pmatrix} l_{j,1}^{k} \\ l_{j,2}^{k} \\ \vdots \\ l_{j,N_T}^{k} \end{pmatrix} + X^{j,4} \begin{pmatrix} r_{j,1}^{k} \\ r_{j,2}^{k} \\ \vdots \\ r_{j,N_T}^{k} \end{pmatrix} + d_{j+1,l}, \end{aligned} \tag{17}$$

3. for $j = N$,

$$\begin{pmatrix} r_{N-1,1}^{k+1} \\ r_{N-1,2}^{k+1} \\ \vdots \\ r_{N-1,N_T}^{k+1} \end{pmatrix} = X^{N,1} \begin{pmatrix} l_{N,1}^{k} \\ l_{N,2}^{k} \\ \vdots \\ l_{N,N_T}^{k} \end{pmatrix} + d_{N-1,r}.$$

Proposition 1. *For the transmission condition involving the operator S_0^2, in the case of linear potential $\mathscr{V} = V(t,x)$, if we assume that the matrices $\mathbb{A}_{j,n} - \mathbb{B}_{j,n}$ $n = 1, 2, ..., N_T$ are not singular, then the N equations (9) could be written in the global form of interface problem (14)*

$$g^{k+1} = \mathcal{L} g^k + d.$$

Proof. First, according to (9), we have

$$\begin{aligned}
(\mathbb{A}_{j,1} - \mathbb{B}_{j,1})\mathbf{v}_{j,1}^k &= \frac{2i}{\Delta t}\mathbb{M}_j \mathbf{u}_{j,0} + e^{-\frac{i\pi}{4}} Q_j^T \sqrt{\frac{2}{\Delta t}} \beta_1 Q_j \mathbf{v}_{j,0}^k - Q_j^T (l_{j,1}^k, r_{j,1}^k)^T, \\
(\mathbb{A}_{j,n} - \mathbb{B}_{j,n})\mathbf{v}_{j,n}^k &= \frac{2i}{\Delta t}\mathbb{M}_j \mathbf{u}_{j,n-1}^k + e^{-\frac{i\pi}{4}} \sqrt{\frac{2}{\Delta t}} Q_j^T \sum_{q=0}^{n-1} \beta_{2-q} Q_j \mathbf{v}_{j,q}^k - Q_j^T (l_{j,n}^k, r_{j,n}^k)^T \\
&= \frac{4i}{\Delta t}\mathbb{M}_j \mathbf{v}_{j,n-1}^k - \frac{2i}{\Delta t}\mathbb{M}_j \mathbf{u}_{j,n-2}^k + e^{-\frac{i\pi}{4}} \sqrt{\frac{2}{\Delta t}} Q_j^T \sum_{q=0}^{n-1} \beta_{2-q} Q_j \mathbf{v}_{j,q}^k - Q_j^T (l_{j,n}^k, r_{j,n}^k)^T, \\
&= \sum_{q=1}^{n-1} \Big((-1)^{n-1-q} \frac{4i}{\Delta t}\mathbb{M}_j + e^{-i\pi/4} \sqrt{\frac{2}{\Delta t}} \beta_{n-q} Q_j^T Q_j \Big) \mathbf{v}_{j,q}^k, \\
&\quad + \Big((-1)^{n-1} \frac{2i}{\Delta t}\mathbb{M}_j + e^{-i\pi/4} \sqrt{\frac{2}{\Delta t}} \beta_n Q_j^T Q_j \Big) \mathbf{u}_{j,0} - Q_j^T (l_{j,n}^k, r_{j,n}^k)^T, \ n \geqslant 2,
\end{aligned}$$

where we recall that $\mathbf{v}_{j,0}^k = \mathbf{u}_{j,0}$. Thus, we could see that

$$\begin{aligned}
\mathbf{v}_{j,n}^k = &- (\mathbb{A}_{j,n} - \mathbb{B}_{j,n})^{-1} Q_j^T (l_{j,n}^k, r_{j,n}^k)^T \\
&+ (\mathbb{A}_{j,n} - \mathbb{B}_{j,n})^{-1} \sum_{q=1}^{n-1} \Big((-1)^{n-1-q} \frac{4i}{\Delta t}\mathbb{M}_j + e^{-i\pi/4} \sqrt{\frac{2}{\Delta t}} \beta_{n-q} Q_j^T Q_j \Big) \mathbf{v}_{j,q}^k \\
&+ (\mathbb{A}_{j,n} - \mathbb{B}_{j,n})^{-1} \Big((-1)^{n-1} \frac{2i}{\Delta t}\mathbb{M}_j + e^{-i\pi/4} \sqrt{\frac{2}{\Delta t}} \beta_n Q_j^T Q_j \Big) \mathbf{u}_{j,0}.
\end{aligned} \tag{18}$$

By induction on n, it is easy to see that $\mathbf{v}_{j,n}^k$ is a linear function of $l_{j,s}^k$ and $r_{j,s}^k, s = 1, 2, ..., n$. Then considering the formulas (10), in order to finish the proof, we need only verify that $\widetilde{S}(Q_{j,l}\mathbf{v}_{j,n}^k)$ and $\widetilde{S}(Q_{j,r}\mathbf{v}_{j,n}^k)$ are linear functions of $\mathbf{v}_{j,s}^k, s = 1, 2, ..., n$.

Proposition 2. *For any transmission condition presented in Section 2, assuming that the matrices $\mathbb{A}_{j,n} - \mathbb{B}_{j,n}$, $n = 1, 2, ..., N_T$ are not singular, then the interface problem in the case of linear potential $\mathscr{V} = V(t, x)$ could be written in the global form (14).*

Proof. The proof is quite similar than that of the previous proposition. For each transmission condition, we only need to recalculate the expression of $\mathbf{v}_{j,n}^k$.

We now turn to the structure of sub-blocks for $\mathscr{V} = V(x)$ and $j = 2, 3, ..., N-1$,

$$X^{j,1} = \{x_{n,s}^{j,1}\}_{1\leqslant n,s\leqslant N_T}, \quad X^{j,2} = \{x_{n,s}^{j,2}\}_{1\leqslant n,s\leqslant N_T},$$
$$X^{j,3} = \{x_{n,s}^{j,3}\}_{1\leqslant n,s\leqslant N_T}, \quad X^{j,4} = \{x_{n,s}^{j,4}\}_{1\leqslant n,s\leqslant N_T}.$$

and $X^{1,4} = \{x_{n,s}^{1,4}\}_{1\leqslant n,s\leqslant N_T}$ and $X^{N-1,1} = \{x_{n,s}^{N-1,1}\}_{1\leqslant n,s\leqslant N_T}$. For 5 time steps, this structure is described below

$$\begin{pmatrix} \star & & & & \\ \times & \star & & & \\ \circ & \times & \star & & \\ \triangleleft & \circ & \times & \star & \\ \diamond & \triangleleft & \circ & \times & \star \end{pmatrix}, \ N_T = 5.$$

thus, each sub-diagonal have an identical element.

Proposition 3. *For the transmission condition involving the operator S_0^2, if $\mathscr{V} = V(x)$ and assuming that $\mathbb{A}_{j,n} - \mathbb{B}_{j,n}$, $n = 1, 2, ..., N_T$ are not singular, then the matrices $X^{1,4}$ $X^{j,1}, X^{j,2}, X^{j,3}, X^{j,4}, j = 2, 3, ..., N-1$ and $X^{N,1}$ are lower triangular matrices and they satisfy*

$$\begin{aligned} &x_{n,s}^{1,4} = x_{n-1,s-1}^{1,4}, \\ &x_{n,s}^{j,1} = x_{n-1,s-1}^{j,1}, \ x_{n,s}^{j,2} = x_{n-1,s-1}^{j,2}, \\ &x_{n,s}^{j,3} = x_{n-1,s-1}^{j,3}, \ x_{n,s}^{j,4} = x_{n-1,s-1}^{j,4}, j = 2, 3, ..., N-1, \\ &x_{n,s}^{N,1} = x_{n-1,s-1}^{N,1}, \end{aligned}$$

for $2 \leqslant s \leqslant n \leqslant N_T$.

Proof. Without loss of generality, we consider here $j = 2, 3, ..., N-1$. First, we design

$$\mathbb{Y}_{n,q}^j = \begin{cases} -(\mathbb{A}_{j,n} - \mathbb{B}_{j,n})^{-1}, \ q = n, \\ (\mathbb{A}_{j,n} - \mathbb{B}_{j,n})^{-1}\Big((-1)^{n-1-q}\frac{4i}{\Delta t}\mathbb{M}_j + e^{\frac{-i\pi}{4}}\sqrt{\frac{2}{\Delta t}}\beta_{n-q}Q_j^TQ_j\Big), \ q = 1, 2, ..., n-1. \end{cases}$$

If the linear potential $\mathscr{V} = V(x)$ is independent of time, then it is easy to see

$$\mathbb{A}_{j,1} = \mathbb{A}_{j,2} = \cdots = \mathbb{A}_{j,N_T}, \ \mathbb{B}_{j,1} = \mathbb{B}_{j,2} = \cdots = \mathbb{B}_{j,N_T}.$$

Thus for $2 \leqslant s \leqslant n \leqslant N_T$,

$$\mathbb{Y}^j_{n,s} = \mathbb{Y}^j_{n-1,s-1}. \tag{19}$$

Then, according to (18), we have

$$\mathbf{v}^k_{j,n} = \mathbb{Y}^j_{n,n} Q_j^T (l^k_{j,n}, r^k_{j,n})^T + \sum_{q=1}^{n-1} \mathbb{Y}^j_{n,q} \mathbf{v}^k_{j,q} + \mathbb{U}_{j,n} \mathbf{u}_{j,0}. \tag{20}$$

where $\mathbb{U}_{j,n} = (\mathbb{A}_{j,n} - \mathbb{B}_{j,n})^{-1} \Big(\frac{2i}{\Delta t} (-1)^{n-1} \mathbb{M}_j + e^{-i\pi/4} \sqrt{\frac{2}{\Delta t}} \beta_n Q_j^T Q_j \Big)$. By induction, we can obtain an expression of $\mathbf{v}^k_{j,n}$:

$$\mathbf{v}^k_{j,n} = \sum_{q=1}^{n} \mathbb{L}^j_{n,q} Q_j^T (l^k_{j,q}, r^k_{j,q})^T + U_{j,n} \mathbf{u}_{j,0}, \tag{21}$$

where $\mathbb{L}^j_{n,q}$, $q = 1, 2, ..., n$ and $U_{j,n}$ are matrix. For example, $\mathbb{L}^j_{n,n} = \mathbb{Y}^j_{n,n}$. We are going to show that for $2 \leqslant s \leqslant n \leqslant N_T$,

$$\mathbb{L}^j_{n,s} = \mathbb{L}^j_{n-1,s-1}. \tag{22}$$

Replacing $\mathbf{v}^k_{j,q}$ in (20) by (21), we have

$$\begin{aligned}
\mathbf{v}^k_{j,n} =& \mathbb{Y}^j_{n,n} Q_j^T (l^k_{j,n}, r^k_{j,n})^T + \sum_{q=1}^{n-1} \mathbb{Y}^j_{n,q} \Big(\sum_{p=1}^{q} \mathbb{L}^j_{q,p} Q_j^T (l^k_{j,p}, n^k_{j,p})^T + U_{j,q} \mathbf{u}_{j,0} \Big) + \mathbb{U}_{j,n} \mathbf{u}_{j,0} \\
=& \mathbb{Y}^j_{n,n} Q_j^T (l^k_{j,n}, r^k_{j,n})^T + \sum_{p=1}^{n-1} \Big(\sum_{q=p}^{n-1} \mathbb{Y}^j_{n,q} \mathbb{L}^j_{q,p} \Big) Q_j^T (l^k_{j,p}, r^k_{j,p})^T + \Big(\sum_{q=1}^{n-1} \mathbb{Y}^j_{n,q} U_{j,q} + \mathbb{U}_{j,n} \Big) \mathbf{u}_{j,0}.
\end{aligned}$$

Comparing the above formula with (21), we have

$$\mathbb{L}_{n,s} = \begin{cases} \mathbb{Y}^j_{n,n}, \\ \sum\limits_{q=s}^{n-1} \mathbb{Y}^j_{n,q} \mathbb{L}^j_{q,s}, 1 \leqslant s < n, \end{cases} \Rightarrow \mathbb{L}_{n-1,s-1} = \begin{cases} \mathbb{Y}^j_{n-1,n-1}, \\ \sum\limits_{q=s-1}^{n-2} \mathbb{Y}^j_{n-1,q} \mathbb{L}^j_{q,s-1}, 2 \leqslant s < n. \end{cases} \tag{23}$$

By using (19) and by induction on n, we get

$$\begin{aligned}
&\mathbb{L}_{n,n} = \mathbb{Y}_{n,n} = \mathbb{Y}_{n-1,n-1} = \mathbb{L}_{n-1,n-1}, \\
&\mathbb{L}_{n,s} = \sum_{q=s}^{n-1} \mathbb{Y}^j_{n,q} \mathbb{L}^j_{q,s} = \sum_{q=s}^{n-1} \mathbb{Y}^j_{n-1,q-1} \mathbb{L}^j_{q-1,s-1} = \sum_{q=s-1}^{n-2} \mathbb{Y}^j_{n-1,q} \mathbb{L}^j_{q,s-1} = \mathbb{L}_{n-1,s-1}, 2 \leqslant s < n.
\end{aligned}$$

The formula (22) is thus demonstrated.

Then we replace $\mathbf{v}_{j,k}^n$ in the first two formulas of (10) by (21). We get

$$\begin{aligned} l_{j+1,n}^{k+1} =& - r_{j,n}^k + 2e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}\sum_{p=1}^{n}\beta_{n-p}\sum_{q=1}^{p}\mathbb{L}_{p,q}^j Q_j^T(l_{j,q}^k, r_{j,q}^k)^T + R_{l,j,n}^k \\ =& - r_{j,n}^k + 2e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}\sum_{q=1}^{n} Q_{j,r}\Big(\sum_{p=q}^{n}\beta_{n-p}\mathbb{L}_{p,q}^j\Big)Q_j^T(l_{j,q}^k, r_{j,q}^k)^T + R_{l,j,n}^k, \\ r_{j-1,n}^{k+1} =& - l_{j,n}^k + 2e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}\sum_{q=1}^{n} Q_{j,l}\Big(\sum_{p=q}^{n}\beta_{n-p}\mathbb{L}_{p,q}^j\Big)Q_j^T(l_{j,q}^k, r_{j,q}^k)^T + R_{r,j,n}^k, \end{aligned} \tag{24}$$

where we denotes the terms that are independent of $l_{j,s}^k$ and $r_{j,s}^k$, $s = 1, 2, ..., N_T$ by remainder terms $R_{l,r}$ to make the proof more readable.

Moreover, according to (17), we have

$$\begin{aligned} l_{2,n}^{k+1} = \sum_{s=1}^{N_T} x_{n,s}^{1,4} r_{1,s}^k + d_{2,l,n},\ r_{N-1,n}^{k+1} = \sum_{s=1}^{N_T} x_{n,s}^{N,1} l_{N,s}^k + d_{N-1,r,n}, \\ r_{j-1,n}^{k+1} = \sum_{s=1}^{N_T} x_{n,s}^{j,1} l_{j,s}^k + \sum_{s=1}^{N_T} x_{n,s}^{j,2} r_{j,s}^k + d_{j-1,r,n}, \\ l_{j+1,n}^{k+1} = \sum_{s=1}^{N_T} x_{n,s}^{j,3} l_{j,s}^k + \sum_{s=1}^{N_T} x_{n,s}^{j,4} r_{j,s}^k + d_{j+1,l,n}. \end{aligned}$$

where $d_{j-1,l,n}$ and $d_{j+1,r,n}$ denote the n-th element of $d_{j-1,l}$ and $d_{j+1,r}$ respectively.

Comparing the above formula with (24), we have for $1 \leqslant n < s \leqslant N_T$,

$$x_{n,s}^{j,1} = x_{n,s}^{j,2} = x_{n,s}^{j,3} = x_{n,s}^{j,4} = 0,$$

and for $1 \leqslant s \leqslant n \leqslant N_T$,

$$\begin{aligned} x_{n,s}^{j,1} = -1 + 2c_2 Q_{j,l}\Big(\sum_{p=s}^{n}\beta_{n-p}\mathbb{L}_{p,s}^j\Big)Q_{j,l}^T,\ x_{n,s}^{j,2} = 2c_2 Q_{j,l}\Big(\sum_{p=s}^{n}\beta_{n-p}\mathbb{L}_{p,s}^j\Big)Q_{j,r}^T, \\ x_{n,s}^{j,3} = 2c_2 Q_{j,r}\Big(\sum_{p=s}^{n}\beta_{n-p}\mathbb{L}_{p,s}^j\Big)Q_{j,l}^T,\ x_{n,s}^{j,4} = -1 + 2c_2 Q_{j,r}\Big(\sum_{p=s}^{n}\beta_{n-p}\mathbb{L}_{p,s}^j\Big)Q_{j,r}^T, \end{aligned}$$

where $c_2 = e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}$ and we use $Q_j^T(l_{j,q}^k, r_{j,q}^k)^T = Q_{j,l}^T l_{j,q}^k + Q_{j,r}^T r_{j,q}^k$.

Finally, using (22), we have for $2 \leqslant s \leqslant n \leqslant N_T$,

$$\begin{aligned} x_{n,s}^{j,1} &= -1 + 2e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}Q_{j,l}\Big(\sum_{p=s}^{n}\beta_{n-p}\mathbb{L}_{p,s}^{j}\Big)Q_{j,l}^{T} \\ &= -1 + 2e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}\ Q_{j,l}\Big(\sum_{p=s}^{n}\beta_{n-p}\mathbb{L}_{p-1,s-1}^{j}\Big)Q_{j,l}^{T} \\ &= -1 + 2e^{-i\pi/4}\sqrt{\frac{2}{\Delta t}}Q_{j,l}\Big(\sum_{p=s-1}^{n-1}\beta_{n-1-p}\mathbb{L}_{p,s}^{j}\Big)Q_{j,l}^{T} = x_{n-1,s-1}^{j,1}. \end{aligned}$$

In the same way, we can prove that $x_{n,s}^{j,2} = x_{n-1,s-1}^{j,2}$, $x_{n,s}^{j,3} = x_{n-1,s-1}^{j,3}$ and $x_{n,s}^{j,4} = x_{n-1,s-1}^{j,4}$.

Proposition 4. *With any transmission condition presented in Section 2, if $\mathscr{V} = V(x)$ and assuming that $\mathbb{A}_{j,n} - \mathbb{B}_{j,n}$, $n = 1, 2, ..., N_T$ are not singular, then the matrices $X^{1,4}$, $X^{j,1}$, $X^{j,2}$, $X^{j,3}$, $X^{j,4}$, $j = 2, 3, ..., N-1$ and $X^{N,1}$ are lower triangular matrices and they satisfy*

$$\begin{aligned} &x_{n,s}^{1,4} = x_{n-1,s-1}^{1,4}, \\ &x_{n,s}^{j,1} = x_{n-1,s-1}^{j,1},\ x_{n,s}^{j,2} = x_{n-1,s-1}^{j,2}, \\ &x_{n,s}^{j,3} = x_{n-1,s-1}^{j,3},\ x_{n,s}^{j,4} = x_{n-1,s-1}^{j,4}, j = 2, 3, ..., N-1, \\ &x_{n,s}^{N,1} = x_{n-1,s-1}^{N,1}, \end{aligned}$$

for $2 \leqslant s \leqslant n \leqslant N_T$.

Proof. The proof is similar to that of Proposition 3. We only need to recompute $\mathbf{v}_{j,n}^{k}$ and $\mathbb{Y}_{n,q}^{j}$ for each transmission condition.

4. New algorithm for time independent linear potential

The standard implementation of the SWR method for the time-independent equations leads to the following classical algorithm

Algorithm 1: Classical algorithm

1: Initialize the iteration by g^0,
2: Solve Schrödinger on each subdomain with g^k.
3: Exchange values at interfaces and compute g^{k+1}.
4: Do again steps 2 and 3 until error $||g^{k+1} - g^k|| < \varepsilon$, $\varepsilon \ll 1$.

As we can see, the classical algorithm requires to solve K times the Schrödinger equation on each subdomain, where K corresponds to the number of iterations required to reach convergence. We are going to present a new algorithm for $\mathscr{V} = V(x)$ which is more efficient. As we will see, it will require to solve the Schrödinger equation on each subdomain only four times in total. This new

algorithm is equivalent to the classical algorithm, but it reduces significantly the calculations.

Before giving this new algorithm, we could see that the classical algorithm is based on (13): $g^{k+1} = \mathcal{R}g^k$, where the operator $\mathcal{R}$ includes the steps 2 and 3. We have shown in Proposition 2 that

$$g^{k+1} = \mathcal{R}g^k = \mathcal{L}g^k + d. \tag{25}$$

It is easy to see that (25) is nothing but the fix point method to solve the equation

$$(I - \mathcal{L})g = d. \tag{26}$$

A big advantage to interpret (25) as a fixed point method to solve(26) is that we can use any other iterative methods to solve this linear system. So we can use Krylov methods (ex. Gmres, Bicgstab) [16], which could accelerate the convergence prospectively. To use the Krylov methods or fixed point method, it is enough to define the application of $I - \mathcal{L}$ to vector g by

$$(I - \mathcal{L})g = I - \mathcal{R}g + d.$$

The classical algorithm could then be rewritten with

Algorithm 2: Classical algorithm, version 2

1: Build $d = \mathcal{R} \cdot \mathbf{0}$ in (26) explicitly,
2: Define the application of $I - \mathcal{L}$ to vector in (26),
3: Solve the linear system (26) by an iterative method (fixed point or Krylov).
4: Solve the Schrödinger equation on each subdomain for each time step using the boundary conditions obtained at step 3.

If the fixed point method is used in Step 3, we recover the first version of the classical algorithm. The second version of the classical algorithm allows the use of Krylov methods to accelerate convergence. However, applying $(I - \mathcal{L})$ to vector g is still a very expensive operation. With the help of Propositions 3 and 4, we propose a new algorithm

Algorithm 3: New algorithm

1: Build $\mathcal{L}$ and d in (26) explicitly,
2: Solve (26) by an iterative method,
3: Solve Schrödinger equation on each subdomain using the boundary conditions obtained at step 2.

We show beloa the construction of the matrix $\mathcal{L}$ and the vector d. As it will be seenn, their computation is not costly. Regarding the implementation, we then show how $\mathcal{L}$ and d are stored for use of parallelism. Here, we use the PETSc library [17]. Using the matrix form in PETSc, the memory required for each MPI process [18] is independent of the number of subdomains.

4.1. Construction of the matrix $\mathcal{L}$ and the vector d

We use the formulas (9) and (10) for the constructions. Numerically, we consider $l_{j,n}^k$ and $r_{j,n}^k$ as inputs, and $l_{j-1,n}^{k+1}$ and $r_{j+1,n}^{k+1}$ as outputs:

$$\text{inputs: } l_{j,n}^k, r_{j,n}^k \longrightarrow (10) \longrightarrow \text{outputs: } l_{j-1,n}^{k+1}, r_{j+1,n}^{k+1}.$$

It is easy to see that

$$d = \left(d_{1,r}^T, d_{2,l}^T, d_{2,r}^T, \cdots, d_{N,l}^T\right)^T = \mathcal{R} \cdot \mathbf{0},$$

where $\mathbf{0}$ is the zero vector. The elements of d are obtained by

$$d_{j-1,r} = \begin{pmatrix} r_{j-1,1}^{k+1} \\ r_{j-1,2}^{k+1} \\ \vdots \\ r_{j-1,N_T}^{k+1} \end{pmatrix}, \quad d_{j+1,l} = \begin{pmatrix} l_{j+1,1}^{k+1} \\ l_{j+1,2}^{k+1} \\ \vdots \\ l_{j+1,N_T}^{k+1} \end{pmatrix},$$

where the scalars $r_{j-1,s}^{k+1}, l_{j+1,s}^{k+1}, s = 1, 2, ..., N_T$ are given by the formula (10) with

$$l_{j,s}^k = r_{j,s}^k = 0, s = 1, 2, ..., N_T.$$

The equation is solved numerically on each subdomain only one time. Note that this construction works for $\mathscr{V} = V(t, x)$.

According to Propositions 4 and 3, if $\mathscr{V} = V(x)$, in order to build the matrix $\mathcal{L}$, it is enough to compute the first columns of blocks $X^{1,4}$, $X^{j,1}$, $X^{j,2}$, $X^{j,3}$, $X^{j,4}$, $j = 2, 3, ..., N-1$ and $X^{N,1}$.

The first column of $X^{j,1}$ is

$$X^{j,1} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \left(X^{j,1} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + X^{j,2} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + d_{j-1,r}\right) - d_{j-1,r} = \begin{pmatrix} r_{j-1,1}^{k+1} \\ r_{j-1,2}^{k+1} \\ \vdots \\ r_{j-1,N_T}^{k+1} \end{pmatrix} - d_{j-1,r}.$$

The first column of $X^{j,3}$ is

$$X^{j,3} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \left(X^{j,3} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + X^{j,4} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + d_{j+1,l}\right) - d_{j+1,l} = \begin{pmatrix} l_{j+1,1}^{k+1} \\ l_{j+1,2}^{k+1} \\ \vdots \\ l_{j+1,N_T}^{k+1} \end{pmatrix} - d_{j+1,l}.$$

The scalars $r_{j-1,s}^{k+1}, l_{j+1,s}^{k+1}, s = 1, 2, ..., N_T$ are computed by the formula (10) with

$$l_{j,s}^k = r_{j,s}^k = 0, s = 1, 2, ..., N_T \text{ except for } l_{j,1}^k = 1.$$

The equation is solved numerically only one time on the subdomain (a_j, b_j).

In the same way, the first columns of $X^{j,2}$ and $X^{j,4}$ are

$$X^{j,2}\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix} = \left(X^{j,2}\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix} + X^{j,4}\begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix} + d_{j-1,r}\right) - d_{j-1,r} = \begin{pmatrix} r_{j-1,1}^{k+1}\\ r_{j-1,2}^{k+1}\\ \vdots\\ r_{j-1,N_T}^{k+1}\end{pmatrix} - d_{j-1,r},$$

and

$$X^{j,4}\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix} = \left(X^{j,2}\begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix} + X^{j,4}\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix} + d_{j+1,l}\right) - d_{j+1,l} = \begin{pmatrix} l_{j+1,1}^{k+1}\\ l_{j+1,2}^{k+1}\\ \vdots\\ l_{j+1,N_T}^{k+1}\end{pmatrix} - d_{j+1,l},$$

where the scalars $r_{j-1,s}^{k+1}, l_{j+1,s}^{k+1}, s = 1, 2, ..., N_T$ are obtained by the formula (10), but with

$$l_{j,s}^{k} = r_{j,s}^{k} = 0, s = 1, 2, ..., N_T \text{ except for } r_{j,1}^{k} = 1.$$

The equation is solved numerically on each subdomain (a_j, b_j) only one time.

In conclusion, it is sufficient to solve the equation (2) on each subdomain three times to construct explicitly the interface problem. The construction is inexpensive. In total, the equation (2) is solved on each subdomain four times in the new algorithm. Numerically, we will compare the classical and the new algorithms in Section 6.1.

4.2. Storage of the matrix $\mathcal{L}$ and the vector d for massive parallel computing

Thanks to the peculiar form of the matrix $\mathcal{L}$, we can build it on parallel computers through an MPI implementation. The transpose of $\mathcal{L}$ is stored in a distributed manner using the library PETSc. As we can see below, the first block column of $\mathcal{L}$ is in MPI process 0. The second and third blocks columns are in MPI process 1, and so on for other processes. The consumed memory for each process is at most the sum of 4 blocks. The size of each block is $N_T \times N_T$. Each block contain $(N_T + 1) \times N_T/2$ non zero elements according to Propositions 2 and 1.

$$\mathcal{L} = \begin{pmatrix} \overbrace{}^{\text{MPI } 0} & \overbrace{}^{\text{MPI } 1} & \overbrace{}^{\text{MPI } 2} & & \overbrace{}^{\text{MPI } N-2} & \overbrace{}^{\text{MPI } N-1} \\ & X^{2,1} \quad X^{2,2} & & & & \\ X^{1,4} & & & & & \\ & & X^{3,1} \quad X^{3,2} & & & \\ & X^{2,3} \quad X^{2,4} & & & & \\ & & & \cdots & & \\ & & X^{3,3} \quad X^{3,4} & & & \\ & & & & X^{N-1,1} \quad X^{N-1,2} & \\ & & & \cdots & & \\ & & & & & X^{N,1} \\ & & & & X^{N-1,3} \quad X^{N-1,4} & \end{pmatrix}. \tag{27}$$

The vector d can also be stored in PETSc form. The first block is in MPI process 0, the second and the third are in MPI process 1, and so on. The last block is in MPI process $N-1$. Each MPI process contain at most $2 \times N_T$ elements.

$$d = \big(\underbrace{d_{1,r}^T}_{\text{MPI } 0}, \underbrace{d_{2,l}^T, d_{2,r}^T}_{\text{MPI } 1}, \cdots, \underbrace{d_{j,l}^T, d_{j,r}^T}_{\text{MPI } j-1}, \cdots, \underbrace{d_{N,l}^T}_{\text{MPI } N-1} \big)^T.$$

5. Preconditioned algorithm for general potentials

In Section 3, we have established the interface problem (13) for Schrödinger equation with time dependent or nonlinear potential. However, it is not possible to construct the interface matrix $\mathcal{L}$ without much computation since the Propositions 3 and 4 only hold for time independent linear potential. Thus, the new algorithm is not suitable here. Instead, to reduce the number of iterations required for convergence, we propose to add a preconditioner P^{-1} (P is a non singular matrix) in (13) which leads to the preconditioned algorithm:

1. for $\mathscr{V} = V(t,x)$,
$$g^{k+1} = I - P^{-1}(I - \mathcal{R})g^k, \tag{28}$$
$$P^{-1}(I - \mathcal{L}) = P^{-1}d, \tag{29}$$
2. for $\mathscr{V} = f(u)$,
$$g^{k+1} = I - P^{-1}(I - \mathcal{R}_{nl})g^k. \tag{30}$$

We now turn to explain which preconditioner is used. The interface problem for the free Schrödinger equation (without potential) is

$$g^{k+1} = \mathcal{L}_0 g^k + d,$$

where the symbol $\mathcal{L}_0$ is used to highlight here the potential is zero. The transmission condition is the same as that for (1). We propose for time dependent or nonlinear potential the preconditioner as

$$P = I - \mathcal{L}_0.$$

We have two reasons to believe that this is a good choice.

1. The matrix $\mathcal{L}_0$ can be constructed easily since a zero potential is independent of time. Therefore, the construction of $\mathcal{L}_0$ only needs to solve the free Schrödinger equation two times on each subdomains. This construction is therefore scalable.
2. Intuitively, the Schrödinger operator without potential is a roughly approximating of the Schrödinger operator with potential:

$$i\partial_t + \partial_{xx} \approx i\partial_t + \partial_{xx} u + \mathscr{V},$$

thus

$$P = I - \mathcal{L}_0 \approx I - \mathcal{L}, \quad P = I - \mathcal{L}_0 \approx I - (\mathcal{R}_{nl} - \mathcal{R}_{nl} \cdot \mathbf{0}).$$

Next, we present the application of preconditioner. The transpose of P is stored in PETSc form. For any vector y, the vector $x := P^{-1}y$ is computed by solving the linear system

$$Px = (I - \mathcal{L}_0)x = y \;\Leftrightarrow\; x^T P^T = y^T. \tag{31}$$

We do not explicitly construct the matrix P^{-1} as the inverse of a distributed matrix numerically is too expensive. The linear system (31) is solved by the Krylov methods (Gmres or Bicgstab) initialized by zero vector using the library PETSc. We will see in Section 6.3 that the computation time for applying this preconditioner is quite small compared with the computation time for solving the Schrödinger equation on subdomains.

6. Numerical results

The physical domain $(a_0, b_0) = (-21, 21)$ is decomposed into N equal subdomains without overlap. We fix in this section the final time to $T = 0.5$, the time step to $\Delta t = 0.001$ and the mesh size to $\Delta x = 10^{-5}$ without special statement. The potentials that we consider in this part and the corresponding initial data are

1. time independent linear potential: $\mathscr{V} = -x^2$, $u_0(x) = e^{-(x+10)^2+20i(x+10)}$,
2. time dependent linear potential: $\mathscr{V} = 5tx$, $u_0(x) = e^{-(x+10)^2+20i(x+10)}$,
3. nonlinear potential: $\mathscr{V} = |u|^2$, $u_0(x) = 2\mathrm{sech}\big(\sqrt{2}(x+10)\big)e^{20i(x+10)}$,

which give rise to solutions that propagates to the right side and undergoes dispersion. Since the matrices $\mathbb{M}_j$, $\mathbb{S}_j$ and $\mathbb{M}_{j,W_n}$ are both tri-diagonal symmetric in one dimension, the consumed memory is low. It is thus possible to solve numerically the Schrödinger equation on the entire domain $(0, T) \times (a_0, b_0)$ with a standard machine. The modulus of solutions at the final time $t = T$ are presented in Figure 2 for $\mathscr{V} = -x^2$ and $\mathscr{V} = |u|^2$.

We use a cluster consisting of 92 nodes (16 cores/node, Intel Sandy Bridge E5-2670, 32GB/node) to implement the SWR algorithms. We fix one MPI

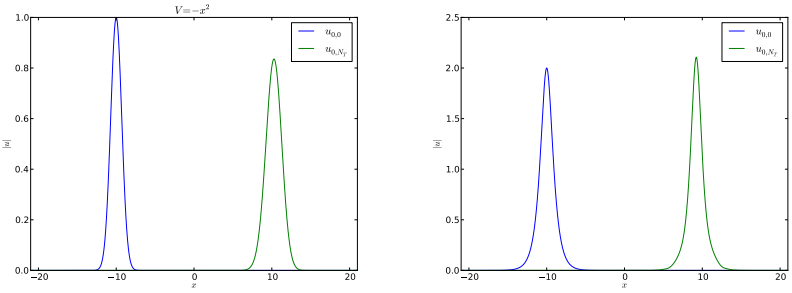

Figure 2: $|u_{0,0}|$ et $|u_{0,N_T}|$ on (a_0, b_0), $\mathscr{V} = -x^2$ (left) and $\mathscr{V} = |u|^2$ (right), $\Delta t = 0.001$, $\Delta x = 10^{-5}$.

process per subdomain and 16 MPI processes per node. The communications are handled by PETSc and Intel MPI. The linear systems (9) and (12) related to the Schrödinger equation are solved by the LU direct method using the MKL Pardiso library. The convergence condition for our SWR algorithm is $\parallel g^{k+1} - g^k \parallel < 10^{-10}$. Two types of initial vectors g^0 are considered in this article. One is the zero vector, another is the random vector. According to our tests, the zero initial vector makes the algorithms to converge faster, but obviously it could not include all the frequencies. As mentioned in [19], using the zero initial vector could give wrong conclusions associated with the convergence. Thus, the zero vector is used when one wants to evaluate the computation time, while the random vector is used when comparing the transmission conditions.

6.1. Comparison of classical and new algorithms

We are interested in this part to observe the robustness of the algorithms, to know whether they converge or not for the time independent potential $\mathscr{V} = -x^2$. Similarly, we will observe the computation time and the high scalability of the algorithms. We denote by T^{ref} the computation time required to solve numerically on a single processor the Schrödinger equation on the entire domain and T^{cls} (resp. T^{new}) the computation time of the classical (resp. new) algorithm for N subdomains. We test the algorithms for $N = 2, 10, 100, 500, 1000$ subdomains with the transmission condition S_0^2. The reason for using S_0^2 for these tests will be explained in Remark 1. The initial vector here is the zero vector.

First, the convergence history and the computation time for the algorithms are shown in Figure 3 and Table 1 where the fixed point method is used on the interface problem. The algorithms converge for 500 sub domains, but not for 1000 sub domains.

Next, we use the Krylov methods (Gmres or Bicgstab) on the interface problem instead of the fixed point method. Table 2 present the computation time. As we can see, the use of Krylov methods allows to obtain robust scalable SWR algorithms. The algorithms converge for 1000 subdomains and are scalable up to 500 subdomains. Besides their computation times are lower than the ones of

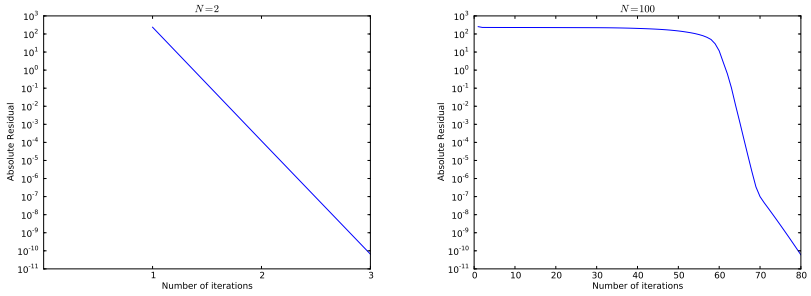

Figure 3: Convergence history, $N = 2, 100$, $\mathscr{V} = -x^2$, $\Delta t = 0.001$, $\Delta x = 10^{-5}$, Fixed point.

Table 1: Computation time in seconds, $\mathscr{V} = -x^2$, $\Delta t = 0.001$, $\Delta x = 10^{-5}$, Fixed point.

N	2	10	100	500
T^{ref}	403.56			
T^{cls}	773.07	2937.77	359.30	284.78
T^{new}	773.72	178.30	18.19	4.76

Table 2: Computation time in seconds, $\mathscr{V} = -x^2$, $\Delta t = 0.001$, $\Delta x = 10^{-5}$, Gmres and Bicgstab.

	N	2	10	100	500	1000
	T^{ref}	403.56				
Gmres	T^{cls}	771.82	2577.51	2249.54	907.06	739.65
	T^{new}	777.42	177.20	18.95	6.86	8.17
Bicgstab	T^{cls}	774.19	2760.11	679.72	799.09	845.65
	T^{new}	774.44	177.02	18.18	6.83	7.12

the classical algorithm. Roughly speaking, in Table 1 and Table 2 we have

$$T^{\text{cls}} = T_{\text{sub}} \times N_{\text{iter}} + ...,$$
$$T^{\text{new}} = T_{\text{sub}} \times 4 + T_{Ld} + ...,$$

where T_{sub} is the computation time for solving the equation on one subdomain, T_{Ld} is the computation time for solving the interface problem, "..." represent the negligible part of computation time such as the construction of matrices for the finite element method. If the number of subdomains N is not so large, then $T_{\text{sub}} \gg T_{Ld}$ and the minimum of N_{iter} is 3 in all our tests. If the number of subdomains N is large, then $T_{Ld} \sim T_{\text{sub}}$ and $N_{\text{iter}} \gg 4$. It is for this reason that the new algorithm takes less computation time. However, as the number of subdomains increase, T_{Ld} becomes larger. Thus, the new algorithm loses scalability if the number of subdomains is large.

In conclusion, the new algorithm with Krylov methods is robust and it takes much less computation time than the classical algorithm.

6.2. Comparison of classical and preconditioned algorithms

In this part, we are interested in observing the robustness, the computation time and the scalability of the preconditioned and non-preconditioned (classical) algorithms for time dependent potential $\mathscr{V} = 5tx$ and nonlinear potential $\mathscr{V} = |u|^2$. We denote by N_{pc} the number of iterations required to obtain convergence with the preconditioned algorithm and T_{pc} the computation time of the preconditioned algorithm. The transmission condition used in this section is S_0^2. We use the zero vector as the initial vector g_0.

First, we present in Figure 4 the convergence history for $\mathscr{V} = 5tx$. If N is not large, then there is no big difference between the classical algorithm and the preconditioned algorithm. However, if N is large, then as at each iteration, one subdomain communicate only with two adjacent subdomains, we can see that the non-preconditioned algorithm converges very slowly in the first interations. The convergence of the preconditioned algorithm improves greatly since the preconditioner allows communication with remote subdomains. The number of

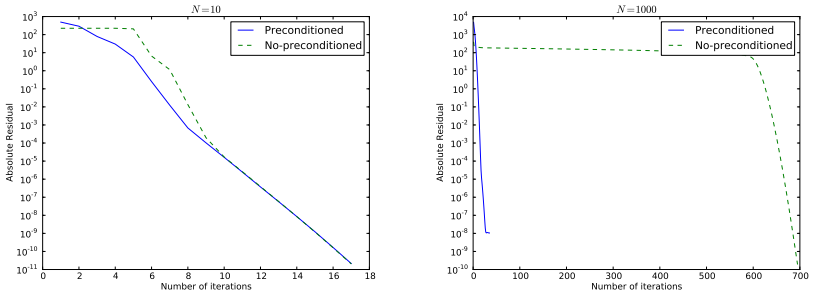

Figure 4: Convergence history, $N = 10, 1000$, $\mathscr{V} = 5tx$, $\Delta t = 0.001$, $\Delta x = 10^{-5}$.

iterations required for convergence and the computation time are presented in

Table 3 for $N = 10$, $N = 100$, $N = 500$ and $N = 1000$. We can see that the preconditioner allows to decrease significantly both the number of iterations and the computation time. The strong scalability of the classical algorithm is very low. Indeed, the number of iterations required increases with the number of subdomains. The preconditioned algorithm is much more scalable (up to 500 subdomains). However, it loses scalability from $N = 500$ to $N = 1000$. There are two reasons. One is that the number of iterations required for $N = 1000$ is a little bit more than that for $N = 500$. The other one is linked to the implementation of the preconditioner. Indeed, the time T_{pc} consists of three major parts: the application of $\mathcal{R}$ to vectors (step 1, denoted by T_1), the construction of the preconditioner (denoted by T_{3c}) and the application of preconditioner (step 3, denoted by T_3). We have thereby

$$T_{\text{pc}} \approx T_1 + T_{3c} + T_3. \tag{32}$$

If N is not very large, $T_1 \sim T_{3c} \gg T_3$. By increasing the number of subdomains, T_1 and T_{3c} decreases and T_3 increases. Thus, if N is large, T_3 is not negligible compared to T_1 and T_{3c}. However, it is not very convenient to estimate T_1 and T_3 in our codes because we use the "free-matrix" solvers in the PETSc library. To confirm our explanation, we make tests using a coarser mesh in space ($\Delta t = 0.001$, $\Delta x = 10^{-4}$). The size of the interface problem (13) is the same, thus T_3 should be similar to that of the previous tests ($\Delta t = 0.001$, $\Delta x = 10^{-5}$). But the size of the problem on a subdomain is ten times smaller. Thus, T_1 and T_{3c} are both smaller. The preconditioned algorithm should be less scalable. The results are shown in Table 4. It can be seen that the computation time T_{pc} for $N = 1000$ is larger than for $N = 500$ and the preconditioned algorithm is not very scalable from $N = 100$ to $N = 500$. Despite this remark, we could conclude from our tests that the preconditioned algorithm reduces a lot the number of iterations and the computing time compared to the classical algorithm.

Table 3: Number of iterations required and computation time of the classical algorithm and the preconditioned algorithm, $\mathscr{V} = 5tx$, $\Delta t = 0.001$, $\Delta x = 10^{-5}$.

N	10	100	500	1000
N_{nopc}	17	71	349	695
N_{pc}	17	32	31	35
T^{ref}	6496.3			
T_{nopc}	10123.1	3217.0	2466.5	2238.0
T_{pc}	10128.9	1432.7	250.0	170.7

Next, we reproduce the same tests for the nonlinear potential $\mathscr{V} = |u|^2$. The convergence history is presented in Figure 5. We show the number of iterations and the computation time in Table 5. The conclusions are quite similar.

6.3. Comparison of the transmission conditions

In this part, we compare the transmission conditions which are presented in Section 2 in the framework of the new algorithm for $\mathscr{V} = -x^2$ and the

Table 4: Number of iterations required and computation time of the classical algorithm and the preconditioned algorithm, $\mathscr{V} = 5tx$, $\Delta t = 0.001$, $\Delta x = 10^{-4}$.

N	10	100	500	1000
N_{nopc}	17	71	349	695
N_{pc}	17	32	26	25
T^{ref}	507.5			
T_{nopc}	681.9	223.8	210.2	191.2
T_{pc}	694.3	107.6	38.4	54.5

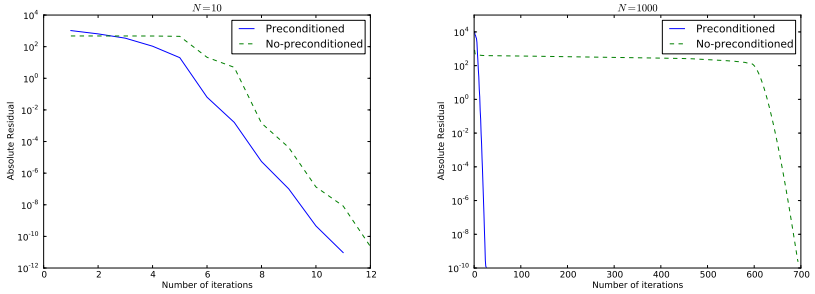

Figure 5: Convergence history, $N = 10, 1000$, $\mathscr{V} = |u|^2$, $\Delta t = 0.001$, $\Delta x = 10^{-5}$.

Table 5: Number of iterations required and computation time of the classical algorithm and the preconditioned algorithm, $\mathscr{V} = |u|^2$, $\Delta t = 0.001$, $\Delta x = 10^{-5}$.

N	10	100	500	1000
N_{nopc}	12	71	349	694
N_{pc}	11	22	25	26
T^{ref}	3200.8			
T_{nopc}	2582.3	1332.2	1248.0	1129.7
T_{pc}	2446.7	408.2	117.6	83.8

preconditioned algorithm for $\mathscr{V} = |u|^2$. The theoretical optimal parameter p in the transmission condition Robin being not at hand, we seek the best parameter numerically. We use in the subsection the random vector as the initial vector g_0 to make sure that all frequencies are present.

6.3.1. Case of linear potential

We first consider the linear potential $\mathscr{V} = -x^2$. We compare the number of iterations, the total computation time to perform a complete simulation and the computation time required (T_{Ld}) to solve the interface problem in Table 6 for $N = 2$ using the fixed point method, Gmres and Bicgstab methods on the interface problem. As can be seen, the total computation times are almost identical. The required computation time for solving the interface problem is relatively close to zero compared with the total computation time. Therefore, we are interested rather in the number of iterations. We can make the following observations

1. the number of iterations required for the Robin transmission condition is greater compared to the other three strategies,
2. in each strategy, the number of iterations is not sensitive to order,
3. for the Padé approximation strategy, the number of iterations decrease as the parameter of Padé (m) increase.

Table 6: Comparison of transmission conditions for $N = 2$, $V = -x^2$, $\Delta t = 10^{-3}$, $\Delta x = 10^{-5}$.

		Fixed point			Gmres			Bicgstab		
Strategy		N_{iter}	T_{Ld}	T_{total}	N_{iter}	T_{Ld}	T_{total}	N_{iter}	T_{Ld}	T_{total}
S_0^M	S_0^2	6	0.005	775.7	5	0.002	774.2	3	0.002	773.8
	S_0^3	6	0.002	774.2	5	0.002	779.6	3	0.002	773.3
	S_0^4	6	0.002	769.0	5	0.002	774.2	3	0.002	773.6
S_1^M	S_1^2	6	0.002	773.4	5	0.002	773.2	3	0.002	773.8
	S_1^4	6	0.002	773.9	5	0.002	773.6	3	0.002	774.5
S_2^M	$S_2^{2,20}$	191	0.062	773.3	28	0.010	774.5	16	0.011	773.1
	$S_2^{2,50}$	76	0.025	773.6	27	0.010	773.3	15	0.010	773.6
	$S_2^{2,100}$	39	0.013	776.3	23	0.008	775.2	13	0.009	773.6
	$S_2^{4,20}$	181	0.059	769.9	28	0.010	774.6	15	0.010	773.6
	$S_2^{4,50}$	77	0.025	776.0	27	0.010	773.5	15	0.010	773.3
	$S_2^{4,100}$	39	0.013	775.4	23	0.008	773.8	13	0.009	774.8
Robin*		1112	0.360	774.7	47	0.017	776.4	27	0.018	777.4

$*$ the parameters for the transmission condition Robin are $p = 44$ (fixed point), $p = 5$ (Gmres) and $p = 5$ (Bicgstab).

We make the same tests for $N = 500$, the results are shown in Table 7. We could see that

1. in each strategy, the number of iterations is not sensitive to order,
2. for the Padé approximation strategy, if the parameter m is small, then the algorithm is not robust,
3. the Krylov methods (Gmres and Bicgstab) could not always reduce the number of iterations.

Table 7: Comparison of transmission conditions for $N = 500$, $V = -x^2$, $\Delta t = 10^{-3}$, $\Delta x = 10^{-5}$.

		Fixed point			Gmres			Bicgstab		
Strategy		N_{iter}	T_{Ld}	T_{total}	N_{iter}	T_{Ld}	T_{total}	N_{iter}	T_{Ld}	T_{total}
S_0^M	S_0^2	357	0.775	4.68	1023	2.883	6.91	368	1.646	5.51
	S_0^3	337	0.734	4.62	977	2.620	6.55	345	1.831	5.77
	S_0^4	337	0.733	4.65	978	2.681	6.54	350	1.739	5.73
S_1^M	S_1^2	341	0.745	4.62	1010	2.364	6.20	353	2.102	6.00
	S_1^4	340	0.743	4.63	1023	3.454	7.19	351	2.225	6.06
S_2^M	$S_2^{2,20}$	-			1240	3.368	7.34	440	2.626	6.64
	$S_2^{2,50}$	-			997	2.320	6.30	352	2.240	6.16
	$S_2^{2,100}$	336	0.735	4.62	998	3.055	7.03	333	1.603	5.62
	$S_2^{4,20}$	-			1216	3.349	7.31	464	2.044	6.05
	$S_2^{4,50}$	-			1043	3.907	7.85	336	1.756	5.63
	$S_2^{4,100}$	336	0.733	4.60	1024	2.424	6.35	334	1.989	5.95
Robin*		1690	3.628	7.52	1060	3.000	6.80	318	1.41	5.32

$*$: the parameters for the transmission condition Robin are $p = 45$ (fixed point), $p = 19$ (Gmres) and $p = 6$ (Bicgstab).
-: the algorithm does not converge before 2000 iterations.

We could conclude that if the number of subdomains N is not very large, the potential strategy in order 2 with Bicgstab method on the interface problem is a good choice. If N is large, the Bicgstab method also allows most of the algorithms to converge, but it is difficult to have a general conclusion for the transmission conditions in the framework of new algorithm.

6.3.2. Case of nonlinear potential

Now we turn to compare the transmission conditions for the nonlinear potential $\mathscr{V} = |u|^2$ in the framework of the preconditioned algorithm. First, we study the influence of the parameter p in the Robin transmission condition. The number of iterations and the computation time are shown in Table 8. It is clear that the convergence is not sensitive to this parameter. Next we compare the three strategies. The numericals results are presented in Table 9. The transmission conditions S_0^4, S_1^4 and S_2^4 include the evaluation of $f(u)$. We don't find a suitable discretization of this term such that the continuity of $\mathbf{v}_j$ at the interfaces ensure the continuity of $\partial_{\mathbf{n}_j} f(u)$. Thus we could not obtain the solution $\mathbf{u}_{j,n}$ that satisfy $\mathbf{u}_{j,n} = R_j \mathbf{u}_{0,n}$. We could see that the number of iterations is not

Table 8: Influence of parameter p in the transmission conditions Robin, $N = 2, 10, 100$, $\mathscr{V} = |u|^2$, $\Delta t = 0.001$, $\Delta x = 10^{-4}$.

		$N = 2$		$N = 10$		$N = 100$	
	p	N_{iter}	T_{total}	N_{iter}	T_{total}	N_{iter}	T_{total}
Robin	5	9	1042.6	12	257.9	21	55.7
	10	8	920.3	11	230.9	22	50.6
	15	8	920.3	11	228.7	22	46.7
	20	8	914.5	11	226.1	22	43.7
	25	8	913.0	11	226.4	22	43.6
	30	8	919.2	11	227.6	22	43.9
	35	8	922.1	11	231.8	22	44.4
	40	8	922.8	12	250.2	22	45.0
	45	8	921.7	12	252.5	22	46.0
	50	8	928.3	12	253.3	22	46.7

sensitive to the transmission condition and its order. However the computation time for the Padé strategy is greater than other strategies. On each subdomain, the non linearity is approximated by a fixed point procedure (see formula (12)). This fixed point procedure converges more slowly using the Padé strategy than the other strategies. This observation is also found in [14]. In conclusion, in the nonlinear case, we also think that the potential strategy of order 2 (S_0^2) is a good choice.

Table 9: Comparison of transmission conditions for $N = 2, 10, 100$, $\mathscr{V} = |u|^2$, $\Delta t = 0.001$, $\Delta x = 10^{-4}$.

		$N = 2$		$N = 10$		$N = 100$	
		N_{iter}	T_{total}	N_{iter}	T_{total}	N_{iter}	T_{total}
S_0^M	S_0^2	8	909.5	11	229.1	22	40.6
	S_0^3	7	802.1	10	205.8	22	41.6
S_1^M	S_1^2	7	802.3	10	205.6	22	41.4
S_2^M	$S_2^{2,20}$	7	1732.5	10	572.0	22	128.6
	$S_2^{2,50}$	7	4042.9	10	1342.3	23	310.3
	$S_2^{2,100}$	7	7900.5	10	2640.0	22	576.0

Remark 1. As we indicated previously, we explain here our choice of transmission condition: the potential strategy of order 2 (S_0^2). Indeed, it seems reasonable to consider it since

1. the algorithm is robust and the computation time for S_0^2 is similar to others transmission conditions,

2. if N is not so large, it is one of the best choice,
3. the implementation of S_0^2 is much easier than other transmission conditions.

6.4. Gpu acceleration

If the number of subdomain N is not so large, then solving the Schrodinger equation on subdomains takes most of the computation time. We move these computations from Cpu to Gpu. In this subsection, we present the numerical experiments of Gpu acceleration. Two Gpu libraries of NVIDIA are used: CUSPARSE (tri-diagonal solver) and CUBLAS (BLAS operations). We use 8 Gpu Kepler K20, and compare the Cpu and Gpu results for $N = 2, 4, 8$. We use always 1 Gpu/MPI process. Gpu could accelerate a lot the computation as

Table 10: Cpu and Gpu computation time, Bicgstab, S_0^2, $\Delta t = 0.001$, $\Delta x = 10^{-5}$, $V = -x^2$.

N	2	4	8
T^{Cpu}	774.4	393.0	203.2
T^{Gpu}	27.90	16.13	12.54
$T^{\mathrm{Cpu}}/T^{\mathrm{Gpu}}$	18	24	16

shown in Table 10. However the algorithm on Gpu is not scalable. The reason is that the size of problem is not large enough for Gpu. Gpu waste some of its ability. We test a larger case only for Gpu: $\Delta t = 0.001$, $\Delta x = 10^{-6}$. The results are shown in Table 11.

Table 11: Gpu computation time, Bicgstab, S_0^2, $\Delta = 0.001$, $\Delta x = 5 \times 10^{-6}$, $V = -x^2$.

N	2	4	8
T^{Gpu}	51.95	28.21	16.30

Finally, we make the same tests for the nonlinear potential in the framework of the preconditioned algorithm. The results are presented in Table 12 and Table 13. The conclusion is similar.

Table 12: Cpu and Gpu computation time, $\Delta t = 0.01$, $\Delta x = 10^{-5}$, $V = |u|^2$.

N	2	4	8
T^{Cpu}	373.6	526.7	316.0
T^{Gpu}	73.9	40.1	34.0
$T^{\mathrm{Cpu}}/T^{\mathrm{Gpu}}$	5	13	9

Table 13: Gpu computation time, $\Delta t = 0.01$, $\Delta x = 5 \times 10^{-6}$, $V = |u|^2$.

N	2	4	8
T^{Gpu}	134.3	73.7	46.0

7. Conclusion

We proposed in this paper a new algorithm of the SWR method for the one dimensional Schrödinger equation with time independent linear potential and a preconditioned algorithm for general potentials. The algorithms for both cases are scalable and could reduce significantly the computation time. Some newly constructed absorbing boundary conditions are used as the transmission condition and compared numerically in the framework of the algorithms proposed by us. We believe that the potential strategy of order 2 is a good choice. Besides, we adapted the codes developed on Cpu to Gpu. According to the experiments, the computation could be accelerated obviously.

Acknowledgements

We acknowledge Pierre Kestener (Maison de la Simulation Saclay France) for the discussions about the parallel programming, especially for his help about Gpu acceleration. This work was partially supported by the French ANR grant ANR-12-MONU-0007-02 BECASIM (Modèles Numériques call).

References

References

[1] L. Halpern, J. Szeftel, Optimized and quasi-optimal Schwarz waveform relaxation for the one dimensional Schrödinger equation, Math. Model. Methods Appl. Sci. 20 (12) (2010) 2167–2199.

[2] L. Halpern, J. Szeftel, Optimized and quasi-optimal Schwarz waveform relaxation for the one-dimensional Schrödinger equation, Tech. rep., CNRS (2006).

[3] F. Caetano, M. J. Gander, L. Halpern, J. Szeftel, Schwarz waveform relaxation algorithms for semilinear reaction-diffusion equations, Networks Heterog. Media 5 (3) (2010) 487–505.

[4] M. J. Gander, L. Halpern, Optimized Schwarz Waveform Relaxation Methods for Advection Reaction Diffusion Problems, SIAM J. Numer. Anal. 45 (2) (2007) 666–697.

[5] T. Hoang, J. Jaffré, C. Japhet, M. Kern, J. Roberts, Space-Time Domain Decomposition Methods for Diffusion Problems in Mixed Formulations 51 (6) (2013) 3532–3559.

[6] M. J. Gander, L. Halpern, F. Nataf, Optimal Schwarz waveform relaxation for the one dimensional wave equation, SIAM J. Numer. Anal. 41 (5) (2003) 1643–1681.

[7] L. Halpern, J. Szeftel, Nonlinear nonoverlapping Schwarz waveform relaxation for semilinear wave propagation, Math. Comput. 78 (266) (2009) 865–889.

[8] V. Dolean, M. J. Gander, L. Gerardo-Giorda, Optimized Schwarz Methods for Maxwell's Equations, SIAM J. Sci. Comput. 31 (3) (2009) 2193–2213.

[9] X. Antoine, E. Lorin, A. D. Bandrauk, Domain Decomposition Methods and High-Order Absorbing Boundary Conditions for the Numerical Simulation of the Time Dependent Schrödinger Equation with Ionization and Recombination by Intense Electric Field.

[10] X. Antoine, C. Besse, S. Descombes, Artificial boundary conditions for one-dimensional cubic nonlinear Schrödinger equations, SIAM J. Numer. Anal. 43 (6) (2006) 2272–2293.

[11] X. Antoine, C. Besse, P. Klein, Absorbing boundary conditions for the one-dimensional Schrödinger equation with an exterior repulsive potential, J. Comput. Phys. 228 (2) (2009) 312–335.

[12] X. Antoine, C. Besse, P. Klein, Absorbing Boundary Conditions for General Nonlinear Schrödinger Equations, SIAM J. Sci. Comput. 33 (2) (2011) 1008–1033.

[13] X. Antoine, C. Besse, J. Szeftel, Towards accurate artificial boundary conditions for nonlinear PDEs through examples, Cubo, A Math. J. 11 (4) (2009) 29–48.

[14] P. Klein, Construction et analyse de conditions aux limites artificielles pour des équations de Schrödinger avec potentiels et non linéarités, Ph.D. thesis, Université Henri Poincaré, Nancy 1 (2010).

[15] A. Durán, J. Sanz-Serna, The numerical integration of relative equilibrium solutions. The nonlinear Schrodinger equation, IMA J. Numer. Anal. 20 (2) (2000) 235–261.

[16] Y. Saad, Iterative methods for sparse linear systems, 2nd Edition, Society for Industrial and Applied Mathematics, 2003.

[17] S. Balay, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, H. Zhang, PETSc Users Manual, Tech. Rep. ANL-95/11 - Revision 3.4, Argonne National Laboratory (2013).

[18] Message Passing Interface Forum, MPI : A Message-Passing Interface Standard Version 3.0, Tech. rep. (2012).

[19] M. J. Gander, Schwarz methods over the course of time, Electron. Trans. Numer. Anal. 31 (2008) 228–255.