DOMAIN DECOMPOSITION ALGORITHMS FOR TWO DIMENSIONAL LINEAR SCHRÖDINGER EQUATION

Christophe Besse, Feng Xing

DOMAIN DECOMPOSITION ALGORITHMS FOR TWO DIMENSIONAL LINEAR SCHRÖDINGER EQUATION

CHRISTOPHE BESSE † AND FENG XING ‡ §

Abstract. This paper deals with two domain decomposition methods for two dimensional linear Schrödinger equation, the Schwarz waveform relaxation method and the domain decomposition in space method. After presenting the classical algorithms, we propose a new algorithm for the free Schrödinger equation and a preconditioned algorithm for the general Schrödinger equation. These algorithms are studied numerically, which shows that the two new algorithms could accelerate the convergence and reduce the computation time. Besides the traditional Robin transmission condition, we also propose to use a newly constructed absorbing condition as the transmission condition.

Key words. Schrödinger equation, Schwarz waveform relaxation method, domain decomposition in space method

AMS subject classifications. 35Q55, 65M55, 65Y05, 65M60

1. Introduction. The aim of this paper is to apply domain decomposition algorithms to two dimensional linear Schrödinger equation defined on $(0,T)\times\Omega$ with a real potential $V(t,x,y)$

$$\begin{cases} \mathscr{L}u := (i\partial_t + \Delta + V)u = 0, & (t,x,y)\in(0,T)\times\Omega, \\ u(0,x,y) = u_0(x,y), & (x,y)\in\Omega, \end{cases} \tag{1.1}$$

where $\Omega = (x_l, x_r)\times(y_d, y_u)$, $x_l, x_r, y_d, y_u \in \mathbb{R}$ is a bounded spatial domain and the initial datum $u_0 \in L^2(\Omega)$. The equation is complemented with the following boundary conditions:

$$\partial_{\mathbf{n}} u = 0,\ y = y_d, y_u, \quad \partial_{\mathbf{n}} u + S_b u = 0,\ x = x_l, x_r,$$

where $\partial_{\mathbf{n}}$ denotes the normal directive, $\mathbf{n}$ being the outwardly unit vector on $\partial\Omega$, and the operator S_b is some transmission operator.

We consider in this paper two domain decomposition methods. One is the Schwarz waveform relaxation method without overlap (SWR) [12, 15, 13], which is based on the time-space domain decomposition. The time-space domain $(0,T)\times\Omega$ is decomposed into some subdomains $(0,T)\times\Omega_j$, $j = 1,2,...,N$. The solution is computed on each subdomain and the time-space boundary values are transmitted via transmission conditions. The derivation of efficient transmission conditions is one of the key points of the SWR method. For Schrödinger equation, some transmission conditions are proposed in [14, 4, 7], such as Robin transmission condition, optimal transmission condition etc..

Another method is the domain decomposition in space method (DDS) [9, 18]. The time dependent equation is first discretized in time with an implicit scheme on the entire spatial domain, which leads to a stationary equation on space. Then the domain decomposition methods (such as the optimized Schwarz method [10, 8, 16])

†Institut de Mathématiques de Toulouse UMR5219, Université de Toulouse; CNRS, UPS IMT, F-31062 Toulouse Cedex 9, France. (christophe.besse@math.univ-toulouse.fr)

‡Maison de la Simulation, CEA Saclay France & Laboratoire Paul Painlevé, Université Lille Nord de France. (feng.xing@unice.fr).

§Current address: Laboratoire J.A. Dieudonné, Université de Nice & Inria Sophia Antipolis, France.

are applied to this stationary equation. The DDS method requires a conforming time discretization. However, it does not cause any problem for us since there is no potential needs to use a nonconforming discretization for the Schrödinger equation, even for the Gross-Pitaevskii equation [6, 1], which is one model for Bose–Einstein condensate.

This paper is organized as follows. In Section 2, we present in detail the classical SWR algorithm and the classical DDS algorithm. The classical DDS algorithm is interpreted as a combination of some classical SWR algorithms. The discretization of the Schrödinger equation and the transmission conditions are also provided. The discrete problem is written globally in time and the global discrete form of the transmission condition is given. In Section 3, we construct the interface problem and analyze its properties. Based on these properties, we propose new algorithms and preconditioned algorithms in the two following sections. In Section 6, we study numerically the performances of these algorithms. A conclusion is drawn in the last section.

2. Domain decomposition algorithms.

2.1. Geometric configuration. The interval (x_l, x_r) is divided into N subintervals (a_j, b_j) without overlap. The points a_j and b_j denote the ends of the subintervals (a_j, b_j). Thus, the entire domain Ω is decomposed into N non overlapping subdomains $\Omega_j = (a_j, b_j) \times (y_d, y_u)$, $j = 1, 2, ..., N$ (see Figure 1 for $N = 3$). We denote the normal directive on subdomain Ω_j by $\partial_{\mathbf{n}_j}$.

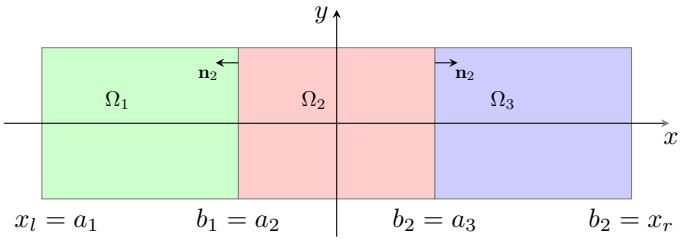

FIG. 1. *Geometric configuration.*

There are obviously other ways to decompose the entire domain. One way is illustrated in Figure 2 (left) for $N = 4$. The intervals (x_l, x_r) and (y_d, y_u) are decomposed into subintervals simultaneously in both spatial directions. In this configuration, an artificial cross point appears. It is well known that the domain decomposition method with cross points is a difficult problem in mathematics since the problem becomes singular at this point. Another possibility is illustrated in Figure 2 (right) for $N = 3$. The entire domain is decomposed into a circle and some rings. This approach has many disadvantages for parallel computing. The sizes of the interfaces are very different, especially if N is large. It is also difficult to control the sizes if the subdomains are similar. Thus, we restrict ourselves in this paper to the first description (see Figure 1).

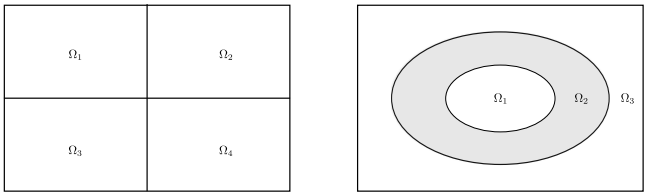

FIG. 2. *Two other ways of domain decomposition.*

2.2. Classical SWR algorithm. The classical SWR algorithm is given by

$$\begin{cases} \mathscr{L} u_j^k = 0, & (t,x,y) \in (0,T) \times \Omega_j, \\ u_j^k(0,x,y) = u_0(x,y), & (x,y) \in \Omega_j, \\ B_j u_j^k = B_j u_{j-1}^{k-1}, & x = a_j, \\ B_j u_j^k = B_j u_{j+1}^{k-1}, & x = b_j, \\ \partial_{\mathbf{n}_j} u_j^k = 0, & y = y_l, y_d, \end{cases} \tag{2.1}$$

with a special treatment for the two extreme subdomains $(0,T) \times \Omega_1$ and $(0,T) \times \Omega_N$ since the boundary conditions are imposed on $x = a_1$ and $x = b_N$

$$B_1 u_1^k = 0,\ x = a_1, \quad B_N u_N^k = 0,\ x = b_N.$$

The notation u_j^k denotes the solution on subdomain $(0,T) \times \Omega_j$ at iteration $k = 1, 2, ...$ of the algorithm. The boundary condition at interface nodes a_j and b_j is given in term of operator B_j defined by

$$B_j = \partial_{\mathbf{n}_j} + S_j,\ j = 1, ..., N, \tag{2.2}$$

where S_j is the transmission operator. Besides the classical widely used Robin transmission condition

$$S_j = -ip,\ p \in \mathbb{R}^+,\ j = 1, 2, ..., N, \tag{2.3}$$

we use in this paper a newly constructed absorbing boundary condition S_{2p}^m [2, 3] as the transmission condition

$$S_j = -i\sqrt{i\partial_t + \Delta_{\Gamma_j} + V},\ j = 1, 2, ..., N, \tag{2.4}$$

where $\Gamma_1 = \{b_1\} \times (y_d, y_u)$, $\Gamma_j = \{a_j, b_j\} \times (y_d, y_u), j = 2, 3, ..., N-1$ and $\Gamma_N = \{a_N\} \times (y_d, y_u)$.

In our case, the Laplace–Beltrami operator Δ_{Γ_j} is ∂_y^2. This absorbing boundary condition is constructed by using some pseudo differential techniques. Numerically, this operator is approximated by Padé approximation where m denotes its order. If the potential $V = 0$ and the spatial domain is $\mathbb{R}^2$, then it could be proven by using Fourier transform in time t and space y that the transmission condition S_{2p}^m is optimal. However, the optimal transmission condition is not always available. That's one of the reasons for using the absorbing operator. To avoid the theoretical problem in continues case, we consider only the discrete version of the transmission operators in this paper.

The classical SWR algorithm is initialized by an initial guess of $B_j u_j^0|_{x=a_j,b_j}$, $j = 1, 2, ..., N$. As shown in (2.5) for $N = 3$, at iteration k, the Schrödinger equation

is solved on each subdomain $(0,T)\times\Omega_j$ (*Solve*) to compute u_j^k and the boundary values are communicated (*Comm.*) via transmission conditions.

(2.5)
$$\cdots\rightarrow\begin{pmatrix}B_1u_1^k|_{x=b_1}\\B_2u_2^k|_{x=a_2}\\B_2u_2^k|_{x=b_2}\\B_3u_3^k|_{x=a_3}\end{pmatrix}\xrightarrow{Solve}\begin{pmatrix}u_1^k\\u_2^k\\u_3^k\end{pmatrix}\rightarrow\begin{pmatrix}B_2u_1^k|_{x=b_1}\\B_1u_2^k|_{x=a_2}\\B_3u_2^k|_{x=b_2}\\B_2u_3^k|_{x=a_3}\end{pmatrix}\xrightarrow{Comm.}\begin{pmatrix}B_1u_1^{k+1}|_{x=b_1}\\B_2u_2^{k+1}|_{x=a_2}\\B_2u_2^{k+1}|_{x=b_2}\\B_3u_3^{k+1}|_{x=a_3}\end{pmatrix}\rightarrow\cdots$$

Let us define the flux at iteration k by

$$g^k=(B_1u_1^k|_{x=b_1},\cdots,B_ju_j^k|_{x=a_j},B_ju_j^k|_{x=b_j},\cdots,B_Nu_N^k|_{x=a_N})^T,$$

where "$.^T$" is the standard notation of the transpose of a matrix or a vector. Thus, the algorithm could be written as

$$g^{k+1}=\mathcal{R}_cg^k, \tag{2.6}$$

where $\mathcal{R}_c$ is a linear operator. Thus, the continuous interface problem is

$$(I-\mathcal{R}_c)g=0, \tag{2.7}$$

where I is identity operator and $g=\lim_{k\to\infty}g^k$.

We now turn to the discretization of (2.1) and (2.6). The time interval $(0,T)$ is discretized uniformly with N_T intervals, $\Delta t=T/N_T$ is the time step. A semi-discrete approximation for the linear Schrödinger equation on $(0,T)\times\Omega_j$ is given by the Crank-Nicolson scheme

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

where $V_n=V(t_n,x,y)$, $t_n=n\Delta t$. It is useful to introduce new variables $v_{j,n}^k=(u_{j,n}^k+u_{j,n-1}^k)/2$ with $v_{j,0}^k=u_0$ and $W_n=(V_n+V_{n-1})/2$. The scheme could be written as

$$\frac{2i}{\Delta t}v_{j,n}^k+\Delta v_{j,n}^k+W_nv_{j,n}^k=\frac{2i}{\Delta t}u_{j,n-1}^k. \tag{2.8}$$

The semi-discrete transmission condition is given by

$$\overline{B}_jv_{j,n}^k=\overline{B}_jv_{j-1,n}^{k-1},\ x=a_j,\quad \overline{B}_jv_{j,n}^k=\overline{B}_jv_{j+1,n}^{k-1},\ x=b_j,$$

where $\overline{B}_j=\partial_{\mathbf{n}_j}+\overline{S}_j$, and $\overline{S}_j$ is the semi-discrete form of S_j

$$\text{Robin}:\quad \overline{S}_jv_{j,n}^k=-ip\cdot v_{j,n}^k,\ j=1,...,N, \tag{2.9}$$

$$S_{2p}^m:\quad \overline{S}_jv_{j,n}^k=-i\sum_{s=0}^m a_s^mv_{j,n}^k+i\sum_{s=1}^m a_s^md_s^m\varphi_{j,s}^{n-1/2},\ j=1,...,N, \tag{2.10}$$

where $a_s^m=e^{i\theta/2}/(m\cos^2(\frac{(2s-1)\pi}{4m}))$, $d_s^m=e^{i\theta}\tan^2(\frac{(2s-1)\pi}{4m})$, $s=0,2,...,m$, $\theta=\frac{\pi}{4}$. For the transmission condition S_{2p}^m, the auxiliary functions $\varphi_{j,s}^{n-1/2}$, $s=1,2,...,m$ are defined as the solutions of the following set of equations

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

Let us introduce the semi-discrete fluxes $l_{j,n}^k$ and $r_{j,n}^k$, $j = 1, 2, ..., N$ by

$$l_{j,n}^k(y) = \partial_{\mathbf{n}_j} v_{j,n}^k(a_j, y) + \overline{S}_j v_{j,n}^k(a_j, y),\ y \in (y_d, y_u),$$
$$r_{j,n}^k(y) = \partial_{\mathbf{n}_j} v_{j,n}^k(b_j, y) + \overline{S}_j v_{j,n}^k(b_j, y),\ y \in (y_d, y_u),$$

with two special cases $l_{1,n}^k = r_{N,n}^k = 0$. It is easy to see that on each subdomain, the boundary conditions are

$$\partial_{\mathbf{n}_j} v_{j,n}^k + \overline{S}_j v_{j,n}^k = l_{j,n}^k,\ x = a_j, \quad \partial_{\mathbf{n}_j} v_{j,n}^k + \overline{S}_j v_{j,n}^k = r_{j,n}^k,\ x = b_j.$$

Besides, the semi-discrete transmission conditions could be written as

$$\left\{ \begin{array}{l} l_{j,n}^k(y) = -r_{j-1,n}^{k-1}(y) + 2\overline{S}_j v_{j-1,n}^{k-1}(b_{j-1}, y),\ j = 2, 3, ..., N, \\ r_{j,n}^k(y) = -l_{j+1,n}^{k-1}(y) + 2\overline{S}_j v_{j+1,n}^{k-1}(a_{j+1}, y),\ j = 1, 2, ..., N-1. \end{array} \right. \tag{2.11}$$

The spatial approximation is realized by the standard $\mathbb{Q}_1$ finite element method. The mesh size of a discrete element is $(\Delta x, \Delta y)$. We denote by N_x (resp. N_y) the number of nodes in x (resp. y) direction on each subdomain. Let us denote by $\mathbf{v}_{j,n}^k$ (resp. $\mathbf{u}_{j,n}^k$) the nodal interpolation vector of $v_{j,n}^k$ (resp. $u_{j,n}^k$), $\mathbf{l}_{j,n}^k$ (resp. $\mathbf{r}_{j,n}^k$) the nodal interpolation vector of $l_{j,n}^k$ (resp. $r_{j,n}^k$), $\mathbb{M}_j$ the mass matrix, $\mathbb{S}_j$ the stiffness matrix and $\mathbb{M}_{j,W_n}$ the generalized mass matrix with respect to $\int_{\Omega_j} W_n v\phi dx$. Let $\mathbb{M}^{\Gamma_j}$ the boundary mass matrix, $\mathbb{S}^{\Gamma_j}$ the boundary stiffness matrix and $\mathbb{M}_{W_n}^{\Gamma_j}$ the generalized boundary mass matrix with respect to $\int_{\Gamma_j} W_n v\phi d\Gamma$. We denote by $Q_{j,l}$ (resp. $Q_{j,r}$) the restriction operators (matrix) from Ω_j to $\{a_j\} \times (y_d, y_u)$ (resp. $\{b_j\} \times (y_d, y_u)$) and $Q_j^T = (Q_{j,l}^T, Q_{j,r}^T)$. The matrix formulation for the transmission condition Robin is therefore given by

$$\text{Robin}: \quad \Big(\mathbb{A}_{j,n} + ip \cdot \mathbb{M}^{\Gamma_j}\Big)\mathbf{v}_{j,n}^k = \frac{2i}{\Delta t}\mathbb{M}_j \mathbf{u}_{j,n-1}^k - \mathbb{M}^{\Gamma_j} Q_j^T \begin{pmatrix} \mathbf{l}_{j,n}^k \\ \mathbf{r}_{j,n}^k \end{pmatrix}, \tag{2.12}$$

where $\mathbb{A}_{j,n} = \frac{2i}{\Delta t}\mathbb{M}_j - \mathbb{S}_j + \mathbb{M}_{j,W_n}$. The size of this linear system is $N_x \times N_y$. If we consider the transmission condition S_{2p}^m, we have

(2.13)

$$S_{2p}^m: \quad \left\{ \begin{array}{l} \Big(\mathbb{A}_{j,n} + i(\sum_{s=0}^m a_s^m) \cdot \mathbb{M}^{\Gamma_j}\Big)\mathbf{v}_{j,n}^k = \frac{2i}{\Delta t}\mathbb{M}_j \mathbf{u}_{j,n-1}^k \\ \qquad\qquad + i\sum_{s=1}^m a_s^m d_s^m \mathbb{M}^{\Gamma_j} Q_j^T \boldsymbol{\varphi}_{j,s}^{n-1/2} - \mathbb{M}^{\Gamma_j} Q_j^T \begin{pmatrix} \mathbf{l}_{j,n}^k \\ \mathbf{r}_{j,n}^k \end{pmatrix}, \\ -Q_j \mathbb{M}^{\Gamma_j} \mathbf{v}_{j,n}^k + Q_j(\frac{2i}{\Delta t}\mathbb{M}^{\Gamma_j} - \mathbb{S}^{\Gamma_j} + \mathbb{M}_{W_n}^{\Gamma_j} + d_s^m \mathbb{M}^{\Gamma_j}) Q_j^T \boldsymbol{\varphi}_{j,s}^{n-1/2} \\ \qquad\qquad = \frac{2i}{\Delta t} Q_j \mathbb{M}^{\Gamma_j} Q_j^T \boldsymbol{\varphi}_{j,s}^{n-1},\ \text{for } 1 \leqslant s \leqslant m, \\ \boldsymbol{\varphi}_{j,s}^0 = \mathbf{0} \text{ for } 1 \leqslant s \leqslant m. \end{array} \right.$$

It is a linear system with unknown $(\mathbf{v}_{j,n}^k, \boldsymbol{\varphi}_{j,1}^{n-1/2}, ..., \boldsymbol{\varphi}_{j,m}^{n-1/2})$ where $\boldsymbol{\varphi}_{j,s}^{n-1/2}$ is the nodal interpolation of $\varphi_{j,s}^{n-1/2}$ on the boundary. Considering (2.11), the discrete form

of SWR algorithm is given by

$$\text{(2.14)}\qquad \text{Robin}: \quad \begin{cases} \mathbf{r}_{j-1,n}^{k+1} = -\mathbf{l}_{j,n}^{k} + 2Q_{j,l}\cdot\widetilde{S}_j\mathbf{v}_{j,n}^{k},\ j=2,3,...,N, \\ \mathbf{l}_{j+1,n}^{k+1} = -\mathbf{r}_{j,n}^{k} + 2Q_{j,r}\cdot\widetilde{S}_j\mathbf{v}_{j,n}^{k},\ j=1,2,...,N-1, \\ \widetilde{S}_j\mathbf{v}_{j,n}^{k} = -ip\cdot\mathbf{v}_{j,n}^{k}, \\ \left(\mathbb{A}_{j,n} + ip\cdot\mathbb{M}^{\Gamma_j}\right)\mathbf{v}_{j,n}^{k} = \dfrac{2i}{\Delta t}\mathbb{M}_j\mathbf{u}_{j,n-1}^{k} - \mathbb{M}^{\Gamma_j}Q_j^T\begin{pmatrix}\mathbf{l}_{j,n}^{k}\\ \mathbf{r}_{j,n}^{k}\end{pmatrix}, \end{cases}$$

$$\text{(2.15)}\qquad S_{2p}^{m}: \quad \begin{cases} \mathbf{r}_{j-1,n}^{k+1} = -\mathbf{l}_{j,n}^{k} + 2Q_{j,l}\cdot\widetilde{S}_j\mathbf{v}_{j,n}^{k},\ j=2,3,...,N, \\ \mathbf{l}_{j+1,n}^{k+1} = -\mathbf{r}_{j,n}^{k} + 2Q_{j,r}\cdot\widetilde{S}_j\mathbf{v}_{j,n}^{k},\ j=1,2,...,N-1, \\ \widetilde{S}_j\mathbf{v}_{j,n}^{k} = -i(\sum_{s=0}^{m}a_s^m)\mathbf{v}_{j,n}^{k} + i\sum_{s=1}^{m}a_s^m d_s^m\boldsymbol{\varphi}_{j,s}^{n-1/2}, \\ \text{Equation (2.13).} \end{cases}$$

The equations (2.12) and (2.13) can be written globally in time. The two following propositions show that the difference between the two transmission conditions appears only on the definition of a matrix. We first define the discrete interface vectors

$$\begin{aligned} \mathbf{g}_1^k &= (\mathbf{r}_{1,1}^{k,T}, \mathbf{r}_{1,2}^{k,T}, \cdots, \mathbf{r}_{1,N_T}^{k,T})^T \in \mathbb{C}^{N_y\times N_T}, \\ \mathbf{g}_N^k &= (\mathbf{l}_{N,1}^{k,T}, \mathbf{l}_{N,2}^{k,T}, \cdots, \mathbf{l}_{N,N_T}^{k,T})^T \in \mathbb{C}^{N_y\times N_T}, \\ \mathbf{g}_j^k &= (\mathbf{l}_{j,1}^{k,T}, \cdots, \mathbf{l}_{j,N_T}^{k,T}, \mathbf{r}_{j,1}^{k,T}, \cdots, \mathbf{r}_{j,N_T}^{k,T})^T \in \mathbb{C}^{2N_y\times N_T},\ j=2,3,...N-1. \end{aligned}$$

It is not hard to verify the following proposition by evaluating the block matrix-vector product for the Robin transmission condition.

PROPOSITION 2.1. *For the Robin transmission condition, the global form in time of the equation* (2.12) *is*

$$\text{(2.16)}\qquad (\mathbf{A}_j - \mathbf{B}_j)\mathbf{v}_j^k = \mathbf{F}_j - \mathbf{M}^{\Gamma_j}\mathbf{Q}_j^T\mathbf{g}_j^k,\ j=1,2,...,N,$$

where $\mathbf{B}_j = -ip\cdot\mathbf{M}^{\Gamma_j} = -ip\cdot\mathrm{diag}_{N_T}\{\mathbb{M}^{\Gamma_j}\}$, $\mathbf{Q}_{j,l}^T = \mathrm{diag}_{N_T}\{Q_{j,l}^T\}$, $\mathbf{Q}_{j,r}^T = \mathrm{diag}_{N_T}\{Q_{j,r}^T\}$ *and*

$$\mathbf{A}_j = \begin{pmatrix} \mathbb{A}_{j,1} & & & & \\ -\frac{4i}{\Delta t}\mathbb{M}_j & \mathbb{A}_{j,2} & & & \\ \frac{4i}{\Delta t}\mathbb{M}_j & -\frac{4i}{\Delta t}\mathbb{M}_j & \mathbb{A}_{j,3} & & \\ \vdots & \vdots & & \ddots & \\ & & & -\frac{4i}{\Delta t}\mathbb{M}_j & \mathbb{A}_{j,N_T} \end{pmatrix},\ \mathbf{v}_j^k = \begin{pmatrix} \mathbf{v}_{j,1}^k \\ \mathbf{v}_{j,2}^k \\ \vdots \\ \vdots \\ \mathbf{v}_{j,N_T}^k \end{pmatrix},$$

$$\mathbf{F}_j = \frac{2i}{\Delta t}\begin{pmatrix} \mathbb{M}_j\mathbf{u}_{j,0}^k \\ -\mathbb{M}_j\mathbf{u}_{j,0}^k \\ \vdots \\ (-1)^{N_T-1}\mathbb{M}_j\mathbf{u}_{j,0}^k \end{pmatrix},\ \mathbf{Q}_j^T = \begin{pmatrix} Q_{j,l}^T & & & Q_{j,r}^T & & \\ & Q_{j,l}^T & & & Q_{j,r}^T & \\ & & \ddots & & & \ddots \\ & & & Q_{j,l}^T & & & Q_{j,r}^T \end{pmatrix}.$$

PROPOSITION 2.2. *If we consider the transmission condition* S_{2p}^m*, then the equation* (2.13) *could be written globally in time as* (2.16)

$$(\mathbf{A}_j - \mathbf{B}_j)\mathbf{v}_j^k = \mathbf{F}_j - \mathbf{M}^{\Gamma_j}\mathbf{Q}_j^T\mathbf{g}_j^k,\ j=1,2,...,N,$$

but with a different definition of $\mathbf{B}_j$*, given by*

(2.17)
$$\mathbf{B}_j = -\begin{pmatrix} c_a\mathbb{M}^{\Gamma_j}+\mathbb{Y}_j^{1,1} & & & & \\ \mathbb{Y}_j^{2,1} & c_a\mathbb{M}^{\Gamma_j}+\mathbb{Y}_j^{2,2} & & & \\ \mathbb{Y}_j^{3,1} & \mathbb{Y}_j^{3,1} & c_a\mathbb{M}^{\Gamma_j}+\mathbb{Y}_j^{3,3} & & \\ \vdots & \vdots & & \ddots & \\ \mathbb{Y}_j^{N_T,1} & \mathbb{Y}_j^{N_T,2} & \mathbb{Y}_j^{N_T,3} & \cdots & c_a\mathbb{M}^{\Gamma_j}+\mathbb{Y}_j^{N_T,N_T} \end{pmatrix},$$

where $c_a = i(\sum_{s=0}^m a_s^m)$.

Proof. According to (2.13), for $s = 1, 2, ..., m$, $n = 1, 2, ..., N_T$, we have

$$\begin{aligned} &-Q_j\mathbb{M}^{\Gamma_j}\mathbf{v}_{j,n}^k + \mathbb{D}_{j,s}^n\boldsymbol{\varphi}_{j,s}^{n-1/2} = \frac{2i}{\Delta t}Q_j\mathbb{M}^{\Gamma_j}Q_j^T\boldsymbol{\varphi}_{j,s}^{n-1}, \quad n \geqslant 1, \\ \Rightarrow\ &\boldsymbol{\varphi}_{j,s}^{n-1/2} = (\mathbb{D}_{j,s}^n)^{-1}Q_j\mathbb{M}^{\Gamma_j}\mathbf{v}_{j,n}^k + (\mathbb{D}_{j,s}^n)^{-1}\frac{2i}{\Delta t}Q_j\mathbb{M}^{\Gamma_j}Q_j^T\boldsymbol{\varphi}_{j,s}^{n-1}, \\ &\text{and } \boldsymbol{\varphi}_{j,s}^{n-1} = 2\boldsymbol{\varphi}_{j,s}^{n-3/2} - \boldsymbol{\varphi}_{j,s}^{n-2},\ \boldsymbol{\varphi}_{j,s}^0 = 0,\ n \geqslant 2, \end{aligned}$$

where $\mathbb{D}_{j,s}^n = Q_j(\frac{2i}{\Delta t}\mathbb{M}^{\Gamma_j} - \mathbb{S}^{\Gamma_j} + \mathbb{M}_{W_n}^{\Gamma_j} + d_s^m\mathbb{M}^{\Gamma_j})Q_j^T$. By induction, we could have an expression of $\boldsymbol{\varphi}_{j,s}^{n-1/2}$:

$$\boldsymbol{\varphi}_{j,s}^{n-1/2} = \sum_{p=1}^n \mathbb{L}_{j,s}^{n,p}\mathbf{v}_{j,p}^k, \tag{2.18}$$

where $\mathbb{L}_{j,s}^{n,p}$ are matrix. Replacing $\boldsymbol{\varphi}_{j,s}^{n-1/2}$ by $\mathbf{v}_{j,p}^k$ in the first formula of (2.13) gives

(2.19)
$$\Big(\mathbb{A}_{j,n} + i(\sum_{s=0}^m a_s^m)\cdot\mathbb{M}^{\Gamma_j}\Big)\mathbf{v}_{j,n}^k + \sum_{p=1}^n \mathbb{Y}_j^{n,p}\mathbf{v}_{j,p}^k = \frac{2i}{\Delta t}\mathbb{M}_j\mathbf{u}_{j,n-1}^k - \mathbb{M}^{\Gamma_j}Q_j^T\begin{pmatrix}\mathbf{l}_{j,n}^k \\ \mathbf{r}_{j,n}^k\end{pmatrix},$$

where $\mathbb{Y}_j^{n,p} := -i\sum_{s=1}^m a_s^m d_s^m\mathbb{M}^{\Gamma_j}Q_j^T\mathbb{L}_{j,s}^{n,p}$ since

$$-i\sum_{s=1}^m a_s^m d_s^m\mathbb{M}^{\Gamma_j}Q_j^T\cdot\sum_{p=1}^n\mathbb{L}_{j,s}^{n,p}\mathbf{v}_{j,p}^k = -i\sum_{p=1}^n\big(\sum_{s=1}^m a_s^m d_s^m\mathbb{M}^{\Gamma_j}Q_j^T\mathbb{L}_{j,s}^{n,p}\big)\mathbf{v}_{j,p}^k.$$

Then, we could define the matrix $\mathbf{B}_j$ according to (2.19), which leads to (2.17). ∎

2.3. Classical DDS algorithm. The other algorithm we consider in this paper is the domain decomposition in space algorithm (DDS). The equation (1.1) is first discretized in time on the entire domain $(0,T)\times\Omega$. We use the notations defined in the previous subsection. The time interval $(0,T)$ is discretized uniformly with N_T intervals. The Crank-Nicolson scheme on $(0,T)\times\Omega$ reads

$$i\frac{u_n - u_{n-1}}{\Delta t} + \Delta\frac{u_n + u_{n-1}}{2} + \frac{V_n + V_{n-1}}{2}\frac{u_n + u_{n-1}}{2} = 0,\ 1 \leqslant n \leqslant N_T.$$

By introducing new variables $v_n = (u_n + u_{n-1})/2$ with $v_0 = u_0$ and $W_n = (V_n + V_{n-1})/2$, we get a stationary equation defined on Ω with unknown v_n

$$\mathcal{L}_{\mathbf{x}}v_n := (\frac{2i}{\Delta t} + \Delta + W_n)v_n = \frac{2i}{\Delta t}u_{n-1}, \tag{2.20}$$

and $u_n = 2v_n - u_{n-1}$. Then, the optimized Schwarz algorithm is applied to the stationary equation (2.20). We denote by $R_j, j = 1, 2, ..., N$ the restriction operator from Ω to Ω_j. At time t_n, the classical algorithm reads

$$\left\{\begin{array}{ll} \mathscr{L}_{\mathbf{x}} v_j^k = \frac{2i}{\Delta t} R_j u_{n-1}, & (t,x,y) \in (0,T) \times \Omega_j, \\ \partial_{\mathbf{n}_j} v_j^k + \overline{S}_j v_j^k = \partial_{\mathbf{n}_j} v_{j-1}^{k-1} + \overline{S}_j v_{j-1}^{k-1}, & x = a_j, \\ \partial_{\mathbf{n}_j} v_j^k + \overline{S}_j v_j^k = \partial_{\mathbf{n}_j} v_{j+1}^{k-1} + \overline{S}_j v_{j+1}^{k-1}, & x = b_j, \\ \partial_{\mathbf{n}_j} v_j^k = 0, & y = y_l, y_d. \end{array}\right. \tag{2.21}$$

with a special treatment for the two extreme subdomains: $\partial_{\mathbf{n}_1} v_1^k + \overline{S}_1 v_1^k = 0, x = a_1$, $\partial_{\mathbf{n}_N} v_N^k + \overline{S}_N v_N^k = 0, x = b_N$. Since the interval (t_{n-1}, t_n) contains only one time step, the DDS algorithm could be numerically interpreted as a combination of some SWR algorithms

Algorithm 1: DDS algorithm

The initial datum is u_0.

for $n = 1, 2, ..., N_T$ **do**

Apply the SWR algorithm to

$$\left\{\begin{array}{l} \mathscr{L} u = 0, \ (t,x,y) \in (t_{n-1}, t_n) \times \Omega, \\ u(0,x,y) = u_{n-1}(x,y), \ (x,y) \in \Omega, \end{array}\right.$$

where $t_n = n\Delta t$.

3. Discrete interface problem. The aim of this section is to study the discrete form of the interface problem (2.7). We show in the next propositions that the N problems (2.16) on each subdomain could be written globally as

$$\mathbf{g}^{k+1} = \mathcal{L}_h \mathbf{g}^k + d, \tag{3.1}$$

where the global interface vector is defined by

$$\mathbf{g}^k = (\mathbf{g}_1^{k,T}, \mathbf{g}_2^{k,T}, ..., \mathbf{g}_N^{k,T})^T \in \mathbb{C}^{(2N-2)\times N_y \times N_T},$$

d is a vector and $\mathcal{L}_h$ is a block matrix

(3.2)

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

The matrix $\mathcal{L}_h$ is called interface matrix in this paper. The formula (3.1) is the discrete form of the interface problem (2.7). Actually, it could be interpreted as the fixed point method for the linear system

$$(I - \mathcal{L}_h)\mathbf{g} = d. \tag{3.3}$$

Interpreting (3.1) as (3.3) allows to use the Krylov methods [17] (ex. Gmres and Bicgstab), which could accelerate the convergence according to the numerical tests in Section 6. We remark that in the classical algorithm, the matrix $\mathcal{L}_h$ is not explicitly known. It is considered as an operator. To implement the fixed point method or Krylov methods, it is enough to define the application of $I - \mathcal{L}_h$ to a vector.

PROPOSITION 3.1. *If we consider the Robin transmission condition, the discrete form of the interface problem is given by* (3.1).

Proof. According to (2.14), (2.16) and the definitions of $\mathbf{g}^k$, it is easy to verify that

$$\begin{aligned} X^{j,1} &= -I - 2ip \cdot \mathbf{Q}_{j,l}(\mathbf{A}_j - \mathbf{B}_j)^{-1}\mathbf{M}^{\Gamma_j}\mathbf{Q}_{j,l}^T, \\ X^{j,2} &= -2ip \cdot \mathbf{Q}_{j,l}(\mathbf{A}_j - \mathbf{B}_j)^{-1}\mathbf{M}^{\Gamma_j}\mathbf{Q}_{j,r}^T, \\ X^{j,3} &= -2ip \cdot \mathbf{Q}_{j,r}(\mathbf{A}_j - \mathbf{B}_j)^{-1}\mathbf{M}^{\Gamma_j}\mathbf{Q}_{j,l}^T, \\ X^{j,4} &= -I - 2ip \cdot \mathbf{Q}_{j,r}(\mathbf{A}_j - \mathbf{B}_j)^{-1}\mathbf{M}^{\Gamma_j}\mathbf{Q}_{j,r}^T. \end{aligned} \tag{3.4}$$

□

PROPOSITION 3.2. *If we consider the transmission condition S_{2p}^m, then* (3.1) *is the discrete form of the interface problem.*

Proof. By using (2.13), (2.15) and (2.18), we have

$$\widetilde{S}_j \mathbf{v}_{j,n}^k = -c_a \mathbf{v}_{j,n}^k + i\sum_{s=1}^m a_s^m d_s^m \sum_{p=1}^n \mathbb{L}_{j,s}^{n,p} \mathbf{v}_{j,p}^k = -c_a \mathbf{v}_{j,n}^k + i\sum_{p=1}^n \big(\sum_{s=1}^m a_s^m d_s^m \mathbb{L}_{j,s}^{n,p}\big)\mathbf{v}_{j,p}^k.$$

We could easily verify that

$$\begin{aligned} X^{j,1} &= -I + 2\mathbf{Q}_{j,l}\mathbf{B}_j^S(\mathbf{A}_j - \mathbf{B}_j)^{-1}\mathbf{M}^{\Gamma_j}\mathbf{Q}_{j,l}^T, \\ X^{j,2} &= 2\mathbf{Q}_{j,l}\mathbf{B}_j^S(\mathbf{A}_j - \mathbf{B}_j)^{-1}\mathbf{M}^{\Gamma_j}\mathbf{Q}_{j,r}^T, \\ X^{j,3} &= 2\mathbf{Q}_{j,r}\mathbf{B}_j^S(\mathbf{A}_j - \mathbf{B}_j)^{-1}\mathbf{M}^{\Gamma_j}\mathbf{Q}_{j,l}^T, \\ X^{j,4} &= -I + 2\mathbf{Q}_{j,r}\mathbf{B}_j^S(\mathbf{A}_j - \mathbf{B}_j)^{-1}\mathbf{M}^{\Gamma_j}\mathbf{Q}_{j,r}^T, \end{aligned} \tag{3.5}$$

where the matrix $\mathbf{B}_j^S$ is defined by

$$\mathbf{B}_j^S = \begin{pmatrix} -c_a I + i\sum_{s=1}^m a_s^m d_s^m \mathbb{L}_{j,s}^{1,1} & & & \\ i\sum_{s=1}^m a_s^m d_s^m \mathbb{L}_{j,s}^{2,1} & -c_a I + i\sum_{s=1}^m a_s^m d_s^m \mathbb{L}_{j,s}^{2,2} & & \\ \vdots & \vdots & \ddots & \\ i\sum_{s=1}^m a_s^m d_s^m \mathbb{L}_{j,s}^{N_T,1} & i\sum_{s=1}^m a_s^m d_s^m \mathbb{L}_{j,s}^{N_T,2} & \cdots & -c_a I + i\sum_{s=1}^m a_s^m d_s^m \mathbb{L}_{j,s}^{N_T,N_T} \end{pmatrix}.$$

□

We are interested in the structure of the subblock of $\mathcal{L}_h$ for time independent potential $V = V(x, y)$

$$\begin{aligned} &X^{1,4} = \{x_{n,s}^{1,4}\}_{1\leqslant n,s\leqslant N_T}, \\ &X^{j,1} = \{x_{n,s}^{j,1}\}_{1\leqslant n,s\leqslant N_T}, X^{j,2} = \{x_{n,s}^{j,2}\}_{1\leqslant n,s\leqslant N_T}, \\ &X^{j,3} = \{x_{n,s}^{j,3}\}_{1\leqslant n,s\leqslant N_T}, X^{j,4} = \{x_{n,s}^{j,4}\}_{1\leqslant n,s\leqslant N_T}, j = 2, 3, ..., N-1, \\ &X^{N,1} = \{x_{n,s}^{N,1}\}_{1\leqslant n,s\leqslant N_T}. \end{aligned} \tag{3.6}$$

where $x_{n,s}^{j,1}, x_{n,s}^{j,2}, x_{n,s}^{j,3}, x_{n,s}^{j,4} \in \mathbb{C}^{N_y \times N_y}$ are submatrices.

This structure is described in Figure 3 for 3 time steps and 6 nodes on the interface between two subdomains. Each sub-diagonal block is identical. We present this property mathematically in proposition 3.3 with the transmission condition Robin or S_{2p}^m. The demonstration is similar to that for one dimensional Schrödinger equation [7]. The formal difference is that the flux are scalar in one dimension, but here $\mathbf{l}_{j,n}^k$ and $\mathbf{r}_{j,n}^k$ are vectors.

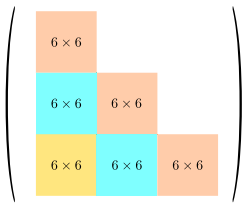

FIG. 3. *Block structure,* $N_T = 3$, $N_y = 6$.

PROPOSITION 3.3. *For the transmission Robin (resp.* S_{2p}^m*), assuming that the linear system* (2.12) *(resp.* (2.13)*) is not singular, if* $V = V(x, y)$*, then the matrix* $X^{1,4}$*,* $X^{j,1}$*,*$X^{j,2}$*,* $X^{j,3}$*,* $X^{j,4}$*,* $j = 2, 3, ..., N-1$ *and* $X^{N,1}$ *are block 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 $2 \leqslant s \leqslant n \leqslant N_T$.

We now consider the structure of the sub-blocks of $\mathcal{L}_h$ for $V = 0$.

PROPOSITION 3.4. *Assuming that the matrix* $\mathbf{A}_j - \mathbf{B}_j$ *is not singular with the Robin or* S_{2p}^m *transmission condition, if the potential* $V = 0$ *and the size of subdomains* Ω_j *are equal* ($b_1 - a_1 = b_2 - a_2 = ... = b_N - a_N$)*, then the subblocks of* $\mathcal{L}_h$ *satisfy*

$$\begin{aligned}
&X^{2,1} = X^{3,1} = \cdots = X^{N,1}, \quad X^{2,2} = X^{3,2} = \cdots = X^{N-1,2},\\
&X^{2,3} = X^{3,3} = \cdots = X^{N-1,3},\ X^{1,4} = X^{2,4} = \cdots = X^{N-1,4}.
\end{aligned} \tag{3.7}$$

Proof. Under the assumptions, it is easy to see that

$$\mathbb{A}_1 = \mathbb{A}_2 = \cdots = \mathbb{A}_N,\ \mathbb{M}_{j,W_n} = 0,\ \mathbb{M}^{\Gamma_1} = \mathbb{M}^{\Gamma_2} = \cdots = \mathbb{M}^{\Gamma_N},$$

and

$$Q_{1,l} = Q_{2,l} = \cdots = Q_{N,l},\ Q_{1,r} = Q_{2,r} = \cdots = Q_{N,r}.$$

Thus

$$\mathbf{A}_1 = \mathbf{A}_2 = \cdots = \mathbf{A}_N,\ \mathbf{B}_1 = \mathbf{B}_2 = \cdots = \mathbf{B}_N,\ \mathbf{Q}_1 = \mathbf{Q}_2 = \cdots = \mathbf{Q}_N,$$

and $\mathbf{B}_1^S = \mathbf{B}_2^S = \cdots = \mathbf{B}_N^S$ for the transmission condition S_{2p}^m. The conclusion is obvious from (3.4) or (3.5). □

4. New algorithms for free Schrödinger equation.

4.1. New SWR algorithm. Based on the propositions in the previous section, we propose here a new SWR algorithm for the free Schrödinger equation $V = 0$. We always suppose that the size of each subdomain is identical.

Algorithm 2: New SWR algorithm, $V = 0$

1: Build $\mathcal{L}_h$ and d in $(I - \mathcal{L}_h)\mathbf{g} = d$ explicitly.
2: Solve the linear system $(I - \mathcal{L}_h)\mathbf{g} = d$ by an iterative method (fixed point method or Krylov methods).
3: Solve the Schrödinger equation on each subdomain on $(0, T) \times \Omega_j$ using the flux obtained from step 2.

Mathematically, this new SWR algorithm is identical to the classical one. The difference is that in the new algorithm, we construct explicitly the matrix $\mathcal{L}_h$ and the vector d (the first step). We are going to show that this construction is not costly and it is super-scalable in theory. In the formulas (2.14) and (2.15), we consider $\mathbf{l}_{j,n}^k$ and $\mathbf{r}_{j,n}^k$ as inputs, and $\mathbf{l}_{j+1,n}^{k+1}$ and $\mathbf{r}_{j-1,n}^{k+1}$ as outputs:

$$\text{inputs: } \mathbf{l}_{j,n}^k, \mathbf{r}_{j,n}^k \longrightarrow \begin{array}{c}\text{Robin : (2.14)}\\ S_{2p}^m : \text{ (2.15)}\end{array} \longrightarrow \text{outputs : } \mathbf{l}_{j+1,n}^{k+1}, \mathbf{r}_{j-1,n}^{k+1}.$$

It is obvious that

$$d = \begin{pmatrix} d_{1,r} \\ d_{2,l} \\ d_{2,r} \\ \vdots \\ d_{N,l} \end{pmatrix} = \mathcal{R}_h \cdot \mathbf{0}, \text{ where } d_{j-1,r} = \begin{pmatrix} \mathbf{r}_{j-1,1}^{k+1} \\ \mathbf{r}_{j-1,2}^{k+1} \\ \vdots \\ \mathbf{r}_{j-1,N_T}^{k+1} \end{pmatrix}, \ d_{j+1,l} = \begin{pmatrix} \mathbf{l}_{j+1,1}^{k+1} \\ \mathbf{l}_{j+1,2}^{k+1} \\ \vdots \\ \mathbf{l}_{j+1,N_T}^{k+1} \end{pmatrix},$$

and the vectors $\mathbf{r}_{j-1,n}^{k+1}$ and $\mathbf{l}_{j+1,n}^{k+1}$, $n = 1, 2, ..., N_T$ are obtained by (2.14) or (2.15) with

$$\mathbf{l}_{j,n}^k = \mathbf{r}_{j,n}^k = \mathbf{0}, \ n = 1, 2, ..., N_T, \ j = 1, 2, ..., N.$$

The Schrödinger equation is solved on each subdomain $(0, T) \times \Omega_j, j = 1, 2, ..., N$ one time. The vector is stored in a distributed manner using the PETSc library [5]

$$d = 2 \begin{pmatrix} d_{1,r} \\ \vdots \\ d_{j,l} \\ d_{j,r} \\ \vdots \\ d_{N,l} \end{pmatrix} \begin{matrix} \} \text{MPI 0, } N_T \times N_y \text{ elements} \\ \\ \Big\} \text{MPI } j, \ 2N_T \times N_y \text{ elements} \\ \\ \} \text{MPI } N, \ N_T \times N_y \text{ elements} \end{matrix}$$

We have seen in Proposition 3.3 and Proposition 3.4 that if the geometries of the subdomains are identical and the potential is zero, then it is sufficient to compute four subblocks to explicitly build the matrix $\mathcal{L}_h$. Without loss of generality, we build the blocks $X^{2,1}$, $X^{2,2}$, $X^{2,3}$ and $X^{2,4}$. We define the vectors $\mathbf{e}_s = (0, 0, ..., 1, ...0) \in \mathbb{C}^{N_T}$, where all the elements are zero except the s-th, which is one. The columns $s, s =$

$1, 2, ..., N_y$ of $X^{2,1}$ and $X^{2,3}$ are

$$X^{2,1}\begin{pmatrix}\mathbf{e}_s\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\end{pmatrix} = \begin{pmatrix}\mathbf{r}_{1,1}^{k+1}\\ \mathbf{r}_{1,2}^{k+1}\\ \vdots\\ \mathbf{r}_{1,N_T}^{k+1}\end{pmatrix} - d_{1,r} \text{ and } X^{2,3}\begin{pmatrix}\mathbf{e}_s\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\end{pmatrix} = \begin{pmatrix}\mathbf{l}_{3,1}^{k+1}\\ \mathbf{l}_{3,2}^{k+1}\\ \vdots\\ \mathbf{l}_{3,N_T}^{k+1}\end{pmatrix} - d_{3,l},$$

where the vectors $\mathbf{r}_{1,n}^{k+1}$ and $\mathbf{l}_{3,n}^{k+1}$, $n = 1, 2, ..., N_T$ are obtained by the formula (2.14) or (2.15) with

$$\mathbf{l}_{2,1}^{k} = \mathbf{e}_s,\ \mathbf{l}_{2,n}^{k} = 0,\ n = 2, 3, ..., N_T,$$
$$\mathbf{r}_{2,n}^{k} = \mathbf{0},\ n = 1, 2, ..., N_T.$$

In the same way, the columns $s, s = 1, 2, ...N_T$ of $X^{2,2}$ and $X^{2,4}$ are

$$X^{2,2}\begin{pmatrix}\mathbf{e}_s\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\end{pmatrix} = \begin{pmatrix}\mathbf{r}_{1,1}^{k+1}\\ \mathbf{r}_{1,2}^{k+1}\\ \vdots\\ \mathbf{r}_{1,N_T}^{k+1}\end{pmatrix} - d_{1,r} \text{ and } X^{2,4}\begin{pmatrix}\mathbf{e}_s\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\end{pmatrix} = \begin{pmatrix}\mathbf{l}_{3,1}^{k+1}\\ \mathbf{l}_{3,2}^{k+1}\\ \vdots\\ \mathbf{l}_{3,N_T}^{k+1}\end{pmatrix} - d_{3,l},$$

where the vectors $\mathbf{r}_{1,n}^{k+1}$ and $\mathbf{l}_{3,n}^{k+1}$, $n = 1, 2, ..., N_T$ are obtained by the formula (2.14) or (2.15) with

$$\mathbf{l}_{2,n}^{k} = \mathbf{0},\ n = 1, 2, ..., N_T,$$
$$\mathbf{r}_{2,1}^{k} = \mathbf{e}_s,\ \mathbf{r}_{2,n}^{k} = 0,\ n = 2, 3, ..., N_T.$$

Concretely, the Schrödinger equation on a single subdomain $(0, T) \times \Omega_2$ is solved $2N_y$ times to build the matrix $\mathcal{L}_h$. The resolutions are all independent. We fix one MPI process per domain. To construct the matrix $\mathcal{L}_h$, we use the N MPI processes to solve the equation on a single subdomain $((0, T) \times \Omega_2)$ $2N_y$ times. Each MPI process solves the Schrödinger equation on a single subdomain maximum

$$N_{\text{mpi}} := [\frac{2N_y}{N}] + 1 \text{ times},$$

where $[x]$ is the integer part of x. This construction is super-scalable in theory. If N is doubled, then the size of subdomain is divided by two and N_{mpi} is also approximately halved.

The transposed matrix of $\mathcal{L}_h$ is stored in a distributed manner in the form of the PETSc library. As shown by (3.2), the first block column of $\mathcal{L}_h$ is in MPI process 0. The second and third blocks columns are in MPI process 1, and so on for other processes. The size of each block is $(N_T \times N_y) \times (N_T \times N_y)$. Each block contains $(N_T + 1) \times N_T/2 \times N_y^2$ nonzero elements.

4.2. New DDS algorithm. We denote the interface problem of

$$\begin{cases} \mathscr{L}u = 0, & (t, x, y) \in (t_{n-1}, t_n) \times \Omega, \\ u(0, x, y) = u_{n-1}(x, y), & (x, y) \in \Omega, \end{cases}$$

by $(I - \mathcal{L}_{h,n})\mathbf{g}_n = d_n$ where $\mathcal{L}_{h,n}$ is matrix and d_n is vector. According to (3.4) or (3.5), the interface matrix is independent of the initial datum, thus we have

PROPOSITION 4.1. *For the transmission Robin and S_{2p}^m, the interface matrix satisfy*

$$\mathcal{L}_{h,1} = \mathcal{L}_{h,2} = ... = \mathcal{L}_{h,N_T}.$$

We have interpreted the DDS method as combination of some SWR algorithms (Algorithm 1). As a direct corollary of the new SWR algorithm, we propose here a new DDS algorithm

Algorithm 3: New DDS algorithm, $V = 0$

1: Build $\mathcal{L}_{h,1}$ explicitly.
2: The initial datum is u_0.
for $n = 1, 2, ..., N_T$ **do**
  2.1: Build d_n on time t_n,
  2.2: Solve the linear system $(I - \mathcal{L}_{h,n})\mathbf{g}_{h,n} = d_n$ by an iterative method, where the initial vector is chosen as $\mathbf{g}_{h,n-1}$.
  2.3: Solve the Schrödinger equation on each subdomain $(t_{n-1}, t_n) \times \Omega_j$ using the flux from step 2.2 to compute u_n.

5. Preconditioned algorithms for general linear potential. In the case of a non zero potential, the proposition 3.4 does not hold. Thus we could not construct easily the matrix $\mathcal{L}_h$. The aim of this section is to present preconditioned algorithms for the Schrödinger equation (1.1) with a nonzero potential $V = V(t, x, y)$. Adding a preconditioner in (3.3) leads to a preconditioned SWR algorithm

$$P^{-1}(I - \mathcal{L}_h)\mathbf{g} = P^{-1}d, \tag{5.1}$$

We propose here

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

where $\mathcal{L}_0$ is the interface matrix in (3.3) defined for the free Schrödinger equation. As mentioned in the previous section, it could be easily constructed numerically and it is stored in a distributed manner. We have explained that the consumed memory of each MPI process is $4(Ny)^2$ for $j = 2, 3, ..., N-1$ and $2(N_y)^2$ for $j = 1, N$. The size of the matrices for each subdomain is $N_x \times N_y$. Thus, the memory consumed by a direct LU method is about $(N_x \times N_y)^2$. It is usually much larger than $4(N_y)^2$. Thus, the storage of P is not very important.

For any vector y, the vector $x := P^{-1}y$ is computed by solving the linear system

$$Px = y.$$

with Krylov methods (Gmres or Bicgstab). However, the size of P increases linearly with the number of subdomains N. Also the number of operations for multiplying $\mathcal{L}_0$ and a vector (which is the basic operation of Krylov method) is not negligible compared with solving the equations on each subdomain if N is large. Thus, the application of the preconditioner increases.

We could derive straightforwardly a preconditioned DDS algorithm from the point of view of Algorithm 1. The preconditioner for all time steps $n = 1, 2, ..., N_T$ is always chosen as

$$P = I - \mathcal{L}_{0,1},$$

where $\mathcal{L}_{0,1}$ is the interface matrix of

$$\begin{cases} i\partial_t u + \Delta u = 0, & (t,x,y) \in (0,\Delta t) \times \Omega, \\ u(0,x,y) = u_0(x,y), & (x,y) \in \Omega. \end{cases}$$

To construct this preconditioner, it is enough to solve the free Schrödinger equation on subdomain $(0,\Delta t) \times \Omega_2$.

6. Numerical results. The complete domain $\Omega = (-16,16) \times (-8,8)$ is decomposed into N equal subdomains (ex. for $N = 3$ see Figure 1). The size of element is $\Delta x \times \Delta y$. We consider two different meshes

$$\Delta x = 1/128,\ \Delta y = 1/8;$$
$$\Delta x = 1/2048,\ \Delta y = 1/128.$$

With the first mesh, it is possible to solve the Schrödinger equation (1.1) on the entire domain on a simgle node of a cluster composed of 92 nodes (16 cores/node, Intel Sandy Bridge E5-2670, 32GB memory/node). Thus we could observe if the parallel algorithms allow to reduce the total computation time of the sequential algorithm. We are interested in the strong scalability up to 1024 subdomains. To make sure that each subdomain contains enough nodes, the second mesh is used for large N. The initial datum in this section is

$$u_0(x,y) = e^{-x^2-y^2-0.5ix}. \quad \text{(see Figure 4)} \tag{6.1}$$

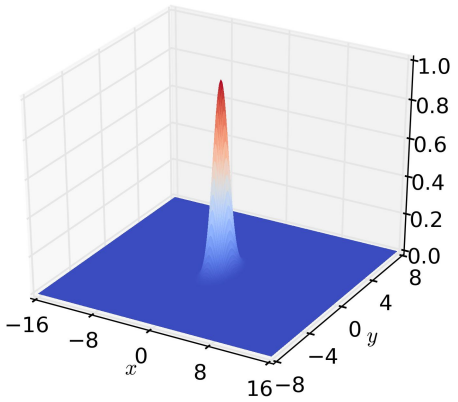

FIG. 4. *Initial datum* $|u_0|$, $u_0 = e^{-x^2-y^2-0.5ix}$.

This section is composed of two subsections, one consider the free Schrödinger equation ($V = 0$), the other consider the Schrödinger equation with a linear potential $V = x^2 + y^2$.

6.1. The free Schrödinger equation. In this part, first we compare the SWR method and the DDS algorithm. Next, the classical DDS algorithm and the new DDS algorithm are compared. Finally, the influence of parameters is studied in Section 6.1.3.

6.1.1. SWR vs. DDS. We compare the SWR method and the DDS method in the framework of classical algorithm. The time step is fixed as $\Delta t = 0.01$. The mesh is $\Delta x = 1/128$, $\Delta y = 1/8$. Both methods are applied to $(0,T) \times \Omega$ with some different final time T. We denote by N^S_{iter} the number of iterations required for convergence of the SWR method, N^D_{iter} the number of iterations of the first time step of the DDS method and T^S (resp. T^D) the total computation time of the SWR method (resp. DDS method). Table 1 presents the numbers of iterations and the computation times of both methods for $N = 2$ and $N = 32$, where the transmission condition is $S^m_{2p}, m = 5$. The initial vector is the zero vector and the Gmres method is used on the interface problem. We could see that N^S_{iter} increase with the final time T and $N^S_{\text{iter}} > N^D_{\text{iter}}$. Thus $T^S > T^D$. In the following subsections, without special

TABLE 1
Number of iterations and computation time of the SWR method and the DDS method for $N = 2, 32$.

T	$N=2$				$N=32$			
	N^S_{iter}	N^D_{iter}	T^S	T^D	N^S_{iter}	N^D_{iter}	T^S	T^D
0.05	17	9	17.6	12.3	17	10	1.5	1.9
0.1	25	9	44.5	21.5	25	10	3.5	1.8
0.15	30	9	78.4	30.9	31	10	6.4	2.6
0.2	44	9	147.9	40.0	45	10	11.8	3.4
0.25	51	9	215.0	49.3	52	10	16.9	4.1
0.3	55	9	271.0	58.6	55	10	21.2	4.8
0.35	58	9	332.1	67.9	59	10	26.3	5.6
0.4	61	9	402.9	77.2	62	10	32.0	6.3
0.45	64	9	474.5	86.4	65	10	37.6	7.1
0.5	68	9	557.6	96.1	73	10	46.3	7.8

statement, we only consider the DDS method since it does requires less computation time.

6.1.2. Comparison of classical and new algorithms. In this part, we are interested in the performance (number of iterations and computation time) of the classical and the new algorithms with the two transmission conditions. We observe the strong scalability of the two algorithms. Both algorithms and transmission conditions are compared in the framework of the DDS method. The final time is $T = 0.5$ and the time step is fixed as $\Delta t = 0.01$. The Gmres method is used on the interface problem. The initial vector is the zero vector. Since there is no theoretical result for us on the choice of the optimal parameter p in the Robin transmission condition, we make tests with different p to find the numerically optimal one. However, we will see in the next subsection that the number of iterations and the computing time are not sensitive to p using the Gmres method on the interface problem. Thus, it is difficult to choose the optimal one. We take $p = 15$ for the mesh $\Delta x = 1/128$, $\Delta y = 1/8$ and $p = 10$ for the mesh $\Delta x = 1/2048$, $\Delta y = 1/64$. For the transmission condition S^m_{2p}, we take $m = 5$ according to the tests conducted in the following subsection.

We first consider the mesh $\Delta x = 1/128$, $\Delta y = 1/8$. Figure 5 presents the convergence history of the first time step for $N = 2$ and $N = 32$. We also show the number of iterations of the first time step and the total computation time in Table 2 for $N = 2, 4, 8, 16, 32$. Since the boundary condition is imposed on the boundary of Ω, the computation time of solving the Schrödinger equation (1.1) on the entire domain are not same. We denote by "Robin ref" for the Robin transmission condition and "S^5_{2p} ref." for the transmission condition S^5_{2p}. We could see that

1. the new algorithm could reduce the computation time compared with the classical algorithm,

2. the number of iterations is almost independent of the number of subdomains and the algorithm is scalable,

3. the algorithm converges faster with the transmission condition S_{2p}^m, but it takes more computational time since the application of the transmission condition S_{2p}^m is more expensive than the Robin transmission condition.

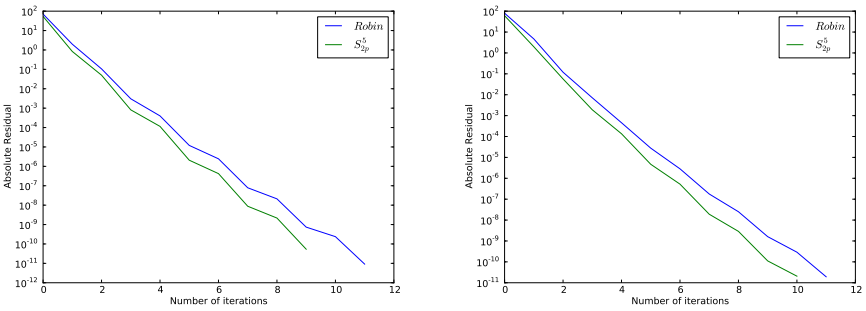

FIG. 5. *Convergence history of the first time step for $N = 2$ (left) ,$N = 32$ (right).*

TABLE 2

Number of iterations of the first time step and total computation time in seconds, $T = 0.5$, $\Delta t = 0.01$, $\Delta x = 1/128$, $\Delta y = 1/8$.

N		2	4	8	16	32
Number of iterations	Robin*	11	11	11	11	11
	S_{2p}^5	9	9	9	9	10
Computation time	Robin ref.	16.0				
	Robin cls.	63.1	32.6	17.5	11.0	5.4
	Robin new	30.0	9.7	4.4	2.5	1.3
	S_{2p}^5 ref.	22.1				
	S_{2p}^5 cls.	96.1	49.8	26.7	15.0	7.8
	S_{2p}^5 new	38.2	14.7	6.6	3.4	1.8

*: $p = 15$.

Next we make tests with the mesh $\Delta x = 1/2048$ $\Delta y = 1/64$. The entire domain is divided into $N = 128, 256, 512, 1024$ subdomains. We present in Figure 6 the convergence history for $N = 256, 1024$ and in Table 3 the numbers of iterations and the total computation time. We can see that the classical and the new algorithms are not very scalable since the number of iterations increases with the number of subdomains. However, the new algorithm takes less computation time. Besides, the number of iterations required of the transmission condition S_{2p}^m is less than the Robin transmission condition, but the computation times are similar.

6.1.3. Influence of parameters. We study in this part the influence of parameters: m in the transmission condition S_{2p}^m and p in the transmission condition Robin. The time step is fixed as $\Delta t = 0.01$. The mesh is $\Delta x = 1/128$, $\Delta y = 1/8$. Three different methods are used on the interface problem: fixed point method, Gmres method and Bicgstab method. Two initial vectors are considered: the zero vector and the random vector. In our tests, the algorithm initialized with the zero vector converges

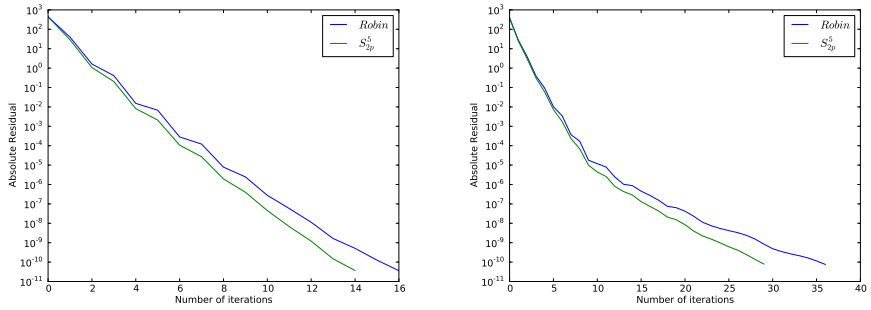

FIG. 6. *Convergence history of the first time step for $N = 256$ (left), $N = 1024$ (right).*

TABLE 3

Number of iterations of the first time step and total computation time in seconds, $T = 0.5$, $\Delta t = 0.01$, $\Delta x = 1/2048$, $\Delta y = 1/64$.

N		128	256	512	1024
Number of iterations	Robin*	14	16	22	36
	S^5_{2p}	12	14	19	29
Computation time	Robin cls.	250.2	143.8	92.5	101.4
	Robin new	59.1	38.1	36.2	52.3
	S^5_{2p} cls.	-	187.5	162.6	127.5
	S^5_{2p} new	-	42.7	36.2	45.0

-: the memory is insufficient; *: $p = 10$.

faster than with the random vector. However, in this subsection, our goal is to study the influence of parameters. As explained in [11], initialization with a zero vector to compute a smooth solution makes that the error contains only low frequencies and could therefore draw the wrong conclusions. Thus we consider both of the two initial vectors, but we have no wish to compare them.

The parameter m in the transmission condition S^m_{2p}. The numbers of iterations of the first time step with some different m are presented in Table 4. We could see that the number of iterations is not sensitive to the parameter m if the Gmres method or the Bicgstab method is used on the interface problem. However, if the fixed point method is used, increasing the order of S^m_{2p} does not ensure better convergence property. The transmission condition S^m_{2p} is based on formal Padé approximation of square root operator, this approximation may deteriorate for large m.

The parameter p in the transmission condition Robin. We present in Table 5 the numbers of iterations with some different p for $N = 32$. From the table, we can see that the algorithm is not sensitive to p if the Gmres method or the Bicgstab method is used on the interface problem, while it exists an optimal p (marked with an underline) if the fixed point method is used. Besides, the Krylov methods could accelerate the convergence, especially if the initial vector is the random vector.

In conclusion, the difference between the two transmission conditions is clear if the fixed point method is applied to the interface problem and the initial vector is the random vector (see Tables 4 and 5 together). The number of iterations for the transmission condition S^m_{2p} is less than that for the transmission condition Robin. But

TABLE 4
Number of iterations for different m, $N = 32$.

	Fixed point		Gmres		Bicgstab	
m	Zero	Random	Zero	Random	Zero	Random
3	34	124	11	30	6	17
4	26	96	10	28	6	16
5	21	79	10	27	5	15
6	18	68	9	26	5	15
7	17	61	9	25	5	14
8	18	56	9	25	5	14
9	19	52	10	25	5	14
10	21	50	10	25	5	14
15	32	46	11	26	6	15
20	43	50	12	28	6	16

TABLE 5
Number of iterations for different p, $N = 32$.

	Fixed point		Gmres		Bicgstab	
p	Zero	Random	Zero	Random	Zero	Random
5	57	580	11	35	6	21
10	34	315	11	32	6	19
15	32	239	11	31	6	18
20	35	209	11	31	6	19
25	40	200	11	32	6	19
30	46	199	11	32	6	19
35	52	204	12	33	6	19
40	59	209	12	33	6	20
45	65	222	12	34	6	20
15	32					
26		198				

the difference is smaller using the Krylov methods and the zero vector as the initial vector. From the point of view of computation time, the zero vector and the Gmres method are a good choice. According to the tests in the previous subsection, the computation time for the two transmission conditions are similar.

6.2. Case of non-zero potential. The aim of this section is to compare the classical algorithm and the preconditioned algorithm in the framework of the DDS method with the fixed point method used on the interface problem. We first consider the potential $V = x^2 + y^2$. Let us donate by N_{nopc} (resp. N_{pc}) the number of iterations required for convergence of the classical algorithm (resp. the preconditioned algorithm) and T_{nopc} (resp. T_{pc}) the computation time of the classical algorithm (resp. the preconditioned algorithm).

First, we present in Figure 7 the convergence history of the first time step for $N = 2, 32$ subdomains. Table 6 shows the number of iterations of the first time step and the total computation time to realize a complete simulation. The mesh is $\Delta x = 1/128$, $\Delta y = 1/8$. As mentioned before, it is possible to solve the Schrödinger equation on the entire domain with this mesh. The computation time is denoted by T^{ref}. We could see that

1. all algorithms are robust and the number of iterations is independent of number of subdomains,

2. the classical algorithm and the preconditioned algorithm are both scalable,

3. the preconditioner allows to reduce the number of iterations and the total computation time,

4. T_{nopc}, T_{pc} are less than T^{ref} if N is large.

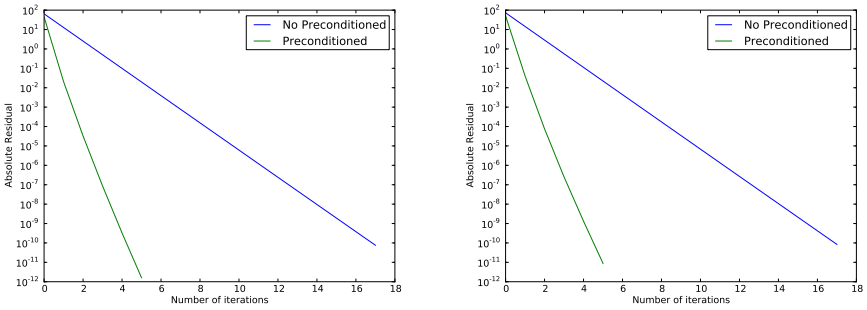

FIG. 7. *Convergence history of the first time step of the classical the the preconditioned algorithm,* $\Delta t = 0.01$, $\Delta x = 1/128$, $\Delta y = 1/8$, $N = 2$ *(left),* $N = 32$ *(right).*

TABLE 6
Number of iterations of the first time step and total computation time in seconds of the classical algorithm and the preconditioned algorithm, $T = 0.5$, $\Delta t = 0.01$, $\Delta x = 1/128$, $\Delta y = 1/8$.

N	2	4	8	16	32
N_{nopc}, $m = 7$	17	17	17	17	17
N_{pc}, $m = 5$	5	5	5	5	5
T^{ref}	16.1				
T_{nopc}	142.7	75.3	40.1	23.9	12.1
T_{pc}	91.3	43.3	22.8	13.1	7.3

Next, we consider the mesh $\Delta x = 1/2048, \Delta y = 1/64$. The convergence history and the computation time are presented in Figure 8 and Table 7. The parameters m used are also presented in Table 7. We can see that the preconditioned algorithm is more robust and requires much less number of iterations. However the computation time of the two algorithms are not quite scalable since:

1. for the classical algorithm, the number of iterations increases with the number of subdomains,

2. for the preconditioned algorithm, the size of preconditioner is $(2N - 2) \times N_T \times N_y$. This increases with the number of subdomains N. Thus, the application of preconditioner takes more computation time with bigger N.

TABLE 7
Computation time in seconds of the classical algorithm and the preconditioned algorithm, $\Delta t = 0.01, \Delta x = 1/2048, \Delta y = 1/64$.

N		256	512
Classical algorithm	m	8	12
	T_{nopc}	417.9	344.9
Preconditioned algorithm	m	5	6
	T_{pc}	259.9	268.7

We finish this sub section by some numerical tests for a time dependent potential $V = 5(x^2 + y^2)(1 + \cos(4\pi t))$. We could get similar conclusion by the results shown in table 8.

In conclusion, the preconditioner allows to reduce significantly the number of iterations and the computation time.

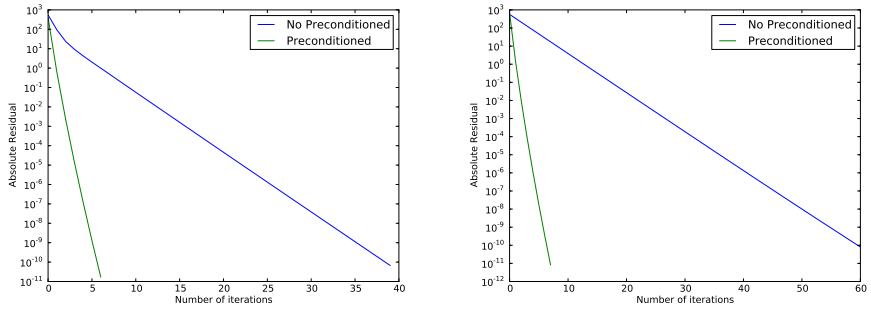

FIG. 8. *Convergence history of the first time step of the classical algorithm and the preconditioned algorithm,* $\Delta t = 0.01$, $\Delta x = 1/2048$, $\Delta y = 1/64$, $N = 256$ *(left),* $N = 512$ *(right).*

TABLE 8
Number of iterations of the first time step and total computation time in seconds of the classical algorithm and the preconditioned algorithm, $T = 0.5$, $\Delta t = 0.01$, $\Delta x = 1/128$, $\Delta y = 1/8$.

N	2	4	8	16	32
N_{nopc}, $p = -10$	31	31	31	31	31
N_{pc}, $p = -10$	9	9	9	9	9
T^{ref}	263.2				
T_{nopc}	272.6	138.5	72.4	40.5	19.5
T_{pc}	205.6	101.4	52.2	29.4	14.8

7. Conclusion. We applied the SWR method and the DDS method to the two dimensional linear Schrödinger equation. We proposed a new algorithm if the potential $V = 0$ and a preconditioned algorithm for a general linear potential, which could reduce the number of iterations and the computation time compared with the classical one. According to the numerical tests, the preconditioned algorithm is not sensitive to the transmission conditions (Robin, S_{2p}^m) and the parameters in these conditions.

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

REFERENCES

[1] XAVIER ANTOINE, WEIZHU BAO, AND CHRISTOPHE BESSE, *Computational methods for the dynamics of the nonlinear Schrödinger/Gross–Pitaevskii equations*, Comput. Phys. Commun., 184 (2013), pp. 2621 – 2633.

[2] XAVIER ANTOINE, CHRISTOPHE BESSE, AND PAULINE KLEIN, *Absorbing Boundary Conditions for the Two-Dimensional Schrödinger Equation With an Exterior Potential Part I: Construction and a Priori Estimates*, Math. Model. Methods Appl. Sci., 22 (2012).

[3] ———, *Absorbing boundary conditions for the two-dimensional Schrödinger equation with an exterior potential. Part II: Discretization and numerical results*, Numer. Math., 125 (2013), pp. 191–223.

[4] X. ANTOINE, E. LORIN, AND A.D. BANDRAUK, *Domain decomposition method and high-order absorbing boundary conditions for the numerical simulation of the time dependent schrödinger equation with ionization and recombination by intense electric field*, J. Sci. Comput., (2014), pp. 1–27.

[5] SATISH BALAY, MARK˜F. ADAMS, JED BROWN, PETER BRUNE, KRIS BUSCHELMAN, VICTOR EIJKHOUT, WILLIAM˜D. GROPP, DINESH KAUSHIK, MATTHEW˜G. KNEPLEY, LOIS CURF-

man McInnes, Karl Rupp, Barry˜F. Smith, and Hong Zhang, *PETSc Users Manual*, Tech. Report ANL-95/11 - Revision 3.4, Argonne National Laboratory, 2013.

[6] Weizhu Bao, Daniel Marahrens, Qinglin Tang, and Yanzhi Zhang, *A Simple and Efficient Numerical Method for Computing the Dynamics of Rotating Bose–Einstein Condensates via Rotating Lagrangian Coordinates*, SIAM J. Sci. Comput., 35 (2013).

[7] Christophe Besse and Feng Xing, *Schwarz waveform relaxation method for one dimensional Schrödinger equation with general potential*, Preprint, arXiv: 1503.02564, (2015).

[8] Yassine Boubendir, Xavier Antoine, and Christophe Geuzaine, *A quasi-optimal non-overlapping domain decomposition algorithm for the Helmholtz equation*, J. Comput. Phys., 231 (2012), pp. 262–280.

[9] Xiao-Chuan Cai, *Multiplicative Schwarz methods for parabolic problems*, SIAM J. Sci. Comput., (1994), pp. 1–18.

[10] Martin J Gander, *Optimized Schwarz Methods*, SIAM J. Numer. Anal., 44 (2006), pp. 699–731.

[11] ———, *Schwarz methods over the course of time*, Electron. Trans. Numer. Anal., 31 (2008), pp. 228–255.

[12] Martin J. Gander, Laurence Halpern, and Frédéric Nataf, *Optimal convergence for overlapping and non-overlapping Schwarz waveform relaxation*, Domain Decomposition Methods in Science and Engineering XI, (1999), pp. 27–36.

[13] ———, *Optimal Schwarz waveform relaxation for the one dimensional wave equation*, SIAM J. Numer. Anal., 41 (2003), pp. 1643–1681.

[14] Laurence Halpern and Jérémie Szeftel, *Optimized and quasi-optimal Schwarz waveform relaxation for the one dimensional Schrödinger equation*, Math. Model. Methods Appl. Sci., 20 (2010), pp. 2167–2199.

[15] Véronique Martin, *An optimized Schwarz waveform relaxation method for the unsteady convection diffusion equation in two dimensions*, Appl. Numer. Math., 52 (2005), pp. 401–428.

[16] Frédéric Nataf and Francois Rogier, *Factorization of the convection-diffusion operator and the Schwarz algorithm*, Math. Model. Methods Appl. Sci., 05 (1995), pp. 67–93.

[17] Yousef Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, 2nd ed., 2003.

[18] Yunhai Wu, Xiao-Chuan Cai, and David E Keyes, *Additive Schwarz methods for hyperbolic equations*, Contemp. Math., 218 (1998), pp. 468–476.