Journal of Computational and Applied Mathematics 383 (2021) 113018

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

New optimized Schwarz algorithms for one dimensional Schrödinger equation with general potential

F. Xing *

Laboratoire de Mathématiques J.A. Dieudonné, UMR 7351 CNRS, University Nice Sophia Antipolis, Parc Valrose 06108 Nice Cedex 02, France
Team COFFEE, INRIA Sophia Antipolis Méditerranée, Parc Valrose 06108 Nice Cedex 02, France
BRGM Orléans, France

ARTICLE INFO

Article history:
Received 30 March 2016
Received in revised form 9 March 2020

Keywords:
Schrödinger equation
Optimized Schwarz method
Parallel algorithms

ABSTRACT

The aim of this paper is to develop new optimized Schwarz algorithms for the one dimensional Schrödinger equation with linear and nonlinear potential. The classical algorithm is an iterative process. In case of time-independent linear potential, we construct explicitly the interface problem and use direct LU method on the interface problem. The algorithm therefore turns to be a direct process. Thus, the algorithm is independent of transmission condition and the numerical computation is smaller. To our knowledge, this is the first time that the Schwarz algorithm is constructed as direct process. Concerning the case of time-dependent linear potential or nonlinear potential, we propose to use a pre-processed linear operator as preconditioner which leads to a preconditioned algorithm. Numerically, the convergence is also independent of the transmission condition. In addition, both of these new algorithms implemented in parallel cluster are robust, scalable up to 256 sub domains (MPI process) and take much less computation time than the classical one, especially for the nonlinear case.
© 2020 Elsevier B.V. All rights reserved.

1. Introduction

This paper deals with the optimized Schwarz method without overlap for the one dimensional Schrödinger equation defined on a bounded spatial domain $\Omega = (a_0, b_0)$, $a_0, b_0 \in \mathbb{R}$ and $t \in (0, T)$, which reads

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

where $\mathscr{L}$ is the Schrödinger operator, the initial value $u_0 \in L^2(\mathbb{R})$ and the general real potential $\mathscr{V}$ consists of a linear and a nonlinear part

$$\mathscr{V} = V(t, x) + f(t, x, u).$$

We complete the equation with homogeneous Neumann boundary conditions on the boundary $\partial_{\mathbf{n}} u(t, x) = 0, \ x = a_0, b_0$ where $\partial_{\mathbf{n}}$ denotes the normal directive, $\mathbf{n}$ being the outward unit vector on the boundary.

* Correspondence to: Laboratoire de Mathématiques J.A. Dieudonné, UMR 7351 CNRS, University Nice Sophia Antipolis, Parc Valrose 06108 Nice Cedex 02, France.
E-mail address: feng.xing@unice.fr.

https://doi.org/10.1016/j.cam.2020.113018
0377-0427/© 2020 Elsevier B.V. All rights reserved.

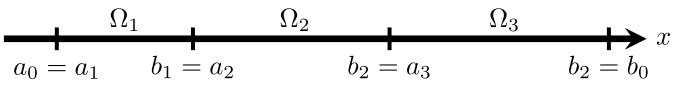


Fig. 1. Domain decomposition without overlap, $N = 3$.

Among the various domain decomposition methods, we focus our attention on the optimized Schwarz method [1], which has been widely studied over the past years for many different equations, like the Poisson equation [2], the Helmholtz equation [3,4] and the convection–diffusion equation [5].

In order to perform the optimized Schwarz method, Eq. (1) is first semi-discretized in time on the entire domain $(0, T) \times (a_0, b_0)$. The time interval $(0, T)$ is discretized uniformly with N_T intervals of length Δt. We denote by u_n (resp. V_n) an approximation of the solution u (resp. V) at time $t_n = n\Delta t$. The usual semi-discrete in time scheme developed by Durán and Sanz-Serna [6] applied to (1) reads as

$$i\frac{u_n - u_{n-1}}{\Delta t} + \partial_{xx}\frac{u_n + u_{n-1}}{2} + \frac{V_n + V_{n-1}}{2}\frac{u_n + u_{n-1}}{2} + f(t_{n+1/2}, x, \frac{u_n + u_{n-1}}{2})\frac{u_n + u_{n-1}}{2} = 0, \ \ 1 \leqslant n \leqslant N_T.$$

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

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

The original unknown u_n is recovered by $u_n = 2v_n - u_{n-1}$.

Eq. (2) is stationary for any $1 \leqslant n \leqslant N_T$. We can therefore apply the optimized Schwarz method. To simplify presentation, we consider optimized Schwarz method for the following equation in the next sections

$$\mathcal{L}_{\mathbf{x}} v := \frac{2i}{\Delta t} v + \partial_{xx} v + U(x)v + R(x, v)v = q(x). \tag{3}$$

Let us decompose the spatial domain Ω into N sub domains $\Omega_j = (a_j, b_j), j = 1, 2, \ldots, N$ without overlap as shown in Fig. 1 for $N = 3$. The classical Schwarz algorithm is an iterative process and we identify the iteration number thanks to label k. We denote by v_j^k the solution on sub domain Ω_j at iteration $k = 0, 1, 2, \ldots$ of the Schwarz algorithm. The optimized Schwarz algorithm for (3) consists in applying the following sequence of iterations for $j = 1, 2, \ldots, N$

$$\begin{cases} \mathcal{L}_{\mathbf{x}} v_j^k = q(x), \ \ x \in \Omega_j, \\ \partial_{\mathbf{n}_j} v_j^k(a_j) + S_j v_j^k(a_j) = \partial_{\mathbf{n}_j} v_{j-1}^{k-1}(a_j) + S_j v_{j-1}^{k-1}(a_j), \\ \partial_{\mathbf{n}_j} v_j^k(b_j) + S_j v_j^k(b_j) = \partial_{\mathbf{n}_j} v_{j+1}^{k-1}(b_j) + S_j v_{j+1}^{k-1}(b_j), \end{cases} \tag{4}$$

with a special treatment for the two extreme sub domains Ω_1 and Ω_N since the boundary conditions are imposed on $\{x = a_1\}$ and $\{x = b_N\}$

$$\partial_{\mathbf{n}_1} v_1^k(a_1) = 0, \quad \partial_{\mathbf{n}_N} v_N^k(b_N) = 0,$$

where S_j is the transmission operator.

The transmission condition is an important issue for this method. In recent years, many works contribute to look for good transmission conditions to obtain faster convergence in the context of the Schwarz waveform relaxation method (SWR) and the optimized Schwarz method for the Schrödinger equation [7–11]. The authors of [7] consider the Robin transmission condition and the optimal transmission condition. The well-posedness and the convergence of the SWR algorithms are shown.

$$\text{Robin} : S_j v = -ip \cdot v, \ \ p \in \mathbb{R}^+ \text{ is parameter.} \tag{5}$$

The authors of [8–10] introduce the idea using some absorbing boundary operators [12,13] as the transmission operator. For example,

$$\begin{aligned} S_{\text{pade}}^m : \ S_j v &= -i\sqrt{\frac{2i}{\Delta t} + U + R(x, v)}\, v, \\ &\approx \Big(-i\sum_{s=0}^{m} a_s^m + i\sum_{s=1}^{m} a_s^m d_s^m \big(\frac{2i}{\Delta t} + U + R(x, v) + d_s^m\big)^{-1}\Big) v, \end{aligned} \tag{6}$$

where $a_s^m = e^{i\theta/2}/\big(m\cos^2(\frac{(2s-1)\pi}{4m})\big)$, $d_s^m = e^{i\theta}\tan^2(\frac{(2s-1)\pi}{4m})$, $s = 0, 1, \ldots, m$ are constant coefficients associated with the Padé approximation of order m. These operators are constructed by using some pseudo differential techniques and

could be seen as different approximate accesses to the optimal operator. However, numerically there are always some parameters in these transmission conditions that need to be optimized for each potential $\mathscr{V}$, each size of time step Δt, each mesh and each number of sub domains N. This leads to huge extra tests before launching the simulation.

The parallel scalability is one of the key objectives of this method. The authors of [9] introduced some new efficient and robust algorithms in the context of the Schwarz waveform relaxation algorithm for the one dimensional Schrödinger equation, based on the idea of constructing explicitly the interface problem. Then, the new algorithms are extended to the two dimensional linear case in the context of the optimized Schwarz algorithm [10]. These new algorithms are much more scalable and efficient than the classical one. However, choosing the transmission condition and the parameters in the transmission condition remain very inconvenient.

The objective of this article is to develop scalable optimized Schwarz algorithms on parallel computers. We propose some new algorithms based on the studies of the classical algorithm. These new algorithms have following main features theoretically or numerically: the convergence is independent of the transmission condition, the size of time step Δt, the mesh and the number of sub domains. In addition, the computation time is scalable with number of process and much smaller than the classical algorithm.

This paper is organized as follows. In Section 2, we study the classical algorithm and construct the interface problem. In Sections 3 and 4, we present the new algorithm for the Schrödinger equation with time-independent linear potential and the preconditioned algorithm for the Schrödinger equation with general potential. Some numerical results are shown in Section 5. Finally, we draw a conclusion.

Remark 1. To simplify the presentation, we only consider the Robin transmission condition (5) in the following paper. Applying other more complex transmission conditions in [7–9] such as (6) is direct, however they are not useful in our new algorithms.

2. Classical optimized Schwarz algorithm

Let us introduce the fluxes ℓ_j^k and $r_j^k, j = 1, 2, \ldots, N$ defined as

$$\ell_j^k = \partial_{\mathbf{n}_j} v_j^k(a_j) + S_j v_j^k(a_j), \quad r_j^k = \partial_{\mathbf{n}_j} v_j^k(b_j) + S_j v_j^k(b_j),$$

with a special definition for the two extreme sub domains: $\ell_1^k = r_N^k = 0$. Thus, the Algorithm (4) can be split into local problems on sub domain $\Omega_j, j = 1, 2, \ldots, N$

$$\begin{cases} \mathscr{L}_{\mathbf{x}} v_j^k = q(x), \\ \partial_{\mathbf{n}_j} v_j^k(a_j) + S_j v_j^k(a_j) = \ell_j^k, \\ \partial_{\mathbf{n}_j} v_j^k(b_j) + S_j v_j^k(b_j) = r_j^k, \end{cases} \tag{7}$$

and flux problems

$$\begin{cases} \ell_j^{k+1} = -r_{j-1}^k + 2S_j v_{j-1}^k(b_{j-1}), \ j = 2, 3, \ldots, N, \\ r_j^{k+1} = -\ell_{j+1}^k + 2S_j v_{j+1}^k(a_{j+1}), \ j = 1, 2, \ldots, N-1. \end{cases} \tag{8}$$

The spatial approximation is realized by the standard $\mathbb{P}_1$ finite element method. If $f = 0$ in (1), i.e. $R = 0$ in (3), then the system (7) is linear. Let us denote by $\mathbf{v}_{j,n}^k$ (resp. $\mathbf{u}_j^k$) the nodal interpolation vector of v_j^k (resp. u_j^k) with N_j nodes, $\mathbb{M}_j$ the mass matrix, $\mathbb{S}_j$ the stiffness matrix, $\mathbb{M}_{j,W_n}$ the generalized mass matrix with respect to $\int_{a_j}^{b_j} U(x)\, v\, \phi\, dx$. Then, the matrix formulation of the N local problems is given by

$$\left(\mathbb{A}_j + ip \cdot \mathbb{M}^{\Gamma_j}\right)\mathbf{v}_j^k = \frac{2i}{\Delta t}\mathbb{M}_j \mathbf{u}_{j,n-1}^k - \mathbb{M}^{\Gamma_j} Q_j{}^{\top} \begin{pmatrix} \ell_j^k \\ r_j^k \end{pmatrix}, \tag{9}$$

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

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

If $R \neq 0$, then Eq. (7) is nonlinear. Assuming that $u_{j,n-1}$ is known, the computation of $v_{j,n}^k$ is accomplished by a fixed point procedure. We take $\zeta_j^0 = u_{j,n-1}$ and compute the solution $v_{j,n}^k$ as the limit of the iterative procedure with respect

to the label q, $q = 1, 2, \ldots$

$$\begin{cases} \Big(\dfrac{2i}{\Delta t} + \partial_{xx} + U\Big)\zeta_j^q = \dfrac{2i}{\Delta t}u_{j,n-1} - R(x, \zeta_j^{q-1})\zeta_j^{q-1}, \\ \partial_{\mathbf{n}}\zeta_j^q - ip\cdot\zeta_j^q = \ell_{j,n}^k,\ x = a_j, \\ \partial_{\mathbf{n}}\zeta_j^q - ip\cdot\zeta_j^q = r_{j,n}^k,\ x = b_j. \end{cases} \tag{10}$$

Let us denote by $\boldsymbol{\zeta}_j^q$ the nodal interpolation of ζ_j^q and $\mathbf{b}_{R(x,\zeta_j^q)}$ the vector with respect to $\int_{a_j}^{b_j} R(x, \zeta_j^q)\zeta_j^q\phi dx$. Thus, the matrix formulation of (10) is

$$\Big(\mathbb{A}_{j,n} + ip\cdot\mathbb{M}^{\Gamma_j}\Big)\boldsymbol{\zeta}_j^q = \frac{2i}{\Delta t}\mathbb{M}_j\mathbf{u}_{j,n-1}^k - \mathbf{b}_{R(x,\zeta_j^{q-1})} - Q_j^\top\begin{pmatrix} l_n^k \\ r_n^k \end{pmatrix}. \tag{11}$$

In addition, the discrete formulation of the flux problems (8) reads

$$\begin{cases} \ell_{j,n}^{k+1} = -r_{j-1,n}^k - 2ip\cdot Q_{j-1,r}\mathbf{v}_{j-1,n}^k,\ j = 1, 2, \ldots, N-1, \\ r_{j,n}^{k+1} = -\ell_{j+1,n}^k - 2ip\cdot Q_{j+1,l}\mathbf{v}_{j+1,n}^k,\ j = 2, 3, \ldots, N. \end{cases} \tag{12}$$

where $Q_{j,l} = (1, 0, \ldots, 0, 0) \in \mathbb{C}^{N_j}$, $Q_{j,r} = (0, 0, \ldots, 0, 1) \in \mathbb{C}^{N_j}$.

The classical algorithm is initialized by an initial guess of ℓ_j^0 and r_j^0, $j = 1, 2, \ldots, N$. The boundary conditions for any sub domain Ω_j at iteration $k+1$ involve the knowledge of the values of the functions on adjacent sub domains Ω_{j-1} and Ω_{j+1} at prior iteration k. Thanks to the initial guess, we can *solve* the Schrödinger equation on each sub domain, allowing to build the new boundary conditions for the next step, *communicating* them to other sub domains. This procedure is summarized in (13) for $N = 3$ sub domains at iteration $k+1$.

$$\begin{pmatrix} r_1^k \\ \ell_2^k \\ r_2^k \\ \ell_3^k \end{pmatrix} \xrightarrow{Solve} \begin{pmatrix} \mathbf{v}_1^k \\ \mathbf{v}_2^k \\ \mathbf{v}_3^k \end{pmatrix} \to \begin{pmatrix} -r_1^k - 2ip\cdot Q_{1,r}\mathbf{v}_1^k \\ -\ell_2^k - 2ip\cdot Q_{2,l}\mathbf{v}_j^k \\ -r_2^k - 2ip\cdot Q_{2,r}\mathbf{v}_j^k \\ -\ell_3^k - 2ip\cdot Q_{3,l}\mathbf{v}_N^k \end{pmatrix} \xrightarrow{Comm.} \begin{pmatrix} r_1^{k+1} \\ \ell_2^{k+1} \\ r_2^{k+1} \\ \ell_3^{k+1} \end{pmatrix}. \tag{13}$$

Let us define the discrete interface vector by

$$\mathbf{g}_n^k = (r_{1,n}^k, \ldots, \ell_{j,n}^k, r_{j,n}^k, \ldots, l_{N,n}^k)^\top.$$

Thanks to this definition, we give a new interpretation to the algorithm which can be written as

$$\mathbf{g}^{k+1} = \mathcal{R}_h\mathbf{g}^k = \mathbf{g}^k - (I - \mathcal{R}_h)\mathbf{g}^k. \tag{14}$$

where I is identity operator and $\mathcal{R}_h$ is an operator. The solution to this iteration process is given as the solution to the discrete interface problem

$$(I - \mathcal{R}_h)\mathbf{g} = 0.$$

3. New algorithm for time-independent linear potential

The first aim of this section is to show that if the potential is linear $f = 0$, i.e. $R = 0$, then the discrete interface problem can be written as

$$(I - \mathcal{L}_{h,n})\mathbf{g}_n = \mathbf{d}_n, \tag{15}$$

where $\mathbf{d}_n$ is a vector and $\mathcal{L}_{h,n} \in \mathbb{C}^{(2N-2)\times(2N-2)}$ is a matrix (the notation "MPI j" above the columns of the matrix will be used after in this section)

$$\mathcal{L}_{h,n} = \begin{pmatrix} \overbrace{}^{\text{MPI } 0} & \overbrace{}^{\text{MPI } 1} & \overbrace{}^{\text{MPI } 2} & & \overbrace{}^{\text{MPI } N-2} & \overbrace{}^{\text{MPI } N-1} \\ & x_n^{2,1} \quad x_n^{2,2} & & & & \\ x_n^{1,4} & & & & & \\ & & x_n^{3,1} \quad x_n^{3,2} & & & \\ & x_n^{2,3} \quad x_n^{2,4} & & & & \\ & & & \cdots & & \\ & & x_n^{3,3} \quad x_n^{3,4} & & & \\ & & & & x_n^{N-1,1} \quad x_n^{N-1,2} & \\ & & & \cdots & & \\ & & & & & x_n^{N,1} \\ & & & & x_n^{N-1,3} \quad x_n^{N-1,4} & \end{pmatrix}. \tag{16}$$

Especially, if the linear potential is time-independent $V = V(x)$, then

$$\mathcal{L}_{h,1} = \mathcal{L}_{h,2} = \cdots = \mathcal{L}_{h,N_T}. \tag{17}$$

Secondly, based on these observations, we propose a new Schwarz algorithm for the Schrödinger equation with time-independent linear potential.

Proposition 1. *Assuming that $f = 0$, then the operator $\mathcal{R}_{h,n}$ is linear*

$$\mathcal{R}_{h,n}\mathbf{g}_n^k = \mathcal{L}_{h,n}\mathbf{g}_n^k + \mathbf{d}_n$$

where $\mathcal{L}_{h,n}$ is a matrix as shown by (16). In addition, if the linear potential is time-independent $V = V(x)$, then the interface matrix $\mathcal{L}_{h,n}$ satisfies (17).

Proof. Firstly, by some straight forward calculations using (9) and (12), we can verify that

$$\begin{aligned}
x_n^{j,1} &= -I - 2ip \cdot Q_{j,l}(\mathbb{A}_{j,n} + ip \cdot \mathbb{M}^{\Gamma_j})^{-1}\mathbb{M}^{\Gamma_j}Q_{j,l}^{\top},\\
x_n^{j,2} &= -2ip \cdot Q_{j,l}(\mathbb{A}_{j,n} + ip \cdot \mathbb{M}^{\Gamma_j})^{-1}\mathbb{M}^{\Gamma_j}Q_{j,r}^{\top},\\
x_n^{j,3} &= -2ip \cdot Q_{j,r}(\mathbb{A}_{j,n} + ip \cdot \mathbb{M}^{\Gamma_j})^{-1}\mathbb{M}^{\Gamma_j}Q_{j,l}^{\top},\\
x_n^{j,4} &= -I - 2ip \cdot Q_{j,r}(\mathbb{A}_{j,n} + ip \cdot \mathbb{M}^{\Gamma_j})^{-1}\mathbb{M}^{\Gamma_j}Q_{j,r}^{\top},
\end{aligned} \tag{18}$$

and $\mathbf{d}_n = (d_{n,1,r}, \ldots, d_{n,j,l}, d_{n,j,r}, \ldots, d_{n,N,r})^{\top}$ with

$$\begin{aligned}
d_{n,j,l} &= 2ip \cdot \frac{2i}{\Delta t} \cdot Q_{j-1,r}(\mathbb{A}_{j-1,n} + ip \cdot \mathbb{M}^{\Gamma_{j-1}})^{-1}\mathbb{M}_{j-1}\mathbf{u}_{j-1,n},\ j = 2, 3, \ldots, N,\\
d_{n,j,r} &= 2ip \cdot \frac{2i}{\Delta t} \cdot Q_{j+1,l}(\mathbb{A}_{j+1,n} + ip \cdot \mathbb{M}^{\Gamma_{j+1}})^{-1}\mathbb{M}_{j+1}\mathbf{u}_{j+1,n},\ j = 1, 2, \ldots, N-1.
\end{aligned} \tag{19}$$

Secondly, since $V = V(x)$, then for $j = 1, 2, \ldots, N$, we have

$$\begin{aligned}
&\mathbb{M}_{j,W_1} = \mathbb{M}_{j,W_2} = \cdots = \mathbb{M}_{j,W_{N_T}},\\
&\Rightarrow \mathbb{A}_{j,1} = \mathbb{A}_{j,2} = \cdots = \mathbb{A}_{j,N_T} = \frac{2i}{\Delta t}\mathbb{M}_j - \mathbb{S}_j + \mathbb{M}_{j,W_n}.
\end{aligned}$$

Thus, the elements of $\mathcal{L}_{h,n}$ satisfy

$$x_1^{j,s} = x_2^{j,s} = \cdots = x_{N_T}^{j,s},$$

for $j = 1, 2, \ldots, N$, $s = 1, 2, 3, 4$. □

Thanks to the analysis yielded in the previous proposition, we can build explicitly $\mathcal{L}_{n,h}$ with not much computation. According to (18), to know the elements $x^{j,1}$ and $x^{j,3}$ (resp. $x^{j,2}$ and $x^{j,4}$), it is necessary to compute one time the application of $(\mathbb{A}_{j,n} + ip \cdot \mathbb{M}^{\Gamma_j})^{-1}$ to vector $\mathbb{M}^{\Gamma_j}Q_{j,l}^{\top}$ (resp. $\mathbb{M}^{\Gamma_j}Q_{j,r}^{\top}$). In total, to know the four elements $x_n^{j,1}$, $x_n^{j,2}$, $x_n^{j,3}$ and $x_n^{j,4}$, it is enough to compute two times the application of $(\mathbb{A}_{j,n}+ip\cdot\mathbb{M}^{\Gamma_j})^{-1}$ to vector. In other words, this amounts to solve the Schrödinger equation on each sub domain two times to build the matrix $\mathcal{L}_{h,n}$. In addition, let us note that $\mathcal{L}_{h,1} = \mathcal{L}_{h,2} = \cdots = \mathcal{L}_{h,N_T}$ if $V = V(x)$.

The construction of the vector $\mathbf{d}_n$ is similar. According to (19), one needs to apply $(\mathbb{A}_{j,n} + ip \cdot \mathbb{M}^{\Gamma_j})^{-1}$ to vector $\mathbb{M}_j\mathbf{u}_{j,n}$ for $j = 1, 2, \ldots, N$. In other words, we solve the Schrödinger equation on each sub domain one time. The vector is stored in a distributed manner using the PETSc library [14]

$$\mathbf{d}_n = \big(\underbrace{d_{n,1,r}}_{\text{MPI 0}}, \underbrace{d_{n,2,l}, d_{n,2,r}}_{\text{MPI 1}}, \ldots, \underbrace{d_{n,j,l}, d_{n,j,r}}_{\text{MPI j-1}}, \ldots, \underbrace{d_{n,N,l}}_{\text{MPI N-1}}\big)^{\top}.$$

Let us now present the new algorithm for the Schrödinger equation with time-independent linear potential.

Algorithm 1: New algorithm, time-independent linear potential

1: Build the interface matrix $\mathcal{L}_{h,n}$ explicitly.
2: The initial datum is u_0.
for $n = 1, 2, \ldots, N_T$ **do**
  2.1: Build the vector $\mathbf{d}_n$ on time step n,
  2.2: Solve the linear system

$$(I - \mathcal{L}_{h,n})\mathbf{g}_n = \mathbf{d}_n. \tag{20}$$

  2.3: Solve the Schrödinger equation on time step n on each sub domain using the flux from step 2.2 to compute u_n.

Concerning the computational phase 2.2 in the new algorithm and the storage of the matrix $\mathcal{L}_{h,n}$, there are some choices.

Choice 1. The transpose of $\mathcal{L}_{h,n}$ is stored in a distributed manner using the library PETSc. As shown by (18), the first column of $\mathcal{L}_{h,n}$ lies in MPI process 0. The second and third columns are in MPI process 1, and so on for other processes. The linear system (20) is solved by any iterative methods, such as the fixed point method and the Krylov methods (GMRES, BiCGStab) [15]. Note that if the fixed point method is used, then formally we recover the classical algorithm (14).

Choice 2. The transpose of $\mathcal{L}_{h,n}$ is stored in a distributed manner using the library PETSc. Unlike the first choice, we solve the linear system (20) by a parallel LU direct method (MPI). One obvious benefit is that the algorithm has no convergence problem. Thus, it is possible to use any $p \in \mathbb{R}^+$ in the Robin transmission condition without looking for the optimal one which makes the algorithm converge with less iterations for each size of time step, each mesh and each number of sub domains.

The size of $\mathcal{L}_{h,n}$ is $2N-2$, which is small for a LU direct method even when N is large, such as $N = 256$. Solving the linear system needs few arithmetic operations. However, two points are not negligible. It is well known that the LU direct method is inherently sequential. In addition, one MPI process contains only two rows of $\mathcal{L}_{h,n}^{\top}$ (four elements). Thus, the parallel LU direct method could suffer considerable communication costs.

Choice 3. The matrix $\mathcal{L}_{h,n}$ is stored locally in one MPI process after construction. For $n = 1, 2, \ldots, N_T$, the distributed vector $\mathbf{d}_n$ is firstly gathered to one MPI process, then the linear system (20) is solved by a sequential LU direct method. Finally, the solution $\mathbf{g}_n$ is broadcasted to all MPI processes. This choice is a variation of the second one. It can avoid communications from the parallel implementation of the LU method.

Remark 2. In the new algorithm with the choice 2 or 3, it is enough to do one time the LU factorization of $I - \mathcal{L}_{h,n}$.

There are two main novelties in the context of the optimized Schwarz method for the one dimensional Schrödinger equation. Firstly, we construct explicitly the matrix $\mathcal{L}_{h,n}$, while in the classical algorithm, $\mathcal{L}_{h,n}$ remains an abstract operator. Secondly, we use the direct method on the interface problem, thus the algorithm turns to be a direct process. It is independent of the transmission condition since the convergence properties of $I - \mathcal{L}_{h,h}$ have no influence on direct linear solver.

4. Preconditioned algorithm for general potential

In the previous sections, we have interpreted the classical algorithm for the Schrödinger equation as (14). However, if $f \neq 0$, then the operator $\mathcal{R}_{h,n}$ is nonlinear. If $f = 0$ and $V = V(t, x)$, it is necessary to construct the interface matrix $\mathcal{L}_{h,n}$ for each time step n. Thus, the new algorithm is not suitable here. Instead, to reduce the number of iterations required for convergence, inspired from [9], we add a preconditioner P^{-1} (P is a non singular matrix) in (14) or (15) which leads to the preconditioned algorithm:

- for $\mathscr{V} = V(t, x)$,

$$\mathbf{g}_n^{k+1} = \mathbf{g}_n^k - P^{-1}(I - \mathcal{R}_{h,n})\mathbf{g}_n^k, \tag{21}$$

$$P^{-1}(I - \mathcal{L}_{h,n})\mathbf{g}_n = P^{-1}\mathbf{d}_n, \tag{22}$$

- for $\mathscr{V} = V(t, x) + f(t, x, u)$,

$$\mathbf{g}_n^{k+1} = \mathbf{g}_n^k - P^{-1}(I - \mathcal{R}_{h,n})\mathbf{g}_n^k. \tag{23}$$

We now turn to explain which preconditioner is used. The interface problem for the free Schrödinger equation ($V = 0$, $f = 0$) is

$$(I - \mathcal{L}_0)\mathbf{g}_n = \mathbf{d}_n,$$

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

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

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

1. Intuitively, the Schrödinger operator without potential is a rough approximation of the Schrödinger operator with potential:

$$i\partial_t u + \partial_{xx} u \approx i\partial_t u + \partial_{xx} u + Vu + f(t, x, u)u.$$

In other words, $Vu + f(t, x, u)u$ is a perturbation of the free Schrödinger operator.
If $\mathscr{V} = 0$, then $I - \mathcal{L}_0 = I - \mathcal{L}_{h,n} = I - (\mathcal{R}_{h,n} - \mathcal{R}_{h,n} \cdot \mathbf{0})$ where $\mathbf{0}$ is the zero vector. Thus, the matrix $\mathcal{L}_0$ could be a good approximation of the matrix $\mathcal{L}_{h,n}$ and of the nonlinear operator $\mathcal{R}_{h,n} - \mathcal{R}_{h,n} \cdot \mathbf{0}$:

$$P = I - \mathcal{L}_0 \approx I - \mathcal{L}_{h,n}, \quad P = I - \mathcal{L}_0 \approx I - (\mathcal{R}_{h,n} - \mathcal{R}_{h,n} \cdot \mathbf{0}), \tag{24}$$

2. The matrix $\mathcal{L}_0$ can be constructed easily as explained in the previous section.

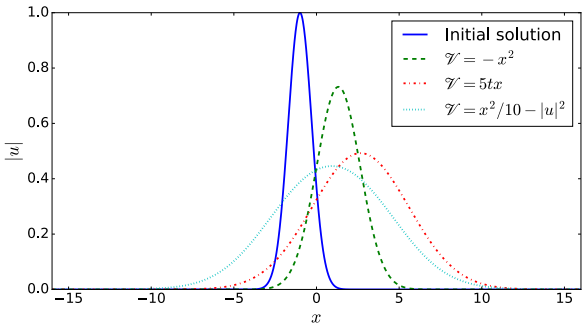


Fig. 2. Initial solution $|u_0|$ and numerical solutions of the Schödinger equation with different potentials at final time $T = 1.0$. The time step is $\Delta t = 0.001$. The meshes are $\Delta x = 1.0 \times 10^{-5}$ for $\mathscr{V} = -x^2$ and $\Delta x = 5.0 \times 10^{-5}$ for the others.

3. The implementation of the preconditioner and the storage of P are same as that of (20), which are discussed in the previous section by the three choices. In effect, for any vector y, the vector $z := P^{-1}y$ is computed by solving the linear system

$$Pz = (I - \mathcal{L}_0)z = y, \tag{25}$$

as the inverse of a matrix numerically is too expensive. Note that using the choice 2 or 3 ensures that the application of the preconditioner is robust.

As shown in the previous section, if the complete domain is decomposed into N sub domains, then the size of $I - \mathcal{R}_{h,n}$ and $P^{-1}(I - \mathcal{R}_{h,n})$ is $2N - 2$, which is small enough to compute all its eigenvalues numerically. We will present the numerical results in the following section.

5. Numerical results

The physical domain $(a_0, b_0) = (-16, 16)$ is decomposed into N equal sub domains without overlap. The final time is fixed to be $T = 1.0$. The initial data is

$$u_0(x) = e^{-(x+1)^2 + i(x+1)}.$$

Let us consider three different potentials in the following three subsections

- time-independent linear potential: $\mathscr{V} = -x^2$,
- time-dependent linear potential: $\mathscr{V} = 5tx$,
- nonlinear potential: $\mathscr{V} = \frac{x^2}{10} - |u|^2$,

which give rise to solutions that propagate to the right side and undergoes dispersion (see Fig. 2). This first subsection devotes to the comparison of the classical algorithm and the new algorithm. In Section 5.2, we mainly study the spectral properties of the classical algorithm and the preconditioned algorithms, i.e. the eigenvalues of the operators $I - \mathcal{L}_{h,n}$ and $P^{-1}(I - \mathcal{L}_{h,n})$. The last one compares the classical algorithm and the preconditioned algorithm for the general nonlinear potential.

All the numerical tests are implemented in a CPU cluster (16 cores/node, Intel Sandy Bridge E5-2670, 64GB memory/node). We fix always one core per MPI process, one MPI process per sub domain and 16 MPI processes per node. The communications are handled by OpenMPI 1.6.5 (GCC 4.8). The linear systems (9) and (11) related to the finite element method are solved by the LU direct method using the MKL Pardiso library. The convergence condition is $\|\mathbf{g}_n^{k+1} - \mathbf{g}_n^k\| < 10^{-10}$.

Remark 3. Without loss of generality, we consider the number of iterations of the first time step. In other words, we study the number of iterations of the evolution between $t_0 \to t_1$ to compute v^1 with the optimized Schwarz method.

Remark 4. For the initial vector g^0, we have two choices: $g^0 = 0$ or g^0 is random vector. To study the computation time, we use zero initial vector since the algorithm converges faster. While to study the convergence, we choose random vector. In [16], the authors claim that the numerical conclusions with regard to convergence and the number of iterations could be wrong with zero initial vector.

Remark 5. The theoretical optimal parameter p in the Robin transmission condition is not at hand for us. We then seek for the best parameter numerically for each case if necessary.

Table 1
Computation time (seconds) of the classical algorithm and the new algorithm with the fixed point method, two Krylov methods (GMRES and BiCGStab) and the LU direct method (sequential implementation and parallel implementation) where $\mathscr{V} = -x^2$, $\Delta t = 0.001$, $\Delta x = 10^{-5}$ and $p = 45$.

	N	2	4	8	16	32	64	128	256
	T_{ref}	594.0							
Fixed Point	T_{cls}	11 102.0	5909.4	2948.7	1596.5	788.9	412.7	199.3	105.4
	T_{new}	730.0	374.7	185.1	101.5	49.0	26.1	13.4	7.9
GMRES	T_{cls}	1461.5	934.5	611.4	423.3	220.0	116.4	55.2	28.1
	T_{new}	728.8	375.5	185.6	101.9	48.9	25.7	12.7	7.0
BiCGStab	T_{cls}	1807.2	1161.5	795.9	489.4	235.3	115.8	50.7	31.2
	T_{new}	727.7	375.4	185.3	101.4	48.8	25.5	12.7	6.9
LU-S	T_{new}	663.3	363.6	183.9	101.0	48.4	25.0	12.1	6.3
LU-P	T_{new}	680.8	356.8	189.3	102.9	51.3	33.6	31.4	65.2

LU-S: LU direct method (sequential implementation) in MKL Pardiso library.
LU-P: LU direct method (parallel implementation, MPI) in MUMPS library [17].

Table 2
Number of iterations with the iterative methods used on the interface problem for some different p where $\mathscr{V} = -x^2$, $\Delta t = 0.001$ and $\Delta x = 10^{-5}$.

p	$N = 2$			$N = 256$		
	Fixed point	GMRES	BiCGStab	Fixed point	GMRES	BiCGStab
5	159	3	3	185	15	9
10	83	3	3	96	15	9
15	58	3	3	67	14	8
20	45	3	3	52	14	8
25	39	3	3	44	13	8
30	35	3	3	40	13	8
35	33	3	3	37	13	8
40	32	3	3	36	13	7
45	31	3	3	35	13	7
50	32	3	3	35	13	7

5.1. Time-independent linear potential

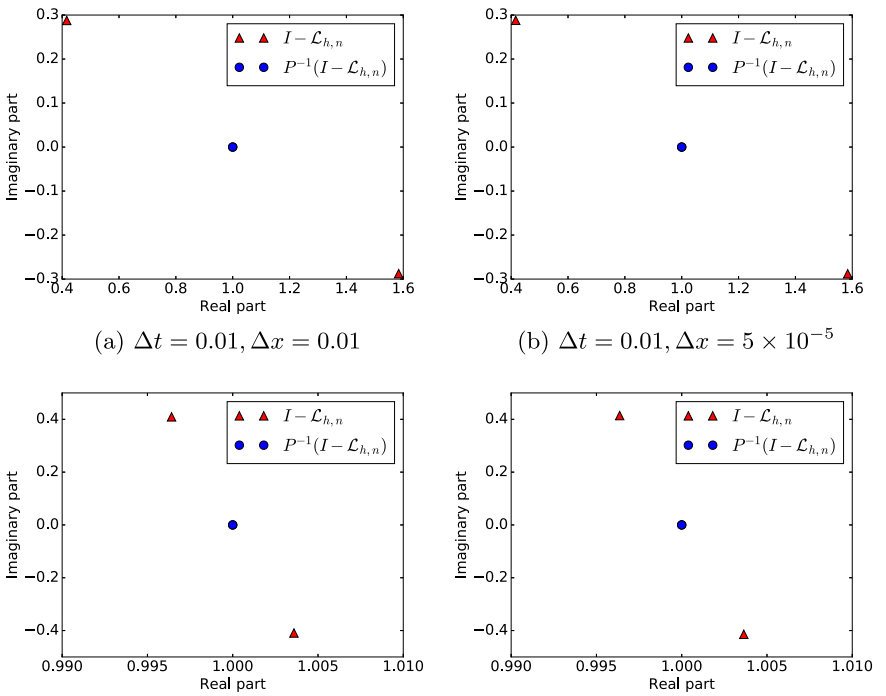
In this part, we are interested in observing the efficiency of our new algorithms for the time-independent linear potential $\mathscr{V} = -x^2$. We denote by T_{ref} the computation time to solve the Schrödinger equation numerically on a single processor on the entire domain and T_{cls} (resp. T_{new}) the computation time of the classical (resp. new) algorithm for N sub domains. We test the algorithms for $N = 2, 4, 8, 16, 32, 64, 128, 256$ sub domains.

The computation times are shown in Table 1 where the fixed point method, two Krylov methods (GMRES and BiCGStab) and the LU direct method (sequential implementation and parallel implementation) are used on the interface problem (Step 2.2 in the Algorithm 1). The time step and the mesh are $\Delta t = 0.001$, $\Delta x = 10^{-5}$. Roughly speaking, in Table 1 we have

$$T_{\text{cls}} = T_{\text{sub}} \times N_{\text{iter}} \times N_T + \cdots,$$
$$T_{\text{new}} = T_{\text{sub}} \times 2 + (T_{\text{sub}} \times 2 + T_{Ld}) \times N_T + \cdots,$$

where T_{sub} denotes the computation time for solving the equation on one sub domain for one time step, 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. Firstly we can see that T_{new} with the parallel LU method takes much more time than others if N is large. As has been explained in Section 3 (Choice 2), since the LU method is inherently sequential and the size of (20) is quite small, the communication is costly in our case. From now on, we abandon our idea using a parallel LU direct solver on the interface problem. For other choices, if the number of sub domains N is not so large, then $T_{\text{sub}} \gg T_{Ld}$ and the minimum of N_{iter} is 3 in all our tests (see Table 2). If the number of sub domains N is large, then $T_{Ld} \sim T_{\text{sub}}$ and $N_{\text{iter}} \geqslant 3$. It is for this reason that the new algorithm takes less computation time and we do not observe big difference when different solvers are used on the interface problem in the new algorithm. However, we emphasize that the new algorithm with the sequential LU method on the interface problem is the best choice. It allows to use any $p \in \mathbb{R}^+$. In other words, the algorithm is independent of the transmission condition. Note that it is possible to use other more complex transmission condition as presented in [7–9]. Normally, this leads to better convergence properties of the classical algorithms. However, the new algorithm with the direct method on the interface problem is a direct algorithm.

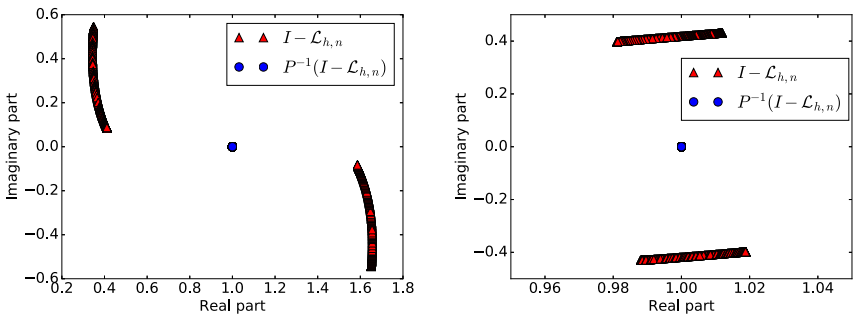
In conclusion, the new algorithm with the sequential LU direct method is robust, scalable and independent of the transmission condition. In addition, it takes less computation time than the classical algorithm.



(a) $\Delta t = 0.01, \Delta x = 0.01$ (b) $\Delta t = 0.01, \Delta x = 5 \times 10^{-5}$

(c) $\Delta t = 0.001, \Delta x = 0.01$ (d) $\Delta t = 0.001, \Delta x = 5 \times 10^{-5}$

Fig. 3. Eigenvalues of $I - \mathcal{L}_{h,n}$ and $P^{-1}(I - \mathcal{L}_{h,n})$ for $N = 2$ where $n = 1$ and $p = 45$.



(a) $\Delta t = 0.01, \Delta x = 5 \times 10^{-5}$ (b) $\Delta t = 0.001, \Delta x = 5 \times 10^{-5}$

Fig. 4. Eigenvalues of $I - \mathcal{L}_{h,n}$ and $P^{-1}(I - \mathcal{L}_{h,n})$ for $N = 256$ where $n = 1$ and $p = 45$.

5.2. Time-dependent linear potential

In this subsection, we consider the preconditioned algorithm and the classical algorithm for the time-dependent potential $\mathcal{V} = 5tx$. According to our numerical investigations in the previous subsection, the preconditioner (25) is implemented using the choice 3 in Section 3, which is robust and time efficient.

Firstly, we are interested in the efficiency of the preconditioner. The explicit construction of the interface problem allows us to study the spectral properties of $I - \mathcal{L}_{h,n}$ and $P^{-1}(I - \mathcal{L}_{h,n})$. Figs. 3 and 4 present their eigenvalues for some different meshes where $N = 2$ and $N = 256$ respectively. Without loss of generality, we only consider $n = 1$. It can be seen that the eigenvalues of the preconditioned linear system are really close to 1+0i, which illustrates that P^{-1} is a good preconditioner.

Next, we show in Figs. 5 and 6 the eigenvalues of $I - \mathcal{L}_{h,n}$ and $P^{-1}(I - \mathcal{L}_{h,n})$ for some different p and for $N = 2, 256$ respectively. It can be seen that the parameter p has almost no influence on $P^{-1}(I - \mathcal{L}_{h,n})$. As for $I - \mathcal{L}_{h,n}$, the distance between the eigenvalues and the value 1+0i depends on p. Thus, if the fixed point method is used on the interface problem, then it is essential to choose the optimal parameter p in the classical algorithm. However, as shown in Tables 3 and 4

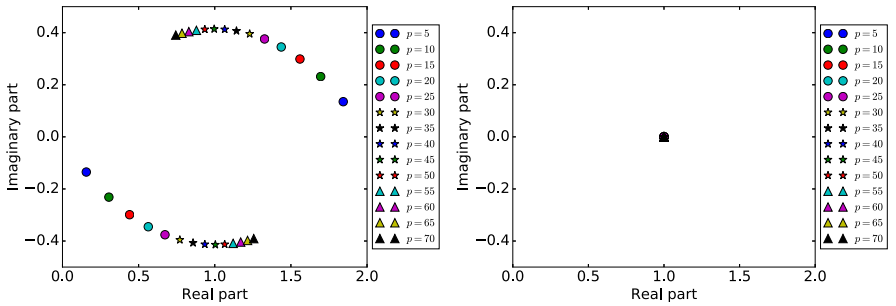


Fig. 5. Eigenvalues of $I - \mathcal{L}_{h,n}$ (left) and $P^{-1}(I - \mathcal{L}_{h,n})$ (right) for different p where $N = 2$ and $\Delta t = 0.001$, $\Delta x = 5 \times 10^{-5}$.

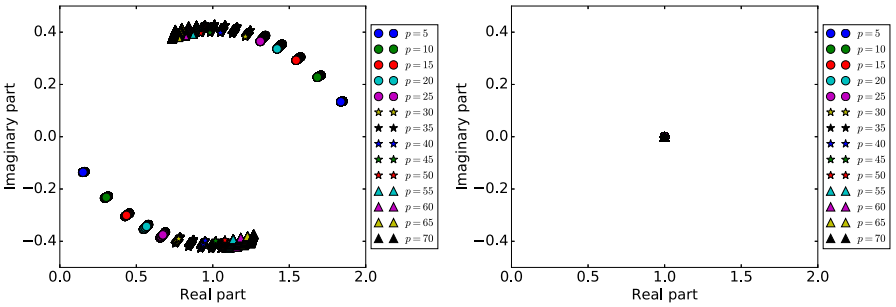


Fig. 6. Eigenvalues of $I - \mathcal{L}_{h,n}$ (left) and $P^{-1}(I - \mathcal{L}_{h,n})$ (right) for different p where $N = 256$ and $\Delta t = 0.001$, $\Delta x = 5 \times 10^{-5}$.

Table 3
Number of iterations of the classical algorithm and the preconditioned algorithm (+PC) for $N = 2$ where $\mathscr{V} = 5tx$, $\Delta t = 0.001$ and $\Delta x = 5 \times 10^{-5}$.

p	5	10	15	20	25	30	35	40	45	50
Fixed point	158	82	58	45	39	35	33	32	31	32
Fixed point + PC	3	3	3	3	3	3	3	3	3	3
GMRES	3	3	3	3	3	3	3	3	3	3
GMRES+PC	3	3	3	3	3	3	3	3	3	3
BiCGStab	3	3	3	3	3	3	3	3	3	3
BiCGStab + PC	2	2	2	2	2	2	2	2	2	2

Table 4
Number of iterations of the classical algorithm and the preconditioned algorithm (+PC) for $N = 256$ where $\mathscr{V} = 5tx$, $\Delta t = 0.001$ and $\Delta x = 5 \times 10^{-5}$.

p	5	10	15	20	25	30	35	40	45	50
Fixed point	184	95	66	52	44	40	37	36	35	35
Fixed point + PC	4	4	4	4	4	3	4	4	4	4
GMRES	14	12	13	12	12	12	12	12	12	12
GMRES+PC	4	4	3	4	4	4	4	4	4	4
BiCGStab	8	8	7	7	7	7	7	7	7	7
BiCGStab + PC	3	3	3	3	3	3	3	3	3	2

(number of iterations of the classical algorithm and the preconditioned algorithm for $N = 2$ and $N = 256$), the use of the Krylov methods (GMRES and BiCGStab) can also reduce the dependency on the parameter p in the transmission condition.

Finally, Table 5 presents the computation times of the classical algorithm and the preconditioned algorithm where the fixed point method, the GMRES method and the BiCGStab method are used on the interface problem. The computation time of the preconditioned algorithm T_{pc} consists of three major parts: the construction of the preconditioner (denoted by T_1), the application of $\mathcal{R}_{h,n}$ to vectors (denoted by T_2), and the application of preconditioner (denoted by T_3). Then we have

$$T_{\text{pc}} \approx T_1 + (T_2 + T_3) \times N_{\text{iter}} \times N_T, \tag{26}$$

Table 5
Computation time (seconds) of the classical algorithm and the preconditioned algorithm (+PC) where $\mathscr{V} = 5tx$, $\Delta t = 0.001$ and $\Delta x = 5 \times 10^{-5}$.

N	2	4	8	16	32	64	128	256
T_{ref}	2100.4							
Fixed point	3438.5	1732.8	840.0	413.9	193.7	80.1	45.9	24.4
Fixed point+PC	1400.3	692.6	332.1	159.7	71.6	28.8	16.5	9.9
GMRES	1250.0	712.5	376.8	188.0	85.3	34.2	19.3	10.4
GMRES+PC	1246.1	664.9	331.9	157.7	70.6	28.4	16.2	10.0
BiCGStab	1235.1	710.8	345.8	179.2	81.1	36.9	17.7	10.7
BiCGStab+PC	1326.7	595.0	287.7	147.9	68.1	31.1	15.2	10.8

Table 6
Computation time (seconds) of the classical algorithm and the preconditioned algorithm where $\mathscr{V} = x^2/10 - |u|^2$, $\Delta t = 0.001$, $\Delta x = 5 \times 10^{-5}$ and $p = 45$.

N	2	4	8	16	32	64	128	256
T_{ref}	2423.9							
T_{cls}	11 152.7	6147.9	3074.4	1615.2	767.0	416.6	202.8	98.0
T_{pc}	2593.4	1362.3	648.3	318.2	153.7	81.0	39.1	18.7

where $T_2 \gg T_3$, $T_1 \approx 2T_2$. As can be seen from Tables 3 and 4, if the fixed point method is used on the interface problem, the preconditioner allows to reduce significantly the number of iterations required for convergence. Thus, $T_{\text{pc}} < T_{\text{cls}}$.

Remark 6. Let us read Tables 4 and 5 together for $N = 256$ and $p = 45$ with the fixed point method used on the interface problem. The reduction of computation time is not proportional to that of number of iterations. In fact, the number of iterations of the preconditioned algorithm is about 10 times less than that of the classical one. However, its computation time is only reduced by about 2.5 times. There are three reasons:

- For each time step n, the linear systems (9) or (11) are solved multiple times. However, it is enough to do one time the LU factorization of the matrix $\mathbb{A}_{j,n} + ip \cdot \mathbb{M}^{\Gamma_j}$, of which the computation times are same for the classical algorithm and the preconditioned algorithm.
- As explained in Remark 4, the initial vector used in Table 4 is a random vector, while it is the zero vector in Table 5.
- The implementation of the preconditioner takes a little computation time.

Before turning to consider the Schrödinger equation with a general nonlinear potential, we can conclude that P^{-1} is a very efficient preconditioner. In addition, the use of Krylov methods on the interface problem allows to reduce both of the number of iterations and the computation time in the framework of the classical algorithm.

5.3. Nonlinear potential

The objective of the last subsection is to investigate the performance of the preconditioned algorithm for the nonlinear potential $\mathscr{V} = x^2/10 - |u|^2$. Unlike the linear case, the interface problem (14) is nonlinear here. Thus, it is not possible to use Krylov methods to accelerate the convergence. Table 6 presents the computation time of the classical algorithm (T_{cls}) and the preconditioned algorithm (T_{pc}) for $N = 2, 4, 8, 16, 32, 64, 128, 256$ where the time step is $\Delta t = 0.001$, the mesh is $\Delta x = 5 \times 10^{-5}$ and $p = 45$. We can see that both of the algorithms are scalable and the preconditioner can significantly reduce the computation time.

We show in Table 7 the number of iterations of the classical algorithm (N_{cls}) and the preconditioned algorithm (N_{pc}) with different parameter p in the transmission condition for $N = 2$ and $N = 256$. As the linear case, the preconditioned algorithm is almost independent of the parameter p. In addition, the preconditioned algorithm needs much less iterations to converge. Note that it is for the same reasons as the linear case that the reduction of computation time is not proportional to that of number of iterations (see Remark 6).

In conclusion, both of the classical algorithm and the preconditioned algorithm are robust. The preconditioned algorithm takes less number of iterations and less computation times than the classical one. In addition, the preconditioned algorithm is not sensible to the transmission condition.

6. Conclusion

We proposed in this paper a new optimized Schwarz algorithm for the one dimensional Schrödinger equation with time-independent linear potential and a preconditioned algorithm for the Schrödinger equation with general potential. These algorithms are robust and scalable. They could reduce significantly both the number of iterations and

Table 7
Number of iterations of the classical algorithm and the preconditioned algorithm for some different p where $\mathscr{V} = x^2/10 - |u|^2$, $\Delta t = 0.001$ and $\Delta x = 5 \times 10^{-5}$.

	$N = 2$		$N = 256$	
p	N_{cls}	N_{pc}	N_{cls}	N_{pc}
5	147	3	170	3
10	79	3	90	3
15	55	3	63	4
20	44	3	50	4
25	38	3	43	4
30	34	3	39	4
35	32	3	36	4
40	31	3	35	4
45	30	3	34	4
50	31	3	34	4

the computation time. In addition, by using the sequential LU direct method on the interface problem in the context of the new algorithm, the convergence is independent of the transmission condition.

The theoretical analysis of the preconditioned system (22) remains an interesting problem for future work. This could allow us to better understand the performance of the preconditioner theoretically.

Acknowledgments

This work was granted access to the HPC and visualization resources of “Centre de Calcul Interactif" hosted by University Nice Sophia Antipolis, France.

References

[1] M.J. Gander, Optimized Schwarz methods, SIAM J. Numer. Anal. 44 (2) (2006) 699–731.
[2] P.-L. Lions, On the Schwarz alternating method. III: a variant for nonoverlapping subdomains, in: Third Int. Symp. Domain Decompos. Methods Partial Differ. Equations, Vol. 6, 1990, pp. 202–223.
[3] Y. Boubendir, X. Antoine, C. Geuzaine, A quasi-optimal non-overlapping domain decomposition algorithm for the Helmholtz equation, J. Comput. Phys. 231 (2) (2012) 262–280.
[4] M.J. Gander, F. Magoules, F. Nataf, Optimized Schwarz methods without overlap for the Helmholtz equation, SIAM J. Sci. Comput. 24 (1) (2002) 38–60.
[5] F. Nataf, F. Rogier, Factorization of the convection-diffusion operator and the Schwarz algorithm, Math. Model. Methods Appl. Sci. 05 (01) (1995) 67–93.
[6] 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.
[7] 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.
[8] X. Antoine, E. Lorin, A. 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) 1–27.
[9] C. Besse, F. Xing, Schwarz waveform relaxation method for one dimensional Schrödinger equation with general potential, Numer. Algorithms 74 (2) (2017) 393–426.
[10] C. Besse, F. Xing, Domain decomposition algorithms for two dimensional linear Schrödinger equation, J. Sci. Comput. 72 (2) (2017) 735–760.
[11] C. Besse, F. Xing, Domain decomposition algorithms for the two dimensional nonlinear Schrödinger equation and simulation of Bose–Einstein condensates, SMAI J. Comput. Math. 2 (2016) 277–300.
[12] 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.
[13] X. Antoine, C. Besse, P. Klein, Absorbing boundary conditions for general nonlinear Schrödinger equations, SIAM J. Sci. Comput. 33 (2) (2011) 1008–1033.
[14] 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.
[15] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., Society for Industrial and Applied Mathematics, 2003.
[16] M.J. Gander, Schwarz methods over the course of time, Electron. Trans. Numer. Anal. 31 (2008) 228–255.
[17] MUMPS library, http://mumps.enseeiht.fr/.