Stable, Gloablly Non-iterative, Non-overlapping Domain Decomposition Parallel Solvers for Parabolic Problems *

Yu Zhuang † Xian-He Sun ‡

Abstract

In this paper, we report a class of stabilized explicit-implicit domain decomposition (SEIDD) methods for the parallel solution of parabolic equations, based on the explicit-implicit domain decomposition (EIDD) methods. EIDD methods are globally non-iterative, non-overlapping domain decomposition methods which, when compared with Schwarz alternating algorithm based parabolic solvers, are computationally and communicationally efficient for each simulation time step but suffer from time step size restrictions due to conditional stability or conditional consistency. By adding a stabilization step to the EIDD methods, the SEIDD methods are freed from time step size restrictions while retaining EIDD's computational and communicational efficiency for each time step, rendering them excellent candidates for large-scale parallel simulations. Three algorithms of the SEIDD type are implemented, which are experimentally tested to show excellent stability, computation and communication efficiencies, and high parallel speedup and scalability.

1. Introduction. In this paper, we report a class of stabilized explicit-implicit domain decomposition (SEIDD) methods for the parallel simulation of time dependent systems governed by the parabolic equation

$$\begin{cases} \frac{\partial}{\partial t}u(t,x,y) = Au, & (x,y)\in\Omega, \quad t \geq 0 \\ u(t,x,y) = b(t,x,y), & (x,y)\in\partial\Omega, \ t \geq 0 \\ u(0,x,y) = u_o(x,y), & (x,y)\in\Omega, \end{cases} \tag{1}$$

where $\Omega \subset \mathbf{R}^2$ is the spatial problem domain, A is a spatial elliptic operator given by $A = \nabla^2 + a(x,y)\frac{\partial}{\partial x} + b(x,y)\frac{\partial}{\partial y} + c(x,y)$. We allow $c(x,y)$ to have positive values and hence the spatial elliptic operator A might be indefinite.

There exists a substantial literature in domain decomposition methods [5]. The Schwarz alternating algorithms [4, 10, 11, 12, 13, 20] are the most intensively studied methods in the last two decades. Schwarz methods are globally iterative methods for elliptic equations. Here the term "globally" refers to the solution process for the problem over the entire domain as opposed to solution processes for subdomain problems which could be either iterative or direct. For parabolic problems, when implicit temporal discretization schemes are employed, an elliptic equation must be solved for each time step. Thus the Schwarz algorithm can be used for the parallel solution of parabolic equations by solving the elliptic equation in parallel. Cai [2, 3] used the Schwarz methods in this manner to solve parabolic problems. Since the Schwarz methods are solvers for the time independent elliptic equations, it has an advantage in preserving the unconditional stability of the implicit temporal discretization as along as the Schwarz solver is iterated until the solution error becomes small enough to have no influence on stability.

In the parallel implementation of domain decomposition algorithms, when different subdomains are assigned to different processors, globally iterative methods incur repeated data transmission among processors. Since communication is more time consuming per byte than

*This research was supported in part by NSF under NSF grant CCR-9972251 and by ONR under PET/Logicon

†Department of Computer Science, Texas Tech University, Lubbock, Texas 79409, zhuang@cs.ttu.edu. The paper was written when the author was visiting the Computer Science Department of Illinois Institute of Technology.

‡Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, sun@cs.iit.edu

computation, it is appealing to keep the global iterations to a small number. In 1988, Kuznetsov [14] proposed a one-iteration overlapping Schwarz algorithm for the approximate solution of the elliptic equation obtained from an implicit temporal discretization, based on a property of the Green's function of the temporal discretization. Kuznetsov's solver has good stability but requires an overlap size of $O(\sqrt{\Delta t}\log\epsilon)$ for an local error tolerance of $O(\epsilon)$. A method of lesser overlap size requirement was developed by Mathew, Polyakov, Russo and Wang [19] in 1998 using the multi-dimensional ADI method [8] combined with domain decomposition based operator splitting. But the stability of their algorithm deteriorates when the number of subdomains goes above two.

For domain decomposition methods, non-overlapping algorithms are obviously more appealing if all other algorithmic conditions are the same. In 1991, Dryja [9] proposed a globally non-iterative, non-overlapping domain decomposition based finite element solver for the elliptic equation obtained from Crank-Nicolson temporal discretization. His solver is a fractional step method from the viewpoint of temporal discretization, and a one-iteration multiplicative Schwarz method from the viewpoint of elliptic solver. Dryja's method is unconditionally stable but has a very low global accuracy of order $O(\sqrt{\Delta t})+O(h)$. A similar non-iterative, non-overlapping method was developed by Laevsky and Rudenko [18] in 1995 which achieves unconditional convergence and a global accuracy $O(\Delta t)+O(h)$, an improvement over Dryja's method in temporal accuracy but still low in spatial accuracy. Another non-iterative, non-overlapping method of Laevsky [15] uses a domain decomposition based ADI-type method to solve the elliptic equation resulting from the Crank-Nicolson scheme with only one iteration. The global error of this algorithm is of order $O(h+(\Delta t)^2+\rho(1+\rho)^{1/2})$ with $\rho=(\Delta t)^2/h$. However the term $\rho(1+\rho)^{1/2}$ has placed a restriction on the ratio of Δt and h. Furthermore, his method becomes only conditionally stable when number of subdomains is larger than two.

One reason for the low accuracy of the aforementioned non-iterative, non-overlapping algorithms is the lack of an adequately accurate interface boundary conditions. Explicit-implicit algorithms have solved the problem of the availability of interior boundary conditions for non-iterative, non-overlapping methods. In 1988, Kuznetsov [14] proposed an explicit-implicit scheme using nonoverlapping domain decomposition. The interior boundary conditions are first predicted using an explicit method. Then a stable implicit temporal scheme can be applied to the equation on the subdomains and the resulted elliptic equation on each subdomain can be solved independently, and thus in parallel. When the forward Euler scheme is used to predict the interface boundary condition and the backward Euler scheme is used for the equation on the subdomains, the explicit-implicit method can be represented as

$$u^{n+1}=(I-\Delta tA_2)^{-1}(I+\Delta tA_1)u^n, \tag{2}$$

where A_1 is the restriction of discretized spatial operator A_h on the interface boundary, and A_2 the restriction of A_h on subdomains. This method has a global error of $O(\Delta t)+O(h^2)$. However, the explicit predictor of the interface boundary condition $(I+\Delta tA_1)$ in (2) causes numerical instability unless the time step size is restricted to $\Delta t=O(h^2)$. To reduce the numerical instability, Dawson, Du and Dupont [7] factorize the fully explicit predictor in (2) into a partially explicit and partially implicit predictor with the explicit prediction carried out *only* across the interface boundaries, resulting in a method mathematically representable as

$$u^{n+1}=(I-\Delta tA_2)^{-1}\underline{(I-\Delta tA_1^y)^{-1}(I+\Delta tA_1^x)}u^n, \tag{3}$$

where the underlined is the partially explicit predictor. Their method retains the same order of accuracy and achieves a better numerical stability. However it is still not unconditionally stable

due to the explicit part in the predictor (see testing results in Tables 1 and 2). A penalized explicit-implicit algorithm proposed by Black [1] has achieved numerically verified unconditional stability. However it has an error term of order $O(\frac{\Delta t}{h})^2$, making the algorithm inconsistent unless paying a price of restricting time step size to an order of $O(h^2)$ to attain a first order temporal accuracy, a restriction quantitatively similar to, though qualitatively different from, the stability related time step size restriction.

To eliminate time step size restrictions of the explicit-implicit domain decomposition (EIDD) algorithms, we introduced a class of stabilized explicit-implicit domain decomposition (SEIDD) algorithms [22] by adding a stabilization step to the EIDD methods. In this paper we report a generic parallel SEIDD method with three specific algorithms of this SEIDD type. The proposed SEIDD algorithms not only are freed from time step size restrictions, but more importantly for parallel computing, the stabilization is proven to add *zero* communication cost and very low computation cost to EIDD algorithms, yielding excellent parallel speedup and scalability confirmed by testings on an SGI Origin 2000 computer.

2. The Stabilized Explicit-implicit Domain Decomposition Method. The domain Ω is divided into p subdomains Ω_1, Ω_2, $\cdots$, Ω_p with interface boundaries denoted by B. The complement of the interface boundary B is denoted by B^c and hence B^c is the union of all subdomains, i.e. $B^c = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_p$. In this paper we assume that the interface boundaries

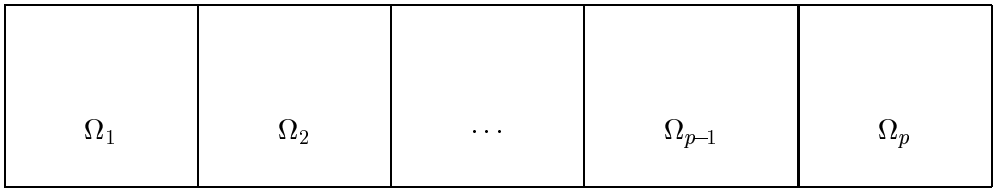

Figure 1

do not cross into each other in the interior of the domain as in Figure 1 and Figure 2. The treatment of more general domain partition schemes that allow cross-over of interface boundaries will be investigated in a future project, and in this paper we choose to focus on the stabilization technique for EIDD methods on such-partitioned domains.

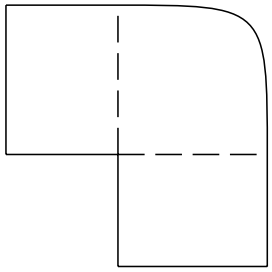

Figure 2

With the decomposition of the original non-discrete domain Ω, we define the partitioning of the discrete domain Ω_h simply by inheriting the partitioning of the original domain:

$$\begin{cases} \Omega_{h,i} = \Omega_h \cap \Omega_i \quad \text{for } \ i = 1,\ 2,\ \cdots,\ p, \\ B_h = \Omega_h \cap B. \end{cases}$$

We denote the interface boundary between subdomains Ω_i and Ω_j by $B_{i,j}$ for $i < j$ ($B_{i,j}$ could be a empty set), and denote the i-th processor by p_i. Now a generic parallel domain decomposition algorithm (SEIDD) for computing the solution u_h^{n+1} at the $(n+1)$-th time step from the current n-th time step is given below.

The Parallel SEIDD Algorithm

0. Assign subdomain Ω_i and interface boundary $B_{i,j}$ to p_i.
1. Compute u_h^{n+1} on B_h using an explicit scheme. Then pass from p_i to p_j the newly predicted u_h^{n+1} on $B_{i,j}$.
2. Compute u_h^{n+1} on the subdomains B_h^c using any unconditionally stable scheme using exterior boundary conditions and the interface boundary conditions computed at step 1. Then pass part of the solution of u_h^{n+1} on Ω_j near $B_{i,j}$ from p_j to p_i.
3. Throw away the interface boundary condition computed at step 1, and bring back u^n on B_h. Then implicitly re-compute u_h^{n+1} on B_h, using solution data u_h^{n+1} on nearby subdomain as boundary conditions. Go back to step 1 for the next time step iteration.

In step 1 of the algorithm, we do not pass the value of any u^n from p_j to p_i before the explicit computation of the interface boundary condition u^{n+1}. The computation of the interface boundary condition does need those data. However since processor p_i already received them from p_j at step 2 of the previous time step and no update has been performed on the data on the subdomains, no new data transfer is necessary. On the other hand, for any EIDD method the two data transfer operations in the first two steps are necessary for predicting the interface boundary condition and for implicitly computing the solution on the neighboring subdomain assigned to another processor.

Furthermore, using any method to compute interface boundary condition needs solution data from nearby. This incurs one data transferring operation. And to compute the solution near the interface boundary also incurs one data transferring operation. Thus, the number of data transferring operations for each time step by any method is at least two. Therefore, we have arrived at the following conclusion concerning the communication cost of the parallel SEIDD methods.

- The stabilization (step 3) of parallel SEIDD adds **zero** communication cost to the EIDD method.
- the parallel SEIDD methods is optimal in terms of number of data transferring operations for each time step.

Since each of the two data transferring operations are carried out by $p-1$ processors simultaneously with almost equal load, the total communication time of each time step satisfies

$$T_{\text{comm}} \leq 2\alpha \frac{N_B}{p-1}, \tag{4}$$

if the communication time is proportional to the amount of data transferred, where α is some system dependent data transfer rate, and N_B denotes the number of grid points on the interface boundaries.

The computation cost of the stabilization depends on the number of grid points on interface boundaries. Since the number of grid points on interface boundaries is much smaller than the total number of grid points, the computation overhead is therefore also very small. Let f_1, f_2 and f_3 denote the computation complexity functions for the predictor, the subdomain solver and the stabilizer respectively. Then the total computation cost for each time step is the sum of

the prediction time $f_1\left(\frac{N_B}{p-1}\right)$, the subdomain solver time $f_2\left(\frac{N-N_B}{p}\right)$ and the stabilization time $f_3\left(\frac{N_B}{p-1}\right)$, yielding

$$T_{\text{comp}} = f_1\left(\tfrac{N_B}{p-1}\right) + f_2\left(\tfrac{N-N_B}{p}\right) + f_3\left(\tfrac{N_B}{p-1}\right), \tag{5}$$

Then using (4) and (5), the parallel speedup, defined as single processor execution time T_1 over parallel execution time T_p, can be estimated to be at least

$$S_p \;=\; \frac{p\ f_2(N)}{f_2(N)+\frac{p}{p-1}[2\alpha N_B+f_2(N_B)]}$$

when assuming $f_2(n) \geq f_3(n) \geq f_1(n)$. And the corresponding parallel efficiency, defined as speedup over number of processors, is

$$E_p \;=\; \frac{f_2(N)}{f_2(N)+\frac{p}{p-1}[2\alpha N_B+f_2(N_B)]}.$$

The SEIDD method given before is generic and allows many choices for the explicit predictor in step 1, the subdomain scheme in step 2, and the stabilizer in step 3. In the following, we give three specific algorithms of the SEIDD types.

The SEIDD1 method. When we use the forward Euler scheme for the predictor and use the backward Euler scheme for the subdomain problems and the stabilizer, we obtain the stabilized Kuznetsov's method

$$u_h^{n+1} = (I - \Delta t A_{h,1})^{-1}\left[\chi_{B_h} + \chi_{B_h^c}\,(I-\Delta t A_{h,2})^{-1}(I+\Delta t A_{h,1})\right] u_h^n, \tag{6}$$

where $A_{h,1}$ is the restriction of the discretized spatial operator A_h on the interface boundaries, $A_{h,2}$ is the restriction of A_h on the subdomains, and χ_S is a diagonal matrix with 1 on entries corresponding to grid points in the set S and zero elsewhere.

It is proven in [21, 22] that when the spatial operator A is self-adjoint (or symmetric in the case that A is a matrix), the SEIDD1 algorithm is unconditionally von Neumann stable and unconditionally convergent in the sense that the convergence requires no restriction on the ratio of time step size to the spatial mesh size.

The SEIDD2 method. To illustrate the design of a stabilizer for a given predictor and to examine the effectiveness of the stabilization technique of the SEIDD method, we choose to stabilize the algorithm of Dawson, Du and Dupont [7]. The predictor they used is $(I-\Delta t A^y_{h,1})^{-1}(I+\Delta t A^x_{h,1})$. To stabilize the predictor we employ the following stabilizer $(I-\Delta t A^x_{h,1})^{-1}(I-\Delta t A^y_{h,1})^{-1}$. Then we arrive at the following stabilized Dawson-Du-Dupont algorithm

$$u^{n+1} = (I-\Delta t A^x_{h,1})^{-1}(I-\Delta t A^y_{h,1})^{-1}\left[\chi_{B_h}+\chi_{B_h^c}\,(I-\Delta t A_{h,2})^{-1}(I-\Delta t A^y_{h,1})^{-1}(I+\Delta t A^x_{h,1})\right] u^n. \tag{7}$$

The SEIDD3 method. The subdomain problems of the two SEIDD algorithms given above involve solving an elliptic equation of the form $(I - \Delta t A)u = r$, which is the major source of computation cost of the two algorithms. When A is nonsymmetric, this computation cost is not low. However the backward Euler scheme used for the subdomain problems can be factorized directionally to reduce the computation cost to linear order $O(N)$ on a grid with N grid points. E.g., for 2-D problems, replacing the subdomain scheme of SEIDD1 $(I-\Delta t A_{h,2})^{-1}$ by $(I-\Delta t A^x_{h,2})^{-1}(I-\Delta t A^y_{h,2})^{-1}$, we arrive at

$$u^{n+1} = (I-\Delta t A_{h,1})^{-1}\left[\chi_{B_h} + \chi_{B_h^c}\,\underline{(I-\Delta t A^x_{h,2})^{-1}(I-\Delta t A^y_{h,2})^{-1}}(I+\Delta t A_{h,1})\right] u_h^n, \tag{8}$$

where the underlined is the subdomain scheme. For a two dimensional problem, the discretized directional components of A are usually tridiagonal matrices. Thus, $(I-\Delta t A_h^x)$ and $(I-\Delta t A_h^y)$ can be easily inverted with a computation cost of linear order.

3. Experimental Results. We have chosen a set of parabolic problems with known solutions, and solved them on an NCSA Origin 2000 machine with a total of 256 nodes, each of 250 MHz, running IRIX 6.5.9 operating system.

Table 1: $u_t = \Delta u$ with $u = e^{-2t}\cos(x+y)$

Δt	1/50	1/100	1/200	1/400	1/800
SEIDD1	$4.2e-03$	$8.9e-04$	$1.5e-04$	$5.3e-05$	$3.1e-05$
SEIDD2	$3.8e-04$	$2.2e-04$	$1.2e-04$	$6.8e-05$	$3.6e-05$
SEIDD3	$4.8e-03$	$1.1e-03$	$2.2e-04$	$3.7e-05$	$1.4e-05$
BEuler	$6.1e-04$	$3.0e-04$	$1.5e-04$	$7.6e-05$	$3.8e-05$
EIDD1	∞	∞	∞	∞	∞
EIDD2	$3.9e+54$	$2.4e+98$	∞	∞	∞
EIDD3	∞	∞	∞	∞	∞

The symbol ∞ denotes an error larger than $1.0e+100$.
The domain is $[0,3]\times[0,1]$ with $h = 1/64$, which is divided into 3 subdomains.

Table 2: $u_t = \Delta u + \sin(x)^2 u_x + [3-\sin(x)\cos(x)]u$ with $u = e^t \sin(x)\sin(y)$

Δt	1/25	1/50	1/100	1/200	1/400	1/800
SEIDD1	5.7e–02	2.8e–02	1.4e–02	7.1e–03	3.7e–03	2.0e–03
SEIDD2	5.7e–02	2.8e–02	1.4e–02	7.2e–03	3.7e–03	2.0e–03
SEIDD3	3.9e–01	1.9e–01	9.1e–02	4.5e–02	2.3e–02	1.1e–02
BEuler	5.7e–02	2.8e–02	1.4e–02	7.2e–03	3.7e–03	2.0e–03
EIDD1	4.4e+45	5.6e+94	∞	∞	∞	∞
EIDD2	8.6e+21	2.2e+42	1.6e+74	∞	∞	∞
EIDD3	2.6e+42	6.1e+88	∞	∞	∞	∞

The domain is $[0,2\pi]\times[0,\pi]$ with mesh size $h = \pi/128$, which is divided into 2 subdomains.

Tables 1 and 2 contain the maximal error of the numerical solutions at $t = 1$ computed using the indicated methods, where BEuler denotes the "most" stable backward Euler method. The EIDD methods listed are the corresponding SEIDD methods without the stabilization step. Thus, EIDD1 is Kunnetsov's method (2), and EIDD2 is the method of Dawson, Du and Dupont (3). It is well known that for an unconditionally stable method, the simulation error remains small even when the time step size Δt is large relative to the spatial mesh size. As indicated by the experimental results in Tables 1-2, the errors of the SEIDD algorithms remain relatively small when the time step size Δt is large, and they are almost as small as those of the backward Euler method, experimentally supporting the effectiveness of the stabilization of the SEIDD algorithms. The proof of the unconditional stability and convergence given in [21, 22] is built upon the self-adjointness (or symmetry) of the spatial operator. Though rigorously it is still open if the SEIDD algorithms remain unconditionally stable for problems with non-symmetric operators, the numerical experiment data in Table 1 and 2 (also see Table 4-5) show that the proposed algorithms are robust and retain the unconditional stability for non-selfadjoint problems.

Table 3 contains the testing data for the comparison of SEIDD methods versus EIDD methods in execution time for solving a heat equation on $[0,8\pi]\times[0,2\pi]$. The domain is divided into

Table 3: Execution Time: SEIDD methods vs. EIDD methods

Method	T_total	T_comp	T_comm	Max_Err	nprocs
SEIDD1	5.3e+01	5.24e+01	3.0e–01	2.1e–04	4
SEIDD2	5.4e+01	5.28e+01	3.0e–01	2.1e–04	4
SEIDD3	2.5e+01	2.44e+01	3.0e–01	1.3e–04	4
EIDD1	5.3e+01	5.22e+01	3.0e–01	2.1e–04	4
EIDD2	5.3e+01	5.25e+01	3.0e–01	2.1e–04	4
EIDD3	2.5e+01	2.46e+01	3.0e–01	1.3e–04	4

The equation is $u_t = u_{xx} + u_{yy}$ with $u = e^{-2t}\cos(x+y)$.
The domain is $[0, 8\pi]\times[0, 2\pi]$, which is divided into 4 subdomains.
Spatial mesh size $h = \pi/64$ with each subdomain of 128×128 grid points.
The simulation time interval is [0,1] with time step size of $\triangle t = 1/2000$.

four square subdomains of equal size, each assigned to a processor. In the test, we choose a sufficiently small time step size so that all EIDD method converge for the testing problem. From the testing data, no increased computation time beyond machine variation range was recorded, experimentally supporting the analysis in Section 2 that SEIDD methods incur very small amount of overhead by stabilizing EIDD methods.

Table 4: Solving $u_t = \Delta u + \sin(x)u_x - \cos(x)u$ with SEIDD1

Processors	T_total	T_comp	T_comm	Speedup	Efficiency	Max-Err
1	2.08e+01	2.08e+01	0.0e-02	1	100%	1.4e-03
2	2.12e+01	2.10e+01	1.1e-01	1.96	98.1%	1.4e-03
3	2.12e+01	2.10e+01	1.9e-01	2.94	98.1%	1.4e-03
4	2.12e+01	2.10e+01	1.5e-01	3.92	98.1%	1.4e-03
5	2.12e+01	2.10e+01	1.7e-01	4.91	98.1%	1.4e-03
6	2.13e+01	2.10e+01	2.5e-01	5.86	97.7%	1.4e-03
7	2.13e+01	2.11e+01	2.9e-01	6.84	97.7%	1.4e-03
8	2.13e+01	2.11e+01	2.2e-01	7.81	97.7%	1.4e-03
10	2.14e+01	2.11e+01	3.1e-01	9.72	97.2%	1.4e-03
12	2.14e+01	2.10e+01	4.1e-01	11.7	97.2%	1.4e-03
14	2.14e+01	2.10e+01	3.7e-01	13.6	97.2%	1.4e-03
16	2.14e+01	2.10e+01	3.8e-01	15.6	97.2%	1.4e-03
20	2.15e+01	2.10e+01	4.8e-01	19.3	96.7%	1.4e-03
24	2.15e+01	2.11e+01	4.3e-01	23.2	96.7%	1.4e-03
28	2.16e+01	2.11e+01	5.0e-01	27.0	96.3%	1.4e-03
32	2.16e+01	2.10e+01	5.6e-01	30.8	96.3%	1.4e-03
48	2.12e+01	2.09e+01	3.3e-01	47.1	98.1%	1.4e-03
64	2.15e+01	2.10e+01	5.1e-01	61.9	96.7%	1.4e-03
128	2.14e+01	2.10e+01	4.4e-01	124	97.2%	1.4e-03

The spatial domain is $[0, p\pi]\times[0, \pi]$ witn $h = \pi/256$, where p is the number of processors.
The testing time interval is $[0, 1]$ with Δt =5.0e–03.

Listed in Table 4-5 are the scalability testing data of methods SEIDD1 and SEIDD3 for a convection diffusion problem. The experimental data show that the computation time almost remains the same as the problem size increases with the machine ensemble size such that the memory usage on each processor remains the same. This phenomenon is well under expectation

Table 5: Solving $u_t = \Delta u + \sin(x)u_x - \cos(x)u$ with SEIDD3

Processors	T_total	T_comp	T_comm	Speedup	Efficiency	Max-Err
1	9.89e+00	9.89e+00	0.0e-02	1.00	100%	8.4e-04
2	1.04e+01	1.02e+01	1.9e-01	1.90	95.1%	8.4e-04
3	1.04e+01	1.02e+01	2.3e-01	2.85	95.1%	8.4e-04
4	1.05e+01	1.02e+01	2.9e-01	3.77	94.2%	8.4e-04
5	1.05e+01	1.02e+01	3.4e-01	4.71	94.2%	8.4e-04
6	1.06e+01	1.02e+01	3.5e-01	5.60	93.3%	8.4e-04
7	1.05e+01	1.02e+01	3.1e-01	6.59	94.2%	8.4e-04
8	1.05e+01	1.02e+01	2.8e-01	7.54	94.2%	8.4e-04
10	1.05e+01	1.02e+01	2.7e-01	9.92	94.2%	8.4e-04
12	1.05e+01	1.02e+01	3.0e-01	11.3	94.2%	8.4e-04
14	1.07e+01	1.01e+01	5.5e-01	12.9	92.4%	8.4e-04
16	1.06e+01	1.02e+01	4.1e-01	14.9	93.3%	8.4e-04
20	1.05e+01	1.01e+01	4.1e-01	18.8	94.2%	8.4e-04
24	1.05e+01	1.01e+01	4.0e-01	22.6	94.2%	8.4e-04
28	1.06e+01	1.02e+01	3.5e-01	26.1	93.3%	8.4e-04
32	1.07e+01	1.01e+01	5.5e-01	29.6	92.4%	8.4e-04
48	1.09e+01	1.00e+01	8.7e-01	43.6	90.7%	8.4e-04
64	1.07e+01	1.01e+01	5.6e-01	59.2	92.4%	8.4e-04
128	1.08e+01	1.01e+01	6.7e-01	117	91.6%	8.4e-04

The spatial domain is $[0, p\pi] \times [0, \pi]$ witn $h = \pi/256$, where p is the number of processors.
The testing time interval is $[0, 1]$ with Δt =5.0e–03.

since the stabilization process has a very low computation overhead. The communication time increases slowly as the number of processors increases. As the number of processors increases from one to one hundred twenty eight, the efficiency has decreased less than 4 percentage points for the SEIDD1 algorithm, and decreased about 9 percentage points for the SEIDD3 algorithm. This rate of performance decrease is very slow, matching well with the analysis given in Section 2. The difference in the decreased percentage between the SEIDD1 and SEIDD3 algorithm is due to the difference of computation costs of the two algorithms. The computation time used by the SEIDD3 algorithm is less than half of that used by the SEIDD1 algorithm while the communication time consumed by the two algorithms are the same.

4. Conclusion. We developed a class of stabilized explicit-implicit domain decomposition methods by adding a stabilization step to the explicit-implicit domain decomposition methods. The EIDD methods are globally non-iterative, non-overlapping domain decomposition methods, which are computationally and communicationally efficient for each time step calculation when compared with Schwarz method based parabolic solvers. However EIDD methods suffer from either stability or consistency related time step size restrictions, while Schwarz methods could maintain the good stability condition of implicit temporal discretization schemes. The proposed SEIDD methods have inherited the advantages of EIDD methods in time-stepwise efficiency but are free from any time step size restriction, thus possessing the advantages of both the EIDD methods and the Schwarz method based parabolic solvers. These merits in stability, parallel speedup and scalability of the SEIDD methods were confirmed by tests on an SGI Origin 2000 computer.

Acknowledgement. The authors would like to thank four anonymous reviewers for their comments and suggestions. The authors are also grateful to the National Center for Supercomputing Applications (NCSA) for providing access to its SGI Origin 2000 parallel computer.

References

[1] K. Black, *Polynomial collocation using a domain decomposition solution to parabolic PDE's via the penalty method and explicit/implicit time marching*, J. Sci. Comput., 7 (1992), no. 4, 313–338.

[2] X.-C. Cai, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), 41–61.

[3] ———, *Mulitiplicative Schwarz methods for parabolic problems*, SIAM J. Sci. Comput., 15 (1994), 587–603.

[4] X.-C. Cai, W. D. Gropp and D. E. Keyes, *A comparison of some domain decomposition algorithms for nonsymmetric elliptic problems*, J. Numer. Lin. Alg. Appl., 1993.

[5] T. F. Chan and T. Mathew, *Domain decomposition algorithms*, Acta Numerica, 1994, 61–143.

[6] C. Dawson and T. Dupont, *Explicit/implicit, conservative domain decomposition procedures for parabolic problems based on block-centered finite difference*, SIAM J. Numer. Anal. 31 (1994), no. 4, 1045–1061.

[7] C. Dawson, Q. Du, and T. Dupont, *A finite difference domain decomposition algorithm for numerical solution of the heat equation*, Math. Comp. 57 (1991), no. 195, 63–71.

[8] J. Douglas and J. Gunn, *A general formulation of alternating direction method: Part I. Parabolic and hyperbolic problems*, Numer. Math., 6 (1964), 428–453.

[9] M. Dryja, *Substructuring methods for parabolic problems*. Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations (Moscow, 1990), 264–271, SIAM, Philadelphia, PA, 1991.

[10] M. Dryja and O. B. Widlund, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Rep. 339, Courant Inst., New York Univ., 1987.

[11] W. D. Gropp and D. E. Keyes, *Domain decomposition on parallel computers*, Impact of Computing in Science and Engineering, 1 (1989), 421–439.

[12] D. E. Keyes, *Domain decomposition: a bridge between nature and parallel computers*, NASA ICASE Technical Report No. 92-44, NASA Langley Research Center Hampton, VA 23681-0001, 1992.

[13] D. E. Keyes and W. D. Gropp, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Statis. Comput., **8** (1987), 166–202.

[14] Y. A. Kuznetsov, *New algorithms for approximate realization of implicit difference schemes*, Sov. J. Numer. Ana.Math. Modell. 3 (1988), 99–114.

[15] Yu. M. Laevsky, *A domain decomposition algorithm without overlapping subdomains for the solution of parabolic equations.* (Russian) Zh. Vychisl. Mat. i Mat. Fiz. 32 (1992), no. 11, 1744–1755; translation in Comput. Math. Math. Phys. 32 (1992), no. 11, 1569–1580.

[16] ———, *Explicit-implicit domain decomposition method for solving parabolic equations.* (Russian) Computing methods and technology for solving problems in mathematical physics (Russian), 30–46, Ross. Akad. Nauk Sibirsk. Otdel., Vychisl. Tsentr, Novosibirsk, 1993.

[17] Y. M. Laevsky and S. V. Gololobov, *Explicit-implicit domain decomposition methods for the solution of parabolic equations.* (Russian) Sibirsk. Mat. Zh. 36 (1995), no. 3, 590–601, ii; translation in Siberian Math. J. 36 (1995), no. 3, 506–516.

[18] Yu. M. Laevsky and O. V. Rudenko, *Splitting methods for parabolic problems in non-rectangular domains.* Appl. Math. Lett. 8 (1995), no. 6, 9–14.

[19] T. Mathew, P. Polyakov, G. Russo and J. Wang, *Domain decomposition operator splittings for the solution of parabolic equations*, SIAM J. Sci. Comput. 19 (1998), no. 3, 912–932.

[20] B. F. Smith, P. E. Bjorstad and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.

[21] Y. Zhuang, *Classically Unstable Approximations for Linear Evolution Equations and Applications*, Ph.D. dissertation, Department of Mathematics, Louisiana State University Baton Rouge, August 2000.

[22] ———, *A Class of Stable, Globally Noniterative, Nonoverlapping Domain Decomposition Algorithms for the Simulation of Parabolic Evolutionary Systems*, Ph.D. dissertation, Department of Computer Science, Louisiana State University Baton Rouge, December 2000.