SIAM J. SCI. COMPUT.
Vol. 34, No. 5, pp. C233–C248

A PARALLEL SPACE-TIME ALGORITHM*

ANDREW J. CHRISTLIEB†, RONALD D. HAYNES‡, AND BENJAMIN W. ONG§

Abstract. With the continued evolution of computing architectures towards many-core computing, algorithms that can effectively and efficiently use many cores are crucial. In this paper, we propose, as a proof of principle, a parallel space-time algorithm that layers time parallelization together with a parallel elliptic solver to solve time dependent partial differential equations (PDEs). The parallel elliptic solver utilizes domain decomposition to divide a spatial grid into subdomains, and applies a parallel Schwarz iteration to find consistent solutions. The high-order parallel time integrator employed belongs to the family of revisionist integral deferred correction methods (RIDC) introduced by Christlieb, Macdonald, and Ong [*SIAM J. Sci. Comput.*, 32 (2010), pp. 818–835], which allows for the small scale parallelization of solutions to initial value problems. The two established algorithms are combined in this proposed space-time algorithm to add parallel scalability. As a proof of concept, we utilize a framework involving classical Schwarz matching conditions and RIDC integrators. It will be shown that the resulting Schwarz iterations can be analyzed using standard domain decomposition analysis, and that the required Schwarz iterations (at each time step) can be evaluated simultaneously in parallel, after initial start-up costs. Additionally, it will be shown that if the domain decomposition iteration converges for the prediction step, the domain decomposition iterations for the correction steps will converge at the same rate. Numerical experiments demonstrate that the RIDC-DD algorithms attain the designed order of accuracy. Several scaling studies are also performed.

Key words. integral deferred correction, parallel time integrators, distributed computing, domain decomposition

AMS subject classifications. 65M55, 65Y05, 68M14

DOI. 10.1137/110843484

1. Introduction.

For the foreseeable future, high performance computing will be synonymous with "many-core" computing. In the field of scientific computing, the development of algorithms that can utilize many cores effectively and efficiently will be crucial. The present authors are interested in the numerical solution of time dependent partial differential equations (PDEs). In this arena, a widely accepted approach is to introduce spatial parallelism using domain decomposition (DD) [28]. Loosely speaking, DD methods fall within the class of *data parallel algorithms*, where a domain of interest is divided into various subdomains, and the same task is performed on each subdomain by a computing node. Each subdomain is assigned a different computing node. While DD methods are largely successful, it is not practical to increase the number of subdomains indefinitely.

In this paper, we present an algorithm to combine spatial parallelism with time parallelism to improve scalability of existing parallel spatial solvers. This is accom-

*Submitted to the journal's Software and High-Performance Computing section August 5, 2011; accepted for publication (in revised form) August 8, 2012; published electronically October 25, 2012. This work was supported by AFOSR grant FA9550-07-1-0092, NSF grant DMS-0934568, XSEDE allocation TG-DMS120004, the High Performance Computing Center (HPCC) at Michigan State University, and NSERC Discovery grant 311796.

http://www.siam.org/journals/sisc/34-5/84348.html

†Department of Mathematics, Michigan State University, East Lansing, MI 48824 (andrewch@math.msu.edu).

‡Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's, NL, A1C 5S7 Canada (rhaynes@mun.ca).

§Corresponding author. Institute for Cyber Enabled Research, Michigan State University, East Lansing, MI 48824 (ongbw@msu.edu).

plished by marrying DD methods with RIDC (revisionist integral deferred correction) integrators, which are *task parallel algorithms*. The general idea is to utilize multiple cores to perform parallel tasks on each subdomain.

RIDC integrators are "parallel across the step" integrators. The "revisionist" terminology was adopted to highlight that (1) RIDC is a *revision* of the standard integral defect correction (IDC) formulation [10, 7, 4] , and (2) successive corrections, running in parallel but lagging in time, *revise* and improve the approximation to the solution [3, 5]. The main idea is to rewrite the defect correction framework so that, after initial start-up costs, each correction loop can be lagged behind the previous correction loop in a manner that facilitates running the predictor and correctors in parallel.

As a proof of concept, we solve the linear heat equation and the advection-diffusion-reaction equation implicitly using classical Schwarz matching conditions and various RIDC integrators. Analysis and numerical experiments show that (1) the Schwarz iterations will converge for the prediction and correction loops of a RIDC algorithm, (2) as usual the rate of convergence is influenced by the size of the overlap region, the size of the time step, and the number of subdomains, and, finally, (3) the RIDC-DD algorithm attains the designed order of accuracy in space and time.

For completeness, we mention that there are some parallel efforts towards developing algorithms that employ both spatial and temporal parallelization. For example, Maday and Turinici [22] combine spatial domain decomposition with a "parareal in time" algorithm [21], which is a temporal domain decomposition integrator (i.e., data-parallel algorithm), to show that classical parallel iterative solvers can be combined with parareal to allow for more rapid solutions if parallel architectures are available. The parareal time integrator has two components which are alternately applied: a coarse (inexpensive) integrator that is applied sequentially, and a fine (expensive) integrator that is applied in parallel. Their preliminary analysis is promising. There is also active research by Emmett and Minion [11] towards casting the parareal in time algorithm in the defect correction framework, and combining this new parallel time integrator with a spatial (potentially parallel) multigrid solver. The framework that they propose further couples the spatial and temporal components by utilizing a coarse mesh for the coarse time integrator, and a fine mesh for the fine time integrator.

The remainder of this paper is organized as follows. In section 2, DD and the RIDC framework are reviewed and then combined to form our hybrid scheme as presented and analyzed in section 3. Then, numerical experiments demonstrating important features of our space-time approach are presented in section 4, followed by concluding remarks in section 5.

2. Review. In this section we provide a brief review and bibliography of DD and RIDC methods. In doing so we also establish necessary notation for the description of our main algorithm in section 3.

2.1. Domain decomposition. Domain decomposition is a divide-and-conquer approach for solving PDEs. First developed by Schwarz [27] as a theoretical tool to establish existence of solutions for Laplace's equation on irregular domains, it has developed [29] into a robust tool for solving elliptic PDEs in distributed computing environments. For example, several legacy codes such as `CHOMBO` [8] and `ICEPIC` [23] incorporate spatial DD algorithms. The DD approach replaces the PDE by a coupled system of PDEs over some partitioning of the spatial domain into overlapping or nonoverlapping subdomains. The coupling is provided by necessary transmission conditions at the subdomain boundaries. These transmission conditions are chosen

to ensure the DD algorithm converges and to optimize the convergence rate.

To establish some necessary notation, suppose that we wish to solve a PDE of the form

$$\begin{aligned} \mathcal{L}(u) &= 0, \quad x \in \Omega, \\ \mathcal{B}(u) &= 0, \quad x \in \partial\Omega, \end{aligned} \tag{2.1}$$

where $\mathcal{L}, \mathcal{B}$ are some differential operators in the spatial variable x and Ω is the spatial domain. We begin our discussion in $\mathbb{R}^1$ where we assume $\Omega = [0, L]$. Suppose Ω is partitioned into D subdomains,

$$\Omega = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_D,$$

where $\Omega_j = [\alpha_j L, \beta_j L]$ with $\alpha_1 = 0$ and $\beta_D = 1$. If $\alpha_{j+1} = \beta_j$, then the subdomains are nonoverlapping, and overlapping if $\alpha_{j+1} < \beta_j$ for $j = 1, \ldots, D-1$. In general we will assume $\Omega_i \cap \Omega_j = \varnothing$ if $|i - j| > 1$, that is, only adjacent subdomains may overlap; this requires $\beta_j \leq \alpha_{j+2}$ for $j = 1, \ldots, D-2$.

Here we will consider the simplest approach, overlapping classical Schwarz which makes use of Dirichlet transmission conditions at the subdomain interfaces. To this end, we solve (2.1) for u by composing the solutions of

$$\begin{aligned} &\mathcal{L}(u_j) = 0, \quad x \in \Omega_j \quad \forall j = 1, 2, \ldots, D, \\ &u_j(\alpha_j L) = u_{j-1}(\alpha_j L), \\ &u_j(\beta_j L) = u_{j+1}(\beta_j L). \end{aligned}$$

The subdomain solutions, u_j, are obtained by iteration. Here we iterate until convergence so u_j and u_{j+1} (for example) will agree within the overlap region. In practice the solution in the overlap region may be obtained by averaging u_j and u_{j+1}. In this paper we will be interested in the parallel Schwarz iteration. For $k = 1, 2, \ldots$, solve

$$\begin{aligned} &\mathcal{L}(u_j^k) = 0, \quad x \in \Omega_j \quad \forall j = 1, 2, \ldots, D, \\ &u_j^k(\alpha_j L) = u_{j-1}^{k-1}(\alpha_j L), \\ &u_j^k(\beta_j L) = u_{j+1}^{k-1}(\beta_j L). \end{aligned}$$

Initial subdomain solutions at the interfaces are needed to begin the algorithm. Iterations of the form (2.3) have been studied for many different classes of PDE, or choices of $\mathcal{L}$.

We now generalize the above presentation for higher dimensions and irregular domains. Suppose that $\Omega \in \mathbb{R}^n$ is still partitioned into D subdomains,

$$\Omega = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_D.$$

Then, overlapping classical Schwarz with Dirichlet transmission conditions decomposes the PDE (2.1) into the following system of PDEs:

$$\begin{aligned} &\mathcal{L}(u_j) = 0, \quad x \in \Omega_j, \\ &u_j(z_{jl}) = u_l(z_{jl}), \quad z_{jl} \in (\partial\Omega_j \cap \Omega_l) \quad \forall j, l \in 1, 2, \ldots, D. \end{aligned} \tag{2.2}$$

Again, the subdomain solutions u_j are obtained by iteration. For $k = 1, 2, \ldots$, solve

$$\begin{aligned} &\mathcal{L}(u_j^k) = 0, \quad x \in \Omega_j, \\ &u_j^k(z_{jl}) = u_l^{k-1}(z_{jl}), \quad z_{jl} \in (\partial\Omega_j \cap \Omega_l) \quad \forall j, l \in 1, 2, \ldots, D. \end{aligned} \tag{2.3}$$

It is important to note that the classical Schwarz iteration (2.3) is typically not a practical method. Indeed, in applications overlapping subdomains are not possible (for example, coupled atmospheric and ocean models), and moreover, convergence is too slow. In practice, one must replace the Dirichlet boundary conditions with more general transmission operators, resulting in optimized and optimal Schwarz methods; see, for example, [15, 13, 16, 19].

The primary problem of interest here is the solution of time dependent PDEs,

$$\begin{aligned} u_t &= \mathcal{N}(t,u), && x \in \Omega \times [0,T], \\ \mathcal{B}(u) &= 0, && x \in \partial\Omega \times [0,T], \\ u(0,x) &= g(x), && x \in \Omega, \end{aligned} \tag{2.4}$$

where $\mathcal{N}, \mathcal{B}$ are (possibly time dependent) spatial differential operators. DD approaches for problems of the form (2.4) can be broadly categorized as either Schwarz waveform relaxation methods [17, 14, 12, 18], which partition $\Omega \times [0,T]$ into overlapping or nonoverlapping space-time domains, $\Omega_j \times [0,T]$, with $\cup_{j=1}^{D} \Omega_j = \Omega$, or, methods which discretize the PDE in time and solve the sequence of elliptic problems using DD approaches like (2.3); see [1, 2]. It is the latter approach which will be the focus of this paper. A backward Euler discretization of (2.4) results in an elliptic problem to be solved at each time step. A schematic of this approach is shown in Figure 2.1. The arrows indicate MPI communications, required for the DD iteration, between the subdomains located on each node.

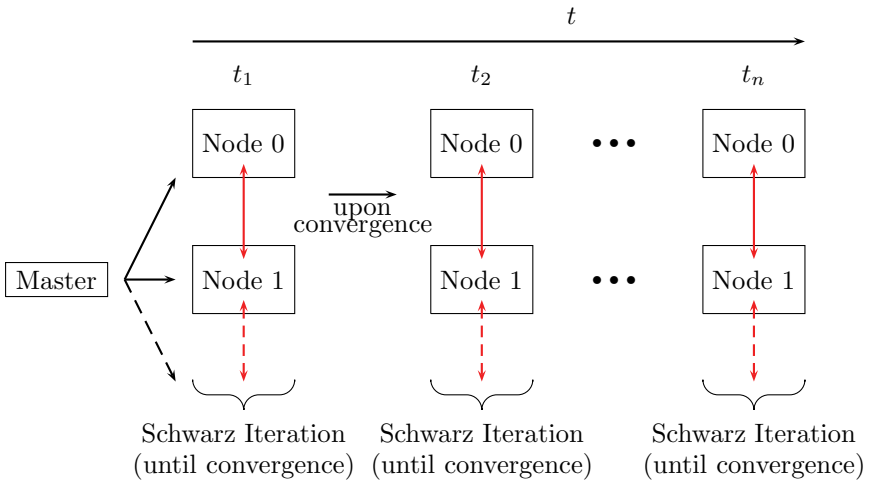

FIG. 2.1. *Communication diagram where backward Euler is used to integrate in time with a classical Schwarz iteration in space.*

2.2. RIDC. RIDC methods are a family of parallel time integrators that can be broadly classified as predictor-corrector algorithms. The main idea is to correct an inaccurate solution which was computed using a low order method (e.g., a Euler integrator). The correction is computed by solving an error equation, which will be derived in section 2.2.1. It was shown previously in [10] that successive application of the correction procedure not only increases the accuracy of the solution, but also increases the formal order of accuracy of the scheme. A simple extension to incorporate parallelism was introduced by Christlieb, Macdonald, and Ong in [3]. This

was accomplished by delaying each correction from the previous level, as illustrated in Figure 2.2 for a backward Euler predictor and corrector—the white circles denote solution values that are computed simultaneously. This staggering in time means that the predictor and each corrector can all be executed concurrently, in parallel.

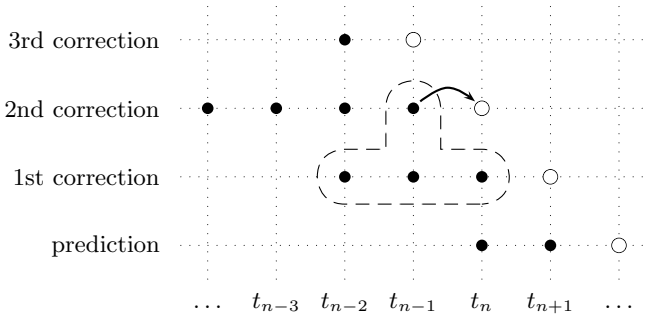

FIG. 2.2. *This plot shows the staggering required for a fourth order RIDC scheme (RIDC4-BE), constructed using backward Euler predictors and correctors. The time axis runs horizontally, and the correction levels run vertically. The white circles denote solution values that are computed simultaneously, e.g., core* 0 *is computing the prediction solution at time* t_{n+2} *while core* 1 *is simultaneously computing the* 1*st correction solution at time* t_{n+1}*, etc.*

2.2.1. Error equation. Suppose an approximate solution $\eta(t,x)$ to (2.4) is computed. Denote the (unknown) exact solution as $u(t,x)$. Then, the error of the approximate solution is

$$e(t,x) = u(t,x) - \eta(t,x). \tag{2.5}$$

Define the residual as $\epsilon(t,x) = \eta_t(t,x) - \mathcal{N}(t,\eta(t,x))$. The time derivative of the error (2.5) satisfies

$$e_t = u_t - \eta_t = \mathcal{N}(t,u) - (\mathcal{N}(t,\eta) + \epsilon).$$

The integral form of the error equation

$$\left[e + \int_0^t \epsilon(\tau,x)\,d\tau\right]_t = \mathcal{N}(t,\eta+e) - \mathcal{N}(t,\eta) \tag{2.6}$$

can then be solved for $e(t,x)$ using the initial condition $e(0,x) = 0$, since we assume that at $t = 0$, the initial condition is exactly specified. Equation (2.6) can be manipulated to give an expression for the corrected solution, $\eta(t,x) + e(t,x)$. We note that this approach of computing a correction to an approximate solution can be bootstrapped. If an approximate solution is denoted as $\eta^p(t,x)$, a correction $e^p(t,x)$ can be computed, and a corrected solution $\eta^{p+1}(t,x) = \eta^p(t,x) + e^p(t,x)$ obtained. The error of this corrected solution, $e^{p+1}(t,x)$, can then be computed to give a further corrected solution, $\eta^{p+2}(t,x) = \eta^{p+1}(t,x) + e^{p+1}(t,x)$. With some algebra and the bootstrapped notation, (2.6) can be expressed as

$$\left[\eta^{p+1} - \int_0^t \mathcal{N}(\tau,\eta^p)\,d\tau\right]_t = \mathcal{N}\left(t,\eta^{p+1}\right) - \mathcal{N}(t,\eta^p). \tag{2.7}$$

2.2.2. Start-up and shutdown. During most of an RIDC calculation, multiple solution levels are computed in parallel using multiple computing cores (or as we will see shortly for our parallel space time algorithm, multiple cores on multiple nodes). However, the computing cores in the RIDC algorithm cannot start simultaneously; each must wait for the previous level to compute sufficient η values before marching all of them in a pipeline fashion. The order in which computations can be performed during start-up is illustrated in Figure 2.3 for a fourth order RIDC constructed with backward Euler integrators. The jth processor (running the jth correction) must initially wait for $j(j+1)/2$ steps.

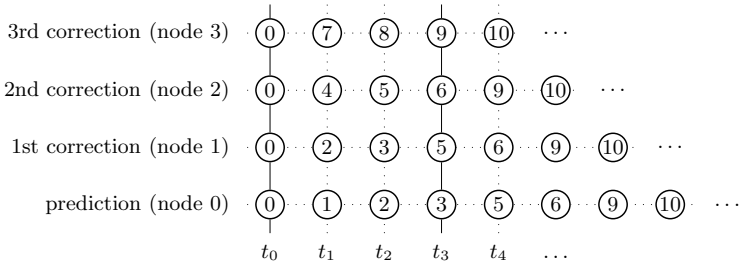

FIG. 2.3. *This figure is a graphical representation of how the RIDC4-BE algorithm is started. The time axis runs horizontally, the correction levels run vertically. All nodes are initially populated with the initial data at* t_0*. This is represented by step* 0 *(enclosed in a circle). At computing step* 1*, node* 0 *computes the predicted solution at* t_1*. The remaining nodes stay idle. At computing step* 2*, node* 0 *computes the predicted solution at time* t_2*, node* 1 *computes the* 1*st corrected solution at time* t_1*, the remaining two nodes stay idle. Note that in this starting algorithm, special care is taken to ensure that minimum memory is used by not letting the computing cores run ahead until they can be marched in a pipeline—in this example, when node* 3 *starts computing* t_3*.*

In terms of shutdown, the calculation ends when the highest level (most accurate) computation reaches the final time $t_N = b$. Note that the predictor and lower level correctors will reach $t_N = b$ earlier and as a consequence some computing threads will sit idle.

2.2.3. Further comments. Readers might be familiar with spectral deferred correction (SDC) methods [10] where one subdivides the time domain into discrete (potentially nonuniform) intervals, $0 = t_0 < t_1 < \cdots < t_N = T$. Each interval is further subdivided using m Gauss–Legendre (or Gauss–Lobatto) nodes, $t_i = t_{i,0} < t_{i,1} < \cdots < t_{i,m-1} = t_{i+1}$. The SDC method is applied on each interval $[t_i, t_{i+1}]$. Indeed, the m Gauss–Legendre nodes are able to approximate the integral of the residual with accuracy $(2m-1)$. Additionally, the initial condition used to start the SDC method in each interval is the most accurate solution obtained in the previous interval. In a standard SDC formulation, the same stencil involving all the quadrature nodes is used to approximate the integral for each correction equation.

In contrast, RIDC methods do not require that each interval is split into m nodes. In fact, it is advantageous to subdivide each interval into $M > m$ nodes to improve the efficacy of the RIDC methods. As formulated, a reduced stencil, which varies for each correction level, is used to approximate the integral for the correction equation. For example, only two quadrature nodes are used to approximate the integral for the first correction equation (assuming first order predictors and correctors are used), whereas three quadrature nodes are used to approximate the integral for the second correction equation. The use of uniformly spaced nodes and reduced stencils minimizes the overhead for starting an RIDC method, before the nodes can be simultaneously

marched in parallel. One could imagine a sequence of Gauss–Lobatto nodes which are used to subdivide an interval at the cost of adversely affecting the starting overhead for RIDC. The improved stability of RIDC constructed with Gauss–Lobatto nodes might be advantageous, but is beyond the scope of this paper. Finally, we comment that we cannot increase the order of RIDC constructed with uniformly spaced nodes indefinitely as (i) it is not practical and (ii) the Runge phenomenon [26], which arises from using equispaced interpolation points, will eventually cause the scheme to become unstable. In practice, however, 12th order RIDC methods have been constructed without any observable instability [6].

3. Space-time algorithm. We now describe a space-time algorithm combining the RIDC framework for integration in time with a classical, parallel, Schwarz iteration in space. A Schwarz iteration is used within each prediction and correction step. Recall the time dependent PDE of interest (2.4),

$$\begin{aligned} u_t &= \mathcal{N}(t,u), \quad x \in \Omega \times [0,T], \\ \mathcal{B}(u) &= 0, \quad x \in \partial\Omega \times [0,T], \\ u(0,x) &= g(x), \quad x \in \Omega, \end{aligned}$$

where $\mathcal{N}, \mathcal{B}$ are (possibly time dependent) spatial differential operators. A schematic of the Schwarz iteration and communication is shown in Figure 3.1. Two main events occur for each step. First, the shared memory on each node (containing the memory footprint for each grid point in the corresponding subdomain) is updated with the converged Schwarz iteration solution from the previous step. This data intensive step benefits from shared memory access. Second, there are four simultaneous Schwarz iterations used to obtain a fourth order method in time. Core p on every node performs a Schwarz iteration for the pth correction step (where $p = 0$ is the prediction loop). Only boundary information is communicated for each Schwarz iteration; thus the communication overhead for using MPI is minimal.

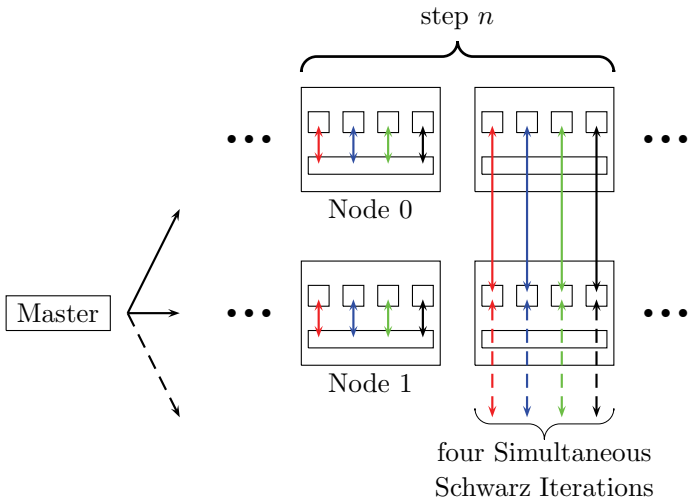

FIG. 3.1. *RIDC framework for time integration with a classical Schwarz iteration in space. For this RIDC4-DD illustration, it is assumed that each node, working on a subdomain, has four cores with access to shared memory. After an initial start-up period, each step consists of updating the memory footprint, followed by four simultaneous Schwarz iterations.*

3.1. Prediction level. We begin by semidiscretizing this problem in time. Let $u^n(x)$ denote our approximation to $u(t_n, x)$ obtained by a first order implicit Euler discretization to (2.4). The prediction steps are given, for $n = 1, 2, \ldots$, as

$$\begin{aligned} u^{n+1} - u^n - \Delta t\, \mathcal{N}(t^{n+1}, u^{n+1}) = 0, \quad x \in \Omega, \\ \mathcal{B}(u^{n+1}(z)) = 0, \quad z \in \partial\Omega, \end{aligned} \tag{3.1}$$

with $u^0(x) = g(x)$. For each n we solve the elliptic problem (3.1) using a classical Schwarz algorithm as described in section 2.1; also cf. [1, 2]. Suppose $\Omega \in \mathbb{R}^n$ is partitioned into D overlapping subdomains,

$$\Omega = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_D.$$

Then the following system of coupled PDEs has to be solved:

$$\begin{aligned} u_j^{n+1} - u_j^n - \Delta t\, \mathcal{N}(t^{n+1}, u_j^{n+1}) = 0, \quad x \in \Omega_j \quad \forall j \in 1, 2, \ldots, D, \\ u_j^{n+1}(z_{jl}) = u_l^{n+1}(z_{jl}), \quad z_{jl} \in (\partial\Omega_j \cap \Omega_l) \setminus \partial\Omega \quad \forall j, l \in 1, 2, \ldots, D, \\ \mathcal{B}(u_j^{n+1}(z)) = 0, \quad z \in \partial\Omega_j \cap \partial\Omega \quad \forall j \in 1, 2, \ldots, D. \end{aligned} \tag{3.2}$$

The subdomain solutions, u_j^{n+1}, are obtained by iteration. For $k = 1, 2, \ldots$, solve

$$\begin{aligned} u_j^{n+1,k} - u_j^n - \Delta t\, \mathcal{N}(t^{n+1}, u_j^{n+1,k}) = 0, \quad x \in \Omega_j \quad \forall j \in 1, 2, \ldots, D, \\ u_j^{n+1,k}(z_{jl}) = u_l^{n+1,k-1}(z_{jl}), \quad z_{jl} \in (\partial\Omega_j \cap \Omega_l) \setminus \partial\Omega \quad \forall j, l \in 1, 2, \ldots, D, \\ \mathcal{B}(u_j^{n+1}(z)) = 0, \quad z \in \partial\Omega_j \cap \partial\Omega \quad \forall j \in 1, 2, \ldots, D. \end{aligned} \tag{3.3}$$

3.2. Correction levels. After an initial wait, the RIDC framework computes corrections to our approximation to $u^{n+1}(x)$ in parallel. These correction steps are also obtained using a Schwarz iteration. Before discretizing the error equation (2.7) in time and space, we need to modify our notation to differentiate between successively corrected solutions, as previously discussed in section 2.2.1. We let $u^{[p]}(t, x)$ denote the approximation to $u(t, x)$ obtained after the pth correction, and let $u^{n,[p]}(x)$ denote the discretized approximation to $u(t_n, x)$ obtained after the pth correction. Hence, $u^{n,[0]}(x)$ is the solution obtained at the prediction level, as described in section 3.1. With this notation, the error equation (2.7) becomes

$$\left[u^{[p+1]} - \int_0^t \mathcal{N}(\tau, u^{[p]})\, d\tau \right]_t = \mathcal{N}\left(t, u^{[p+1]}\right) - \mathcal{N}\left(t, u^{[p]}\right). \tag{3.4}$$

With the change of variables $q = u^{[p+1]} - \int_0^t \mathcal{N}(\tau, u^{[p]})$, (3.4) becomes

$$q_t = \mathcal{N}\left(t, q + \int_0^t \mathcal{N}(\tau, u^{[p]})\, d\tau\right) - \mathcal{N}\left(t, u^{[p]}\right).$$

Applying an implicit backward Euler integrator gives

$$\frac{q^{n+1} - q^n}{\Delta t} = \mathcal{N}\left(t^{n+1}, q^{n+1} + \int_0^{t^{n+1}} \mathcal{N}(\tau, u^{n+1,[p]})\, d\tau\right) - \mathcal{N}\left(t^{n+1}, u^{n+1,[p]}\right).$$

Now, changing the variables back gives

$$u^{n+1,[p+1]} - u^{n,[p+1]} - \Delta t\,\mathcal{N}(t^{n+1}, u^{n+1,[p+1]}) + \Delta t\,\mathcal{N}(t^{n+1}, u^{n+1,[p]}) = \int_{t^n}^{t^{n+1}} \mathcal{N}(t, u^{[p]})\,dt.$$

Applying a quadrature rule gives

$$u^{n+1,[p+1]} - u^{n,[p+1]} - \Delta t\,\mathcal{N}(t^{n+1}, u^{n+1,[p+1]}) + \Delta t\,\mathcal{N}(t^{n+1}, u^{n+1,[p]}) = \begin{cases} \displaystyle\sum_{\nu=0}^{p+1} \alpha_\nu \mathcal{N}(t^{n+1-\nu}, u^{n+1-\nu,[p]}) & \text{if } n \geq p, \\ \displaystyle\sum_{\nu=0}^{p+1} \alpha_i \mathcal{N}(t^{\nu}, u^{i,[p]}) & \text{if } n < p, \end{cases} \tag{3.5}$$

where α_ν are quadrature weights used to approximate $\int_{t_n}^{t_{n+1}} \mathcal{N}(\tau, u^{[p-1]}(\tau))\,d\tau$, i.e.,

$$\alpha_\nu = \begin{cases} \displaystyle\int_{t^n}^{t^{n+1}} \prod_{i=0, i\neq\nu}^{p+1} \frac{(t - t^{n+1-i})}{(t^{n+1-l} - t^{n+1-i})}\,dt & \text{if } n \geq p, \\ \displaystyle\int_{t^n}^{t^{n+1}} \prod_{i=0, i\neq\nu}^{p+1} \frac{(t - t^{i})}{(t^{l} - t^{i})}\,dt & \text{if } n < p. \end{cases} \tag{3.6}$$

Several important observations should be made. Equation (3.5) is an elliptic equation for the unknown variable $u^{n+1,[p+1]}$, provided $u^{n+1,[p]}$ and $u^{\cdot,[p]}$ are known. This elliptic equation will also be solved using the classical Schwarz algorithm described in section 2.1. Second, the number of terms in the sum (3.6) increases with level p because the integral must be approximated with increasing accuracy. Finally, the choice of uniform stencils picked for the quadrature is not unique; in practice, however, the different quadrature stencils do not seem to significantly affect the accuracy of the solution.

The coupled system of PDEs for the correction level is thus

$$\begin{aligned} &u_j^{n+1,[p+1]} - u_j^{n,[p+1]} - \Delta t\,\mathcal{N}(t^{n+1}, u_j^{n+1,[p+1]}) + \Delta t\,\mathcal{N}(t^{n+1}, u_j^{n+1,[p]}) \\ &= \begin{cases} \displaystyle\sum_{\nu=0}^{p+1} \alpha_\nu \mathcal{N}(t^{n+1-l}, u_j^{n+1-l,[p]}) & \text{if } n \geq p, \\ \displaystyle\sum_{\nu=0}^{p+1} \alpha_\nu \mathcal{N}(t^{l}, u_j^{l,[p]}) & \text{if } n < p, \end{cases} \qquad x \in \Omega_j \quad \forall j \in 1, 2, \ldots, D, \\ &u_j^{n+1,[p+1]}(z_{jl}) = u_l^{n+1,[p+1]}(z_{jl}), \quad z_{jl} \in (\partial\Omega_j \cap \Omega_l) \setminus \partial\Omega \quad \forall j, l \in 1, 2, \ldots, D, \\ &\mathcal{B}(u_j^{n+1,[p+1]}(z)) = 0, \quad z \in \partial\Omega_j \cap \partial\Omega \quad \forall j \in 1, 2, \ldots, D. \end{aligned} \tag{3.7}$$

These are solved by a parallel Schwarz iteration. For $k = 1, 2, \ldots$, solve

$$u_j^{n+1,[p+1],k} - u_j^{n,[p+1]} - \Delta t\,\mathcal{N}(t^{n+1}, u_j^{n+1,[p+1],k}) + \Delta t\,\mathcal{N}(t^{n+1}, u_j^{n+1,[p]})$$
$$= \begin{cases} \displaystyle\sum_{\nu=0}^{p+1} \alpha_\nu \mathcal{N}(t^{n+1-\nu}, u_j^{n+1-\nu,[p]}) & \text{if } n \geq p, \\ \displaystyle\sum_{\nu=0}^{p+1} \alpha_\nu \mathcal{N}(t^{\nu}, u_j^{\nu,[p]}) & \text{if } n < p, \end{cases} \qquad x \in \Omega_j \quad \forall j \in 1, 2, \ldots, D,$$

(3.8)
$$u_j^{n+1,[p+1],k}(z_{jl}) = u_l^{n+1,[p+1],k-1}(z_{jl}), \quad z_{jl} \in (\partial\Omega_j \cap \Omega_l) \setminus \partial\Omega \quad \forall j, l \in 1, 2, \ldots, D,$$
$$\mathcal{B}(u_j^{n+1,[p+1],k}(z)) = 0, \quad z \in \partial\Omega_j \cap \partial\Omega \quad \forall j \in 1, 2, \ldots, D.$$

3.3. Convergence of the parallel Schwarz iteration. The convergence properties of the parallel Schwarz iterations for the prediction step (3.3) and correction steps (3.8) follow quite naturally from the typical analysis of the Schwarz iteration for the elliptic problems which result upon discretization in time; cf. [1, 2].

Denote the errors between the subdomain solution $u_j^{n+1,[p+1]}$ and the iterates $u_j^{n+1,[p+1],k}$ by $e_j^{n+1,[p+1],k}$, i.e.,

$$e_j^{n+1,[p+1],k} = u_j^{n+1,[p+1]} - u_j^{n+1,[p+1],k}$$

for $p = 0, 1, 2, 3$. We use $p = 0$ to denote the solution of the prediction step.

Subtracting (3.3) from (3.2) and (3.8) from (3.7), we see the errors for the prediction and all correction steps satisfy the same system

(3.9)
$$e_j^{n+1,[p+1],k} - \Delta t \left[\mathcal{N}(t^{n+1}, u_j^{n+1,[p+1]}) - \mathcal{N}(t^{n+1}, u_j^{n+1,[p+1],k})\right] = 0,$$
$$e_j^{n+1,[p+1],k}(z_{jl}) = e_l^{n+1,[p+1],k-1}(z_{jl}), \quad z_{jl} \in (\partial\Omega_j \cup \Omega_l) \setminus \partial\Omega \;\forall j, l \in 1, 2, \ldots, D,$$
$$B(u_j^{n+1,[p+1]}(z_{jl})) - B(u_j^{n+1,[p+1],k}(z_{jl})) = 0, \; z_{jl} \in \partial\Omega_j \cup \Omega \;\forall j \in 1, 2, \ldots, D.$$

The contraction (to zero) of solutions of (3.9) is known for many operators $\mathcal{N}$ including second-order linear elliptic problems (see section 4), Navier–Stokes for incompressible and compressible flows, and Euler's equations of gas dynamics; see, for example, [25]. The iteration for many of these problems is described for both overlapping and nonoverlapping domains, the latter requiring additional interface conditions beyond the Dirichlet conditions of the classical Schwarz iteration presented here. For example, if $\mathcal{N}$ is simply the diffusion operator d^2/dx^2, the contraction rate for classical Schwarz, ρ, on two subdomains and over two iterations, is given by

$$\rho = \exp\left(-2L/\sqrt{\Delta t}\right),$$

where L is the overlap; see, for example, [15]. As is typical, the contraction rate improves as the overlap increases and the time step decreases.

Indeed the main observation is that if a DD iteration is known to converge for the solution of the prediction step (3.1), then the same DD iteration will converge for the correction steps in the RIDC algorithm. Moreover, (3.9) suggests that the Schwarz iterations for the prediction and correction steps will converge in approximately the same number of iterations (though the initial errors will be slightly different). Hence, on any particular node, each core will finish its correction in unison, allowing the algorithm to step forward seamlessly. That is, the algorithm enjoys a natural load balancing.

4. Numerical examples.

4.1. A first example. We apply our parallel space-time algorithm, RIDC-DD with classical Schwarz iteration in space, to the linear heat equation in two spatial dimensions,

$$\begin{aligned} u_t &= u_{xx} + u_{yy}, \quad \Omega \in [0,1] \times [0,1], \quad t \in [0, 0.1], \\ u(0,x,y) &= \exp(-10\sqrt{(x-0.5)^2 + (y-0.5)^2}), \\ u(t,0,y) &= u(t,1,y) = u(t,x,0) = u(t,x,1) = 0. \end{aligned}$$

The spatial derivative is approximated using centered finite differences. The domain is discretized into 40×60 cells. These cells are then grouped into four overlapping subdomains in a 2×2 grid, with an overlap of two cells. The Schwarz iterations are performed until a tolerance of 10^{-12} is reached in the predictors and correctors. In practice, this tolerance should be chosen in tandem with the local error tolerance for the time integration, and potentially, a posteriori error estimates for the spatial problem; cf. [20]. A reference solution is computed using a DIRK4 (diagonally implicit Runge–Kutta) method on a single domain. The error is computed by mapping the decomposed solution back to a single grid, taking averages in the overlap region as necessary, and comparing the composed solution with the reference solution.

In Figure 4.1, the convergence plots show that our RIDC-DD algorithm with classical Schwarz iterations in space converges with the designed orders of accuracy in time.

In Figure 4.2, a convergence study shows second order convergence in space for the fourth order RIDC-DD algorithm as the spatial resolution is refined.

In Figure 4.3, the average number of Schwarz iterations per time step is plotted as a function of Δt for both the prediction and correction Schwarz iterations. Consistent with the analysis in section 3.3, the average number of Schwarz iterations decreases with Δt for the predictors and correctors. Hence, it can be argued that there will be minimal load balancing issues for this RIDC-DD algorithm as formulated.

As mentioned in section 3.3, the convergence of overlapping Schwarz improves as the size of the overlap region increases. We demonstrate this in Figure 4.4 by illustrating a representative convergence study at a fixed instant in the time integration

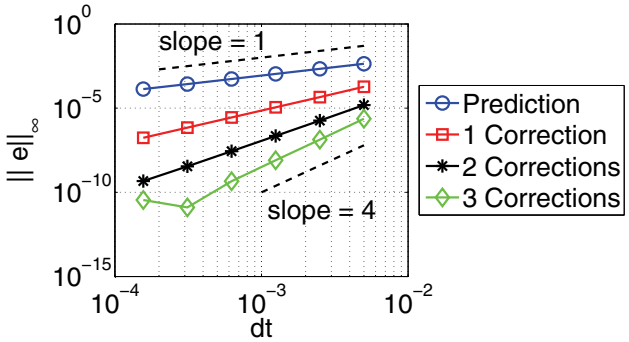

FIG. 4.1. *Second, third, and fourth order RIDC-DD with classical Schwarz iterations in space converge to the reference solution with the designed orders of accuracy. The Schwarz iterations are iterated until a tolerance of* 10^{-12} *is reached for the predictors and correctors (hence the flat line attained after three corrections). Here,* Δx *is fixed while* Δt *is varied.*

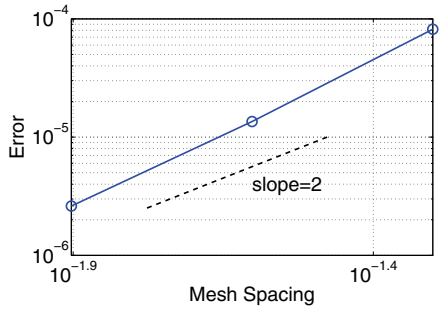

FIG. 4.2. *Second order convergence in space is demonstrated for the fourth order RIDC-DD algorithm. Here, Δt is fixed while Δx is varied. The Schwarz iterations are iterated until a tolerance of 10^{-12} is reached for the predictors and correctors.*

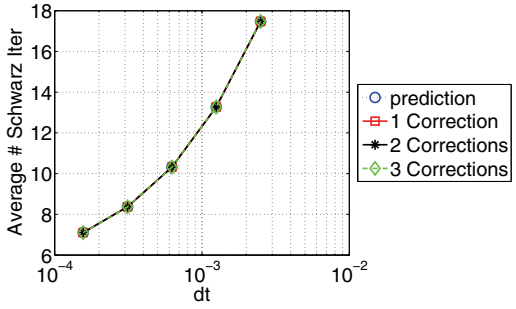

FIG. 4.3. *This graph shows the average number of Schwarz iterations per time step for various time step sizes, with a fixed tolerance. Each prediction and correction takes the same number of Schwarz iterations for each step. This is due, in part, to the stopping criterion for the Schwarz iteration being fixed at the same value for the prediction and each correction loop. One could, in practice, relax the tolerance for the prediction and the earlier correction levels.*

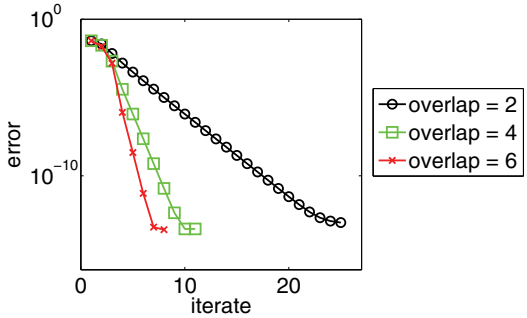

FIG. 4.4. *Effect of overlap on the convergence of the Schwarz iteration for prediction and correction steps. As the size of the overlap increases, the rate of convergence of the classical Schwarz iteration increases.*

for varying amounts of overlap. The overlaps are chosen as $2\Delta x, 4\Delta x$, and $6\Delta x$. As expected, the rate of convergence improves as the number of cells in the overlap region increases. In this plot, a time step of 2.5×10^{-3} was used in addition to the domain decomposition parameters described above. The rate of convergence was plotted for the final time step, $t = 0.1$.

To test the parallel efficacy of this algorithm, a hybrid OpenMP–MPI code was developed. Implementation details will be discussed in [24]. The domain is discretized into 600×300 cells. These cells are then grouped into 18 overlapping domains in a 6×3 grid with an overlap of four cells. As before the Schwarz iterations are performed until a tolerance of 10^{-12} is reached. Figure 4.5 shows a soft scaling study, which shows that the wall clock time to compute a first order (in time) DD solution with $N = 18$ cores is approximately the same as the wall clock time to compute a pth order (in time) RIDC-DD solution using pN cores. Error bars are computed using the standard deviation of wall clock times based on ten numerical experiments per data point. The large error bars are likely due to varying network traffic and architecture on the high performance computing system used.

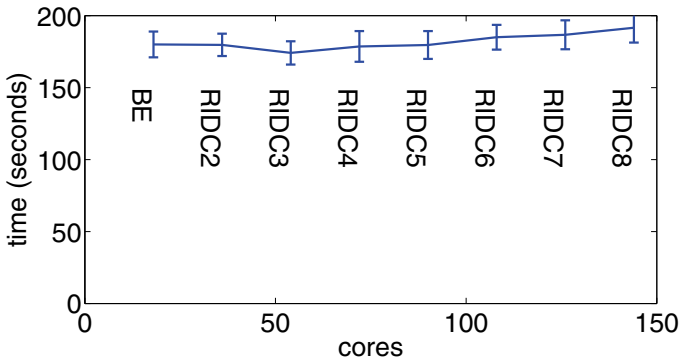

FIG. 4.5. *Weak scaling study for the linear diffusion equation. The wall clock time to compute a first order (in time) solution using N cores is approximately the same as the wall clock time to compute a pth order solution, using pN cores.*

A hard scaling study is shown in Figure 4.6, where the heat equation is solved using an eighth order RIDC-DD algorithm with a varying number of simultaneous Schwarz iterations, as dictated by the number of cores. (The number of subdomains is held fixed at 18 for this study.) If only 18 cores are available, then one OpenMP thread is set for the RIDC loop, meaning that only one Schwarz iteration is performed at a time. If 36 cores are available, then two OpenMP threads are set for the RIDC loop, meaning that two simultaneous Schwarz iterations are performed. If $p \cdot 18$ cores are available, then p simultaneous Schwarz iterations are performed. It is unclear presently whether the drop in efficiency is due to (i) the higher network traffic that results from the simultaneous Schwarz iterations, (ii) saturated memory bandwidth, or (iii) poor code design.

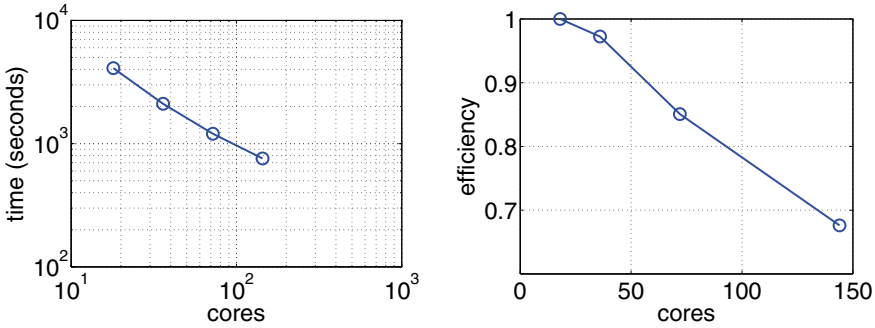

FIG. 4.6. *Hard scaling study for the linear diffusion equation. The wall clock time to compute an eighth order (in time) RIDC-DD solution, using a varying number of cores, is used to compute the efficiency of the method.*

4.2. A second example. Next, we consider a linear advection-diffusion-reaction equation

$$
\begin{aligned}
u_t + a \cdot \nabla u &= \gamma \Delta u - cu, \quad x \in \Omega = [0,1] \times [0,1], \\
u(0,x,y) &= \exp(-10\sqrt{(x-0.5)^2 + (y-0.5)^2}), \\
u(t,0,y) &= u(t,1,y) = u(t,x,0) = u(t,x,1) = 0,
\end{aligned} \tag{4.1}
$$

where $\gamma = 0.1$, $a = (1,1)$, $c = 1$ [9]. Upwind differencing is used to approximate the advection operator, and centered finite differences are used to approximate the diffusion operator. Schwarz iterations are performed until a tolerance of 10^{-12} is reached in the predictors and the correctors. The domain is discretized into 600×300 cells. These cells were then grouped into 18 overlapping domains in a 6×3 grid with an overlap of four cells. A soft scaling study and hard scaling study (described earlier in section 4.1) show a similar behavior (in terms of speed-up and efficiency) when solving the advection-diffusion-reaction equation; see Figures 4.7 and 4.8.

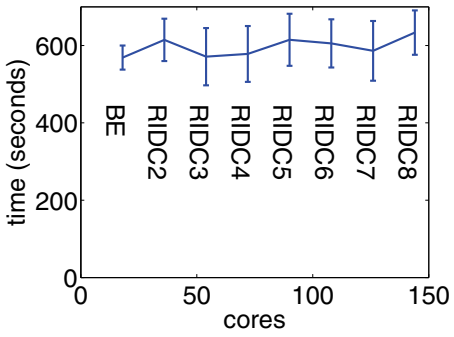

FIG. 4.7. *Soft scaling study for the advection-diffusion-reaction equation. The wall clock time to compute a first order (in time) solution using N cores is approximately the same as the wall clock time to compute a pth order solution, using pN cores.*

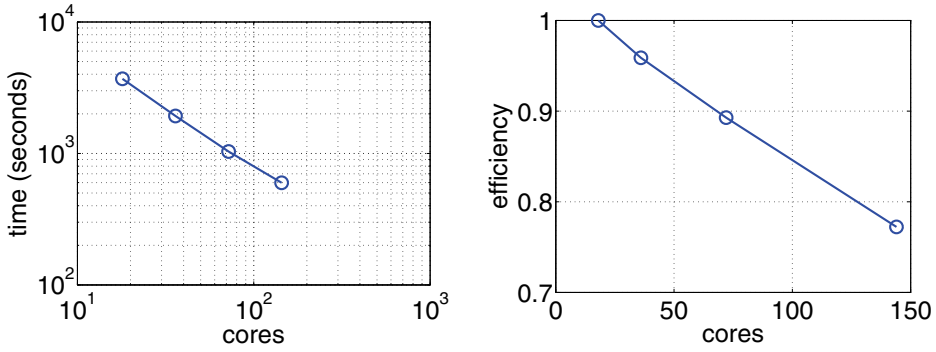

FIG. 4.8. *Hard scaling study for the advection-diffusion-reaction equation. The wall clock time to compute an eighth order (in time) RIDC-DD solution, using a varying number of cores, is used to compute the efficiency of the method.*

5. Conclusions. In this paper we proposed a parallel space-time solver for time dependent PDEs based on domain decomposition in space and RIDC, a parallel predictor-corrector method, in time. As a proof of concept, we utilize a framework involving classical Schwarz matching conditions and RIDC integrators. We show that the proposed algorithm, referred to as RIDC-DD for short, requires multiple Schwarz iterations per step that can be evaluated simultaneously in parallel (after initial start-up costs), provided the appropriate computing resources are available. Analysis is presented to demonstrate that RIDC-DD algorithms will converge, and that the rate of convergence is consistent with results from standard domain decomposition analysis. Numerical experiments demonstrate that RIDC-DD algorithms attain their designed order of accuracy, and load balancing appears to be a nonissue with our implemented version of the proposed algorithm. In practice, the tolerance for the DD iterations should be chosen in tandem with the tolerance of the local error control/adaptive time stepping mechanism. An important step in this direction has been given in [20]. The authors are presently exploring optimal/optimized transmission conditions at the subdomain boundaries and a coarse grid correction [29] to handle the degradation of the DD convergence rate for large numbers of subdomains.

Acknowledgments. The authors would like to thank Dirk Colbry for insightful comments and enlightening discussions related to this work, and Benjamin Mintz for the technical help in getting our programs to run on XSEDE resources.

REFERENCES

[1] X.-C. Cai, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), pp. 41–61.
[2] X.-C. Cai, *Multiplicative Schwarz methods for parabolic problems*, SIAM J. Sci. Comput., 15 (1994), pp. 587–603.
[3] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput., 32 (2010), pp. 818–835.
[4] A. Christlieb, M. Morton, B. Ong, and J.-M. Qiu, *Semi-implicit integral deferred correction constructed with additive Runge–Kutta methods*, Commun. Math. Sci., 9 (2011), pp. 879–902.
[5] A. Christlieb and B. Ong, *Implicit parallel time integrators*, J. Sci. Comput., 49 (2011), pp. 167–179.

[6] A. Christlieb, B. Ong, and J.-M. Qiu, *Comments on high-order integrators embedded within integral deferred correction methods*, Commun. Appl. Math. Comput. Sci., 4 (2009), pp. 27–56.

[7] A. Christlieb, B. Ong, and J.-M. Qiu, *Integral deferred correction methods constructed with high order Runge-Kutta integrators*, Math. Comp., 79 (2010), pp. 761–783.

[8] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. Van Straalen, *Chombo software package for amr applications-design document*, available online at http://seesar.lbl.gov/anag/chombo/ChomboDesign-3.0.pdf, 2000.

[9] O. Dubois, *Optimized Schwarz Methods for the Advection-Diffusion Equation and for Problems with Discontinuous Coefficients*, Ph.D. Thesis, McGill University, Canada, ProQuest LLC, Ann Arbor, MI, 2007.

[10] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT, 40 (2000), pp. 241–266.

[11] M. Emmett and M. Minion, *Toward an efficient parallel in time method for partial differential equations*, Comm. App. Math. and Comp. Sci., 7 (2012), pp. 105–132.

[12] M. J. Gander and C. Rohde, *Overlapping Schwarz waveform relaxation for convection-dominated nonlinear conservation laws*, SIAM J. Sci. Comput., 27 (2005), pp. 415–439.

[13] M. J. Gander and S. Vandewalle, *On the superlinear and linear convergence of the parareal algorithm*, Lect. Notes Comput. Sci. Eng. 55, Springer, Berlin, 2007, pp. 291–298.

[14] M. J. Gander, *A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations*, Numer. Linear Algebra Appl., 6 (1999), pp. 125–145.

[15] M. J. Gander, *Optimized Schwarz methods*, SIAM J. Numer. Anal., 44 (2006), pp. 699–731.

[16] M. J. Gander, L. Halpern, and F. Nataf, *Optimized Schwarz methods*, in Proceedings of the Twelfth International Conference on Domain Decomposition Methods, Chiba, Japan, T. Chan, T. Kako, H. Kawarada, and O. Pironneau, eds., Domain Decomposition Press, Bergen, 2001, pp. 15–28.

[17] M. J. Gander and A. M. Stuart, *Space-time continuous analysis of waveform relaxation for the heat equation*, SIAM J. Sci. Comput., 19 (1998), pp. 2014–2031.

[18] E. Giladi and H. B. Keller, *Space-time domain decomposition for parabolic problems*, Numer. Math., 93 (2002), pp. 279–313.

[19] L. Halpern, *Absorbing boundary conditions and optimized Schwarz waveform relaxation*, BIT, 46 (2006), pp. S21–S34.

[20] P. Jiránek, Z. Strakoš, and M. Vohralík, *A posteriori error estimates including algebraic error and stopping criteria for iterative solvers*, SIAM J. Sci. Comput., 32 (2010), pp. 1567–1590.

[21] J. L. Lions, Y. Maday, and G. Turinici, *A "parareal" in time discretization of PDEs*, C.R. Acad. Sci. Sér. Paris I Math., 332 (2001), pp. 661–668.

[22] Y. Maday and G. Turinici, *The parareal in time iterative solver: A further direction to parallel implementation*, in Domain Decomposition Methods in Science and Engineering, Lect. Notes Comput. Sci. Eng. 40, Springer, Berlin, 2005, pp. 441–448.

[23] P. Mardahl, A. Greenwood, T. Murphy, and K. Cartwright, *Parallel Performance Characteristics of ICEPIC*, 2003 User Group Conference, Bellevue, WA, 2003.

[24] B. Ong and R. Haynes, *Hybrid MPI-OpenMP algorithms for the parallel space-time solution of time dependent PDEs*, submitted. Available online at http://mathgeek.us/research/papers/ridc-tdd_implementation.pdf.

[25] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, New York, 1999.

[26] C. Runge, *Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten*, Zeit. für Math. und Phys., 46 (1901), pp. 224–243.

[27] H. A. Schwarz, *Über einen grenzübergang durch alternierendes verfahren*, Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15 (1870), pp. 272–286.

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

[29] A. Toselli and O. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.