

A PARALLEL SPACE–TIME ALGORITHM

ANDREW CHRISTLIEB*, RONALD D. HAYNES†, AND BENJAMIN 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 classic, 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 in 2010, 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 a fourth order RIDC method. It will be shown that the resulting Schwarz iterations can be analyzed using standard domain decomposition analysis, and that the four Schwarz iterations per time step can be evaluated simultaneously in parallel, after initial start-up costs. Numerical experiments demonstrate that the RIDC–DD algorithm attains the designed order of accuracy, and that load balancing appears to be a non-issue with our implemented version of the proposed algorithm.

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

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

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) [25]. Loosely speaking, DD methods fall in 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 (where 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 to existing (parallel) spatial solvers. This is accomplished by marrying DD methods with so-called RIDC (revisionist integral deferred correction) integrators, which are loosely speaking, *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 [9, 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 re-write the defect correction framework so that,

*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, Canada, A1C 5S7, rhaynes@mun.ca

‡Corresponding author, Institute for Cyber Enabled Research, Michigan State University, East Lansing, MI, 488824, ongbw@msu.edu

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 heat equation implicitly using classical Schwarz matching conditions and a fourth order RIDC method. Analysis and numerical experiments show that (1) the Schwarz iterations will converge for the prediction and correction loops of a RIDC algorithm, (2) 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 time.

For completeness, we mention that there are some “parallel” (pun intended) efforts towards developing algorithms that employ both spatial and temporal parallelization. For example, Maday and Turinici [21], combine spatial domain decomposition with “parareal in time” algorithm [20], 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 (cheap) 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 Minion et al. [10] towards casting the parareal in time algorithm in the defect correction framework, and combining this new parallel time integrator with a spatial (potentially parallel) multi-grid 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 the paper is organized as follows. In Section 2, domain decomposition and the RIDC framework are reviewed, and then combined to form our hybrid scheme as presented 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 (DD) is a divide and conquer approach to solving partial differential equations (PDEs). First developed by Schwarz [24] as a theoretical tool to establish existence of solutions for Laplace’s equation on irregular domains, it has developed [26, 8, 22] into a robust tool for solving elliptic PDEs in distributed computing environments. The DD approach replaces the PDE by a coupled system of PDEs over some partitioning of the spatial domain into overlapping or non–overlapping 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 we wish to solve some elliptic PDE

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

where \mathcal{L}, \mathcal{B} are some differential operators in the spatial variable x and Ω is the spatial domain. For concreteness we assume $\Omega = [0, L]$, however, the ideas can be

easily generalized to high dimensions and irregular domains. We partition Ω into D subdomains as

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

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 non-overlapping, and overlapping if $\alpha_{j+1} < \beta_j$ for $j = 1, \dots, D-1$. In general we will assume $\Omega_i \cap \Omega_j = \emptyset$ if $|i-j| > 1$, that is, only adjacent subdomains may overlap, this requires $\beta_j \leq \alpha_{j+2}$ for $j = 1, \dots, D-2$.

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

$$\begin{aligned} \mathcal{L}(u_j) &= 0, & x \in \Omega_j, \\ u_j(\alpha_j L) &= u_{j-1}(\alpha_j L), \\ u_j(\beta_j L) &= u_{j+1}(\beta_j L). \end{aligned} \tag{2.2}$$

The subdomain solutions u_j are obtained by iteration. In this paper we will be interested in the parallel Schwarz iteration. For $k = 1, 2, \dots$, solve

$$\begin{aligned} \mathcal{L}(u_j^k) &= 0, & x \in \Omega_j, \\ u_j^k(\alpha_j L) &= u_{j-1}^{k-1}(\alpha_j L), \\ u_j^k(\beta_j L) &= u_{j+1}^{k-1}(\beta_j L). \end{aligned} \tag{2.3}$$

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} .

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, [13, 14, 15, 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, [16, 12, 11, 17], which partition $\Omega \times [0, T]$ into overlapping or non-overlapping space-time domains, $[\alpha_j L, \beta_j L] \times [0, T]$, **or** methods which discretize the PDE in time and solve the sequence of elliptic problems using DD approaches like (2.3), [1, 2]. It is the latter approach which will be the focus of this paper. A schematic of how this is typically implemented is shown Figure 2.1.

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. an Euler

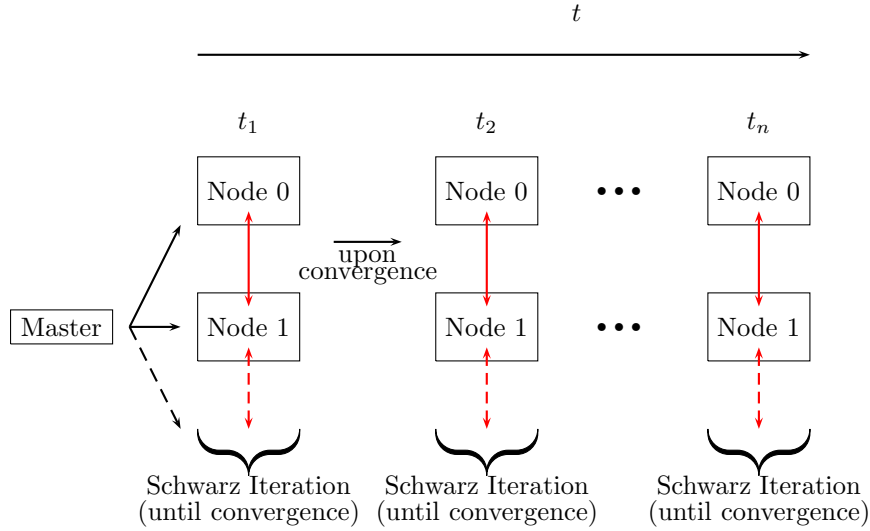


Fig. 2.1: Communication diagram when backward Euler is used to integrate in time with a classical Schwarz iteration in space.

integrator). The correction is computed by solving an error equation, which will be derived in Section 2.2.1. It was shown previously in [9] 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 simultaneously computed. This staggering in time means that the predictor and each corrector can all be executed simultaneously, in parallel.

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

$$e(t, x) = u(t, x) - \eta(t, x). \quad (2.5)$$

Define the residual as $\epsilon(t, x) = \eta_t(t, x) - \mathcal{N}(t, \eta(t, x))$. Then 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,

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

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

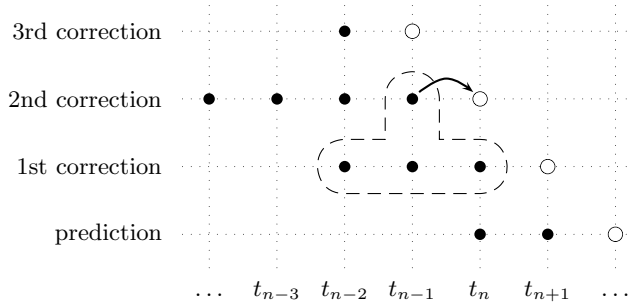


Fig. 2.2: (RIDC4-BE) This plot shows the staggering required for a fourth order RIDC scheme, constructed using backward Euler predictors and correctors. The time axis runs horizontally, and the correction levels run vertically. The white circles denote solution values that are simultaneously computed,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

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 boot-strapped notation, equation (2.6) can be expressed as

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

2.2.2. Start-up and Shutdown. During most of a 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 j^{th} processor (running the j^{th} correction) must initially wait for $j(j+1)/2$ steps.

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.

Finally, we comment that we cannot increase the order of RIDC indefinitely as (i) it is not practical and (ii) the Runge phenomenon [23], which arises from using equi-spaced interpolation points, will eventually cause the scheme to become unstable. In practice, 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

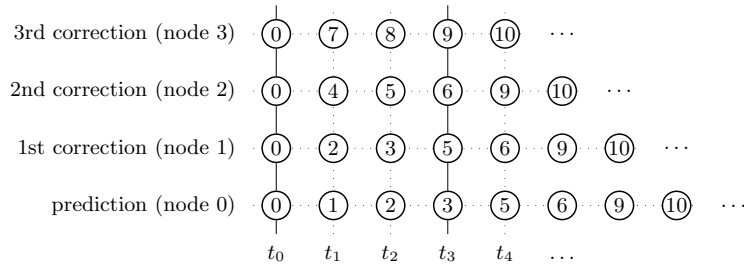


Fig. 2.3: This figure is a graphical representation of how the RIDC4-BE algorithm is started. The time axis runs horizontally, the correction levels run vertically. All nodes are initially populated with the initial data at t_0 . This is represented by computing step 0 (enclosed in a circle). At computing step 1, node 0 computes the predicted solution at t_1 . The remaining nodes remain idle. At computing step 2, node 0 computes the predicted solution at time t_2 , node 1 computes the 1st corrected solution at time t_1 , the remaining two nodes remain 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 .

iteration in space. A Schwarz iteration is used within each prediction and correction step.

To demonstrate the approach, we describe the RIDC-DD space-time algorithm for the linear heat equation in one spatial dimension,

$$\begin{aligned}
 u_t &= u_{xx}, & x &\in [a, b] \\
 u(t, a) &= g_a(t), & u(t, b) &= g_b(t), & t &\in [0, T] \\
 u(0, x) &= u_0(x), & x &\in [a, b],
 \end{aligned} \tag{3.1}$$

on two subdomains. Extensions to higher dimensions and an arbitrary number of sub-domains are straightforward. In fact, the implementation and numerical examples presented in Section 4 will solve equation (3.1) using RIDC-DD in two spatial dimensions using a grid of 2×2 subdomains.

A schematic of the iteration and communication is shown in Figure 3.1. The specifics of the start-up are illustrated in Section 3.4. The primary observation that should be made, is that two main events occur for each step. (1) The shared memory on each node (containing the memory footprint for the corresponding subdomain) is updated with the converged Schwarz iteration solution from the previous step; this is communication intensive since each cell in the subdomain needs to be updated. (2) Core p on every node performs a Schwarz iteration for the p th correction step, (where $p = 0$ is the prediction loop). Only boundary information is communicated for each Schwarz iteration, and no updates to the shared memory is required.

3.1. Prediction Level. We begin by semi-discretizing 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 equation (3.1). The prediction steps are given, for $n = 1, 2, \dots$

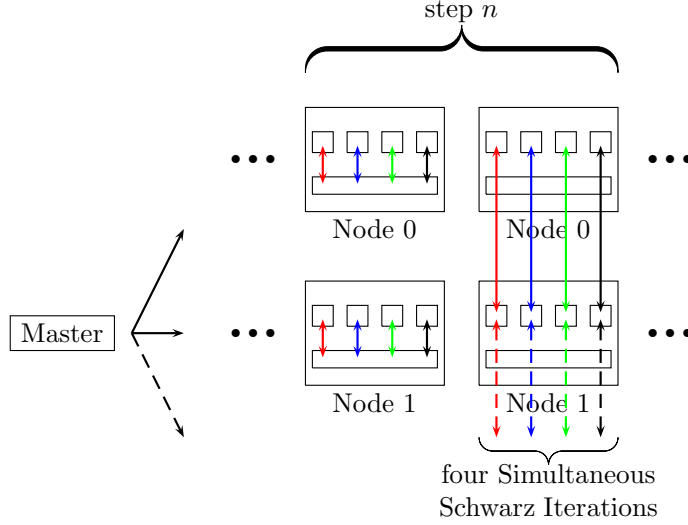


Fig. 3.1: RIDC framework for time integration with a classical Schwarz iteration in space. For this RIDC4-DD illustration, it is assumed that each node, working on a subdomain, has four cores with access to shared memory. At each “step”, two main events occur. (1) The shared memory on each node (containing the memory footprint for the corresponding subdomain) is updated with the converged Schwarz iteration solution from the previous step. This is communication intensive, and benefits from the proposed shared memory paradigm. (2) Then, four simultaneous Schwarz iterations are performed, where core 0 on each node computes the Schwarz iteration for the prediction loop, core 1 on each node computes the Schwarz iteration for correction #1, etc. Only boundary information is communicated for each Schwarz iteration, and no updates to the shared memory is required.

as

$$\begin{aligned} u^{n+1} - u^n - \Delta t u_{xx}^{n+1} &= 0, \\ u^{n+1}(a) &= g_a(t_{n+1}), \quad u^{n+1}(b) = g_b(t_{n+1}), \end{aligned} \quad (3.2)$$

with $u^0(x) = u_0(x)$. For each n we solve the elliptic problem (3.2) using a classical Schwarz algorithm as described in Section 2.1.

To illustrate, suppose we decompose $\Omega = [a, b]$ into two overlapping subdomains $\Omega_1 = [a, \beta]$ and $\Omega_2 = [\alpha, b]$ and obtain u^{n+1} by composing the subdomain solutions $v^{n+1}(x)$ on Ω_1 and $w^{n+1}(x)$ on Ω_2 . The subdomain solutions satisfy

$$\begin{aligned} v^{n+1} - u^n - \Delta t v_{xx}^{n+1} &= 0, \quad x \in \Omega_1 \\ v^{n+1}(a) &= g_a(t_{n+1}), \quad v^{n+1}(\beta) = w^{n+1}(\beta), \end{aligned} \quad (3.3)$$

and

$$\begin{aligned} w^{n+1} - u^n - \Delta t w_{xx}^{n+1} &= 0, \quad x \in \Omega_2, \\ w^{n+1}(\alpha) &= v^{n+1}(\alpha), \quad w^{n+1}(b) = g_b(t_{n+1}). \end{aligned} \quad (3.4)$$

This coupled system of PDEs is solved by a parallel Schwarz iteration. For $k = 1, 2, \dots$, solve

$$\begin{aligned} v^{[k],n+1} - u^n - \Delta t v_{xx}^{[k],n+1} &= 0, \quad x \in \Omega_1, \\ v^{[k],n+1}(a) &= g_a(t_{n+1}), \quad v^{[k],n+1}(\beta) = w^{[k-1],n+1}(\beta), \end{aligned} \quad (3.5)$$

and

$$\begin{aligned} w^{[k],n+1} - u^n - \Delta t w_{xx}^{[k],n+1} &= 0, \quad x \in \Omega_2, \\ w^{[k],n+1}(\alpha) &= v^{[k-1],n+1}(\alpha), \quad w^{[k],n+1}(b) = g_b(t_{n+1}). \end{aligned} \quad (3.6)$$

We arbitrarily set $w^{[0],n+1}(\beta) = u^n(\beta)$, and $v^{[0],n+1}(\alpha) = u^n(\alpha)$ in equations (3.5) and (3.6). In practice, any reasonable initial iterate can be chosen. More importantly, the Schwarz iteration completely decouples the solution of (3.5) and (3.6), allowing the subdomain solves to occur in parallel. We set $u^{n+1}(x)$ to be the composition of the limiting (converged) solutions $v^{[\infty],n+1}, w^{[\infty],n+1}$.

We discretize (3.5–3.6) in space using $N + 1$ points on Ω_1 and $M + 1$ points on Ω_2 , so that

$$N\Delta x = \beta - a \quad \text{and} \quad M\Delta x = b - \alpha.$$

The solution is stepped from time level n to $n + 1$ by the iteration: For $k = 1, 2, \dots$

$$\begin{aligned} v_i^{[k],n+1} - u_i^n - \frac{\Delta t}{\Delta x^2} \left(v_{i+1}^{[k],n+1} - 2v_i^{[k],n+1} + v_{i-1}^{[k],n+1} \right) &= 0, \quad i = 1, \dots, N - 1, \\ v_0^{[k],n+1} &= g_a(t_{n+1}), \quad v_N^{[k],n+1} = w_\gamma^{[k-1],n+1}, \end{aligned} \quad (3.7)$$

and

$$\begin{aligned} w_j^{[k],n+1} - u_j^n - \frac{\Delta t}{\Delta x^2} \left(w_{j+1}^{[k],n+1} - 2w_j^{[k],n+1} + w_{j-1}^{[k],n+1} \right) &= 0, \quad j = 1, \dots, M - 1, \\ w_0^{[k],n+1} &= v_{N-\gamma}^{[k-1],n+1}, \quad w_M^{[k],n+1} = g_b(t_{n+1}). \end{aligned} \quad (3.8)$$

The quantity γ here denotes the number of intervals in the overlap region $[\alpha, \beta]$.

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 themselves obtained using a Schwarz iteration. Before discretizing the error equation (2.7) in time and space, we (unfortunately) need to modify our notation to differentiate between successively corrected solutions, as previously discussed in Section 2.2.1. We let $u^{n,p}(x)$ denote the approximation to $u(t_n, x)$ obtained after the p th correction. Hence, $u^{n,0}(x)$ is the solution obtained at the prediction level, as described in Section 3.1.

We now semi-discretize the error equation (2.7) in time. Applying a first order implicit Euler integrator gives (after some algebra)

$$\begin{aligned} u^{n+1,p} - u^{n,p} - \Delta t u_{xx}^{n+1,p} + \Delta t u_{xx}^{n+1,p-1} &= \begin{cases} \sum_{l=0}^p \alpha_l u_{xx}^{n+1-l,p-1} & \text{if } (n \geq p - 1) \\ \sum_{l=0}^p \alpha_l u_{xx}^{l,p-1} & \text{if } (n < p - 1) \end{cases} \\ u^{n+1,p}(a) &= g_a(t_{n+1}), \quad u^{n+1,p}(b) = g_b(t_{n+1}), \end{aligned} \quad (3.9)$$

where α_l are quadrature weights which approximate $\int_{t_n}^{t_{n+1}} u_{xx}^{p-1}(\tau) d\tau$, i.e.,

$$\alpha_l = \begin{cases} \int_{t_n}^{t_{n+1}} \prod_{i=0, i \neq l}^p \frac{(t - t^{n+1-i})}{(t^{n+1-l} - t^{n+1-i})} dt, & \text{if } (n \geq p-1) \\ \int_{t_n}^{t_{n+1}} \prod_{i=0, i \neq l}^p \frac{(t - t^i)}{(t^l - t^i)} dt, & \text{if } (n < p-1) \end{cases} \quad (3.10)$$

Several important observations should be made. Equation (3.9) is an elliptic equation for the unknown variable $u^{n+1,p}$ provided $u^{n,p}$ and $u^{n,p-1}$ are known. This elliptic equation will also be solved using the classical Schwarz algorithm described in Section 2.1. Secondly, the number of terms in the sum (3.10) increases with level p because the integral must be approximated with increasing accuracy. Finally, the choice of stencils picked for the quadrature is not unique; in practice, the different quadrature stencils do not seem to significantly affect the accuracy of the solution.

In keeping with the presentation of ideas in Section 3.1, the corrected subdomain solutions, $v^{n+1,p}$ on Ω_1 and $u^{n+1,p}$ on Ω_2 , must satisfy

$$v^{n+1,p} - u^{n,p} - \Delta t v_{xx}^{n+1,p} + \Delta t u^{n+1,p-1} = \begin{cases} \sum_{l=0}^p \alpha_l u_{xx}^{n+1-l,p-1} & \text{if } (n \geq p-1) \\ \sum_{l=0}^p \alpha_l u_{xx}^{l,p-1} & \text{if } (n < p-1) \end{cases}$$

$$v^{n+1,p}(a) = g_a(t_{n+1}), \quad v^{n+1,p}(b) = w^{n+1,p}(\beta), \quad x \in \Omega_1$$

and

$$w^{n+1,p} - u^{n,p} - \Delta t w_{xx}^{n+1,p} + \Delta t u^{n+1,p-1} = \begin{cases} \sum_{l=0}^p \alpha_l u_{xx}^{n+1-l,p-1} & \text{if } (n \geq p-1) \\ \sum_{l=0}^p \alpha_l u_{xx}^{l,p-1} & \text{if } (n < p-1) \end{cases}$$

$$w^{n+1,p}(\alpha) = v^{n+1,p}(\alpha), \quad w^{n+1,p}(b) = g_b(t_{n+1}), \quad x \in \Omega_2.$$

The coupled system of PDEs is then solved by a parallel Schwarz iteration. For $k = 1, 2, \dots$ solve

$$v^{[k],n+1,p} - u^{n,p} - \Delta t v_{xx}^{[k],n+1,p} + \Delta t u^{n+1,p-1} = \begin{cases} \sum_{l=0}^p \alpha_l u_{xx}^{n+1-l,p-1} & \text{if } (n \geq p-1) \\ \sum_{l=0}^p \alpha_l u_{xx}^{l,p-1} & \text{if } (n < p-1) \end{cases}$$

$$v^{[k],n+1,p}(a) = g_a(t_{n+1}), \quad v^{[k],n+1,p}(b) = w^{[k-1],n+1,p}(\beta), \quad x \in \Omega_1, \quad (3.11)$$

and

$$w^{[k],n+1,p} - u^{n,p} - \Delta t w_{xx}^{[k],n+1,p} + \Delta t u^{n+1,p-1} = \begin{cases} \sum_{l=0}^p \alpha_l u_{xx}^{n+1-l,p-1} & \text{if } (n \geq p-1) \\ \sum_{l=0}^p \alpha_l u_{xx}^{l,p-1} & \text{if } (n < p-1) \end{cases}$$

$$w^{[k],n+1,p}(\alpha) = v^{[k-1],n+1,p}(\alpha), \quad w^{[k],n+1,p}(b) = g_b(t_{n+1}), \quad x \in \Omega_2. \quad (3.12)$$

As before, we set $w^{[0],n+1,p}(\beta) = u^{n,p}(\beta)$, and $v^{[0],n+1,p}(\alpha) = u^{n,p}(\alpha)$ in equations (3.11) and (3.12). In the implementation, (3.11–3.12) are discretized in a similar manner as the prediction steps (3.7–3.8).

3.3. Convergence of the Parallel Schwarz Iteration. The convergence properties of the parallel Schwarz iteration for the prediction step (3.5–3.6) and correction steps (3.11–3.12) follows quite naturally from the typical analysis of the Schwarz iteration for the elliptic problems which result upon discretization in time.

For example, the solution of the prediction step, (3.2), for u^{n+1} , involves a Schwarz iteration for a linear reaction diffusion problem. The solution at the previous time step u^n provides a known source term and does not effect the analysis. Likewise, the terms $u^{n,p}$ and $u_{xx}^{n+1-l,p-1}$ in (3.9) are known non-homogeneous terms.

Denote the errors on Ω_1, Ω_2 by $e_1^{[k]} = u^{n+1} - v^{[k],n+1}$ and $e_2^{[k]} = u^{n+1} - w^{[k],n+1}$ respectively. Subtracting (3.5) and (3.6) from (3.2) we see that the error functions satisfy the BVPs

$$e_1^{[k]} - \Delta t e_{1,xx}^{[k]} = 0, \quad (3.13)$$

$$e_1^{[k]}(a) = 0, \quad e_1^{[k]}(\beta) = e_2^{[k-1]}(\beta),$$

and

$$e_2^{[k]} - \Delta t e_{2,xx}^{[k]} = 0, \quad (3.14)$$

$$e_2^{[k]}(\alpha) = e_1^{[k-1]}(\alpha) \quad e_2^{[k]}(b) = 0.$$

The classical maximum principle [18] for second order elliptic problems immediately gives the required contraction. Alternatively, in this simple case, one can explicitly find expressions for the error functions and show $e_1^{[k]}, e_2^{[k]} \rightarrow 0$ as $k \rightarrow \infty$ for any $\Delta t > 0$. The contraction rate improves as $\Delta t \rightarrow 0$ and as the overlap $\beta - \alpha$ increases. More details can be found in [1, 2]. The error functions for the correction steps also satisfy (3.13–3.14) as can be seen by subtracting (3.11) and (3.12) from (3.9). This suggests that the Schwarz iterations for the prediction and correction steps will converge in approximately the same number of iterations (only 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.

3.4. Starting RIDC. In Figure 3.2, we illustrate graphically the first few steps of the RIDC-DD algorithm. As explained in Section 2.2.2, special care is taken to ensure that minimum memory is used by not letting computing cores/nodes race ahead. Figure 3.2 is an extension of Figure 2.3 for the parallel space time algorithm. This start-up step is needed to fill the memory footprint before the nodes are marched in a pipe as in Figure 3.1.

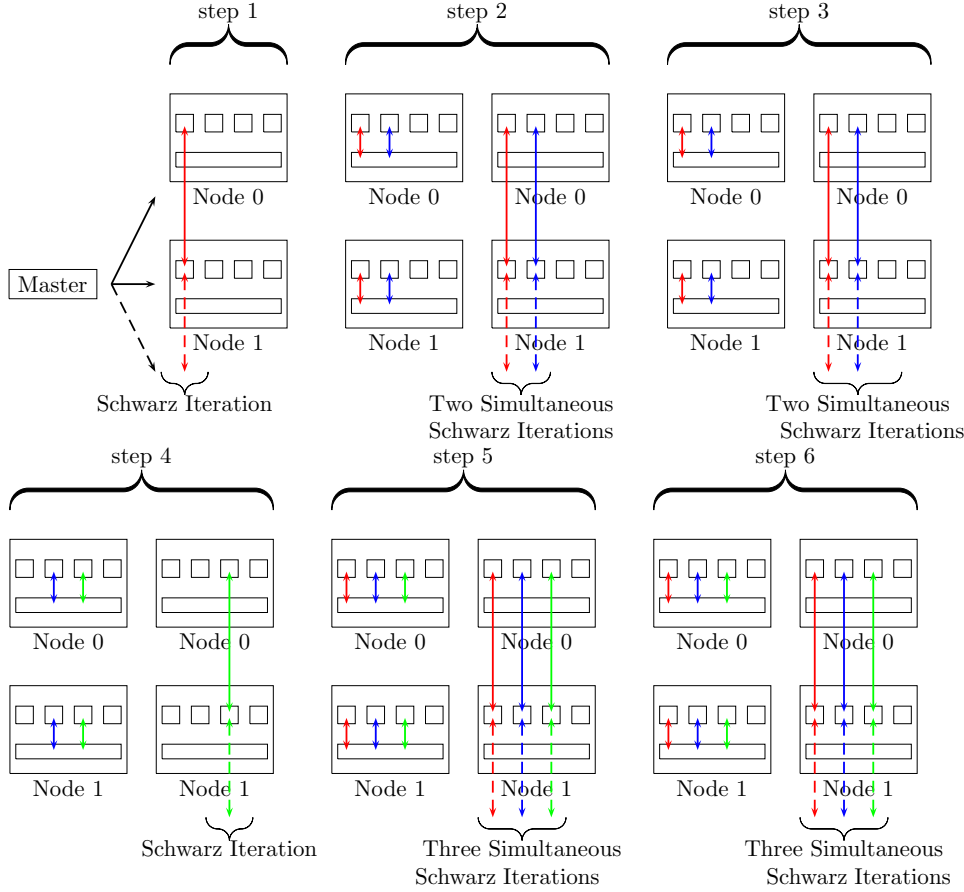


Fig. 3.2: Starting the RIDC framework for time integration with a classical Schwarz iteration in space. Note that in the start-up, some cores sit idle while the memory footprint is being filled. (Note that this figure is an extension of Figure 2.3) As before, observe that each step comprises of two main parts: a local communication with shared memory within a node, followed by the corresponding number of simultaneous Schwarz iterations between nodes.

4. Numerical Examples. We apply our parallel space-time algorithm, RIDC-DD with classical Schwarz iteration in space, to the linear heat equation in two spatial dimensions,

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

The 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. 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 clearly show that our RIDC-DD with classical Schwarz iterations in space converge with the designed orders of accuracy.

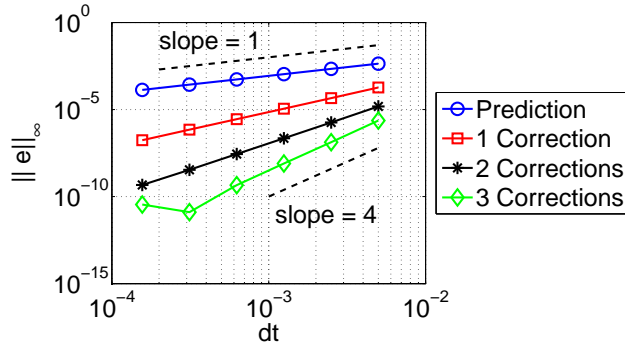


Fig. 4.1: 2nd, 3rd and 4th order RIDC-DD with classical Schwarz iterations in space converge to the reference solution with the designed orders of accuracy. The Schwarz iterations are iterated until a tolerance of 10^{-12} is reached for the predictors and correctors (hence the flat line attained after 3 corrections).

In Figure 4.2, the average number of Schwarz iterations per time step are plotted as a function of Δt for both the prediction and correction Schwarz iterations. Consistent with intuition, the average number of Schwarz iterations decreases with Δt for the predictors and correctors since the initial iterate is more accurate with decreasing Δt and as indicated in Section 3.3 the rate of convergence of the DD iteration improves as $\Delta t \rightarrow 0$. As expected, the predictors and correctors take the same number of Schwarz iterations for each step. This can be explained by observing that the initial iterate for the predictors and correctors always has an $\mathcal{O}(\Delta t)$ error, and the rate of convergence for the Schwarz iteration is dependent on the size of the overlap region and Δt . Furthermore, as mentioned previously, the error equations for the DD iteration for the correction and prediction steps are identical and hence will have the same rate of convergence. Thus, it can be argued that there will be minimal load balancing issues for this RIDC-DD algorithm as formulated. A natural question to ask, is if the initial iterate for the correctors in this RIDC-DD algorithm can be improved to reduce the number of iterations for the correctors? For example, one could choose $w^{[0],n+1,p}(\beta) = u^{n+1,p-1}(\beta)$ and $v^{[0],n+1,p}(\alpha) = u^{n+1,p-1}(\alpha)$ as the initial boundary conditions for equations (3.11–3.12). In lieu of trying to optimize this classical Schwarz matching RIDC-DD algorithm, we choose instead to explore optimal/optimized transmission conditions at the subdomain boundaries and possibly a coarse grid correction [26] to handle the degradation of the DD convergence rate for large numbers of subdomains. Work in this direction is ongoing.

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 4.3 by illustrating a representative convergence study at a fixed instant in the time integration

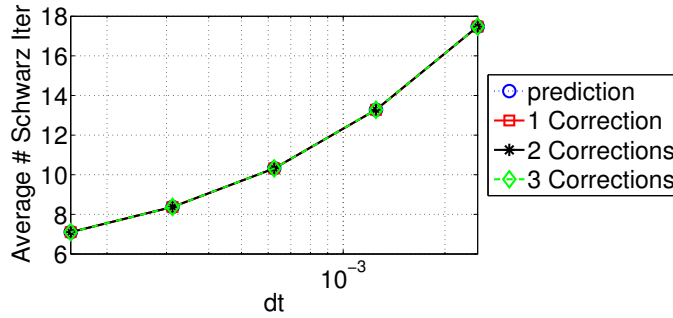


Fig. 4.2: This graph shows the average number of Schwarz iterations per time step for various time step sizes, with a fixed tolerance. Each prediction and correction takes the same number of Schwarz iterations for each step. This is due, in part, to the stopping criterion for the Schwarz iteration being 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.

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, $t = 0.1$.

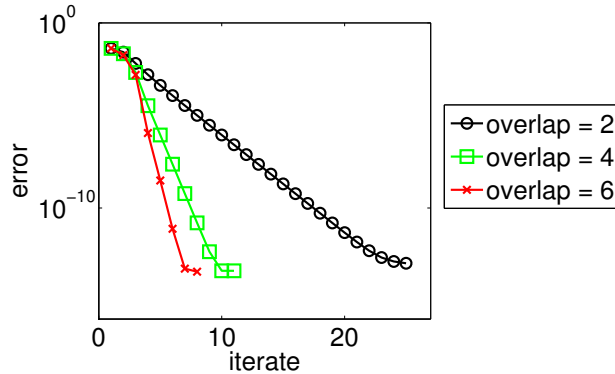


Fig. 4.3: Effect of overlap on the convergence of the Schwarz iteration for prediction and correction steps. As the size of the overlap increases, the rate of convergence of the classical Schwarz iteration increases.

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 a fourth order RIDC method. We show that the proposed algorithm, referred to as RIDC-DD for short, requires four Schwarz iterations per step (after initial start-up costs) that can be evaluated

simultaneously in parallel, provided the appropriate hardware is in place. An ideal architecture for our proposed algorithm has each node handling a subdomain, with each node having four computing cores. Some analysis is presented to demonstrate that the RIDC-DD algorithm will converge, and that the rate of convergence is consistent with results from standard domain decomposition analysis. Numerical experiments demonstrate that the RIDC-DD algorithm attains the designed order of accuracy, and that load balancing appears to be a non-issue 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. Present investigations are ongoing into using optimal/optimized transmission conditions at the subdomain boundaries and a coarse grid correction [26] to handle the degradation of the DD convergence rate for large numbers of subdomains. The authors are also investigating a hybrid OpenMP/MPI implementation to verify the parallel scaling of the algorithm discussed in this paper.

Acknowledgments. This work was supported by AFOSR grant number FA9550-07-1-0092, NSF grant number DMS-0934568, the High Performance Computing Center (HPCC) at Michigan State University, and NSERC Discovery Grant 311796. The authors would also like to thank Dirk Colbry for insightful comments and enlightening discussions related to this work.

REFERENCES

- [1] XIAO-CHUAN CAI, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), pp. 41–61.
- [2] ———, *Multiplicative Schwarz methods for parabolic problems*, SIAM J. Sci. Comput., 15 (1994), pp. 587–603. Iterative methods in numerical linear algebra (Copper Mountain Resort, CO, 1992).
- [3] ANDREW CHRISTLIEB, COLIN MACDONALD, AND BENJAMIN ONG, *Parallel high-order integrators*, SIAM J. Sci. Comput., 32 (2010), pp. 818–835.
- [4] ANDREW CHRISTLIEB, MAUREEN MORTON, BENJAMIN ONG, AND JING-MEI QIU, *Semi-implicit integral deferred correction constructed with additive Runge-Kutta methods*, Commun. Math. Sci., 9 (2011), pp. 879–902.
- [5] ANDREW CHRISTLIEB AND BENJAMIN ONG, *Implicit parallel time integrators*. to appear.
- [6] ANDREW CHRISTLIEB, BENJAMIN ONG, AND JING-MEI QIU, *Comments on high order integrators embedded within integral deferred correction methods*, Comm. Appl. Math. Comput. Sci., 4 (2009), pp. 27–56.
- [7] ———, *Integral deferred correction methods constructed with high order Runge-Kutta integrators*, Math. Comput., 79 (2010), pp. 761–783.
- [8] P. COLELLA, DT GRAVES, TJ LIGOCKI, DF MARTIN, D. MODIANO, DB SERAFINI, AND B. VAN STRAALLEN, *Chombo software package for amr applications-design document*, 2000.
- [9] ALOK DUTT, LESLIE GREENGARD, AND VLADIMIR ROKHLIN, *Spectral deferred correction methods for ordinary differential equations*, BIT, 40 (2000), pp. 241–266.
- [10] M. EMMETT AND M. MINION, *Toward and efficient parallel in time method for partial differential equations*, Journal of Computational Physics, (submitted).
- [11] 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.
- [12] MARTIN J. GANDER, *A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations*, Numerical Linear Algebra with Applications, 6 (1998), pp. 125–145.
- [13] MARTIN J. GANDER, *Optimized Schwarz methods*, SIAM J. Numer. Anal., 44 (2006), pp. 699–731 (electronic).
- [14] M. J. GANDER AND L. HALPERN, *Optimized Schwarz waveform relaxation methods for advection reaction diffusion problems*, SIAM J. Numer. Anal., 45 (2007), pp. 666–697 (electronic).
- [15] M. J. GANDER, L. HALPERN, AND F. NATAF, *Optimized Schwarz methods*, in Twelfth International Conference on Domain Decomposition Methods, Chiba, Japan, Tony Chan, Takashi Kako, Hideo Kawarada, and Olivier Pironneau, eds., Bergen, 2001, Domain Decomposition Press, pp. 15–28.

- [16] 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.
- [17] ELDAR GILADI AND HERBERT B. KELLER, *Space-time domain decomposition for parabolic problems*, Numer. Math., 93 (2002), pp. 279–313.
- [18] DAVID GILBARG AND NEIL S. TRUDINGER, *Elliptic partial differential equations of second order*, Classics in Mathematics, Springer-Verlag, Berlin, 2001. Reprint of the 1998 edition.
- [19] L. HALPERN, *Absorbing boundary conditions and optimized Schwarz waveform relaxation*, BIT, 46 (2006), pp. S21–S34.
- [20] J.L. LIONS, Y. MADAY, AND G. TURINICI, *A “parareal” in time discretization of PDEs*, Comptes Rendus de l’Academie des Sciences Series I Mathematics, 332 (2001), pp. 661–668.
- [21] YVON MADAY AND GABRIEL TURINICI, *The parareal in time iterative solver: a further direction to parallel implementation*, in Domain decomposition methods in science and engineering, vol. 40 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2005, pp. 441–448.
- [22] P. MARDAHL, A. GREENWOOD, T. MURPHY, AND K. CARTWRIGHT, *Parallel performance characteristics of ICEPIC*, 2003.
- [23] CARL RUNGE, *Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten*, Zeit. für Math. und Phys., 46 (1901), pp. 224–243.
- [24] H.A. SCHWARZ, *Über einen grenzübergang durch alternierendes verfahren*, Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15 (1870), pp. 272–286.
- [25] BARRY F. SMITH, PETTER E. BJØRSTAD, AND WILLIAM D. GROPP, *Domain decomposition*, Cambridge University Press, Cambridge, 1996. Parallel multilevel methods for elliptic partial differential equations.
- [26] ANDREA TOSELLI AND OLOF WIDLUND, *Domain decomposition methods—algorithms and theory*, vol. 34 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2005.