Michigan Technological University

Michigan Technological University
Digital Commons @ Michigan Tech

Department of Mathematical Sciences Publications

Department of Mathematical Sciences

1-15-2019

Schwarz waveform relaxation with adaptive pipelining

Felix Kwok
Hong Kong Baptist University

Benjamin W. Ong
Michigan Technological University

Follow this and additional works at: https://digitalcommons.mtu.edu/math-fp

 Part of the Mathematics Commons

Recommended Citation

Kwok, F., & Ong, B. W. (2019). Schwarz waveform relaxation with adaptive pipelining. *SIAM Journal on Scientific Computing, 41*(1), A339-A364. http://dx.doi.org/10.1137/17M115311X
Retrieved from: https://digitalcommons.mtu.edu/math-fp/9

Follow this and additional works at: https://digitalcommons.mtu.edu/math-fp

 Part of the Mathematics Commons

SIAM J. SCI. COMPUT.
Vol. 41, No. 1, pp. A339–A364
© 2019 Society for Industrial and Applied Mathematics

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

SCHWARZ WAVEFORM RELAXATION WITH ADAPTIVE PIPELINING*

FELIX KWOK† AND BENJAMIN W. ONG‡

Abstract. Schwarz waveform relaxation (SWR) methods have been developed to solve a wide range of diffusion-dominated and reaction-dominated equations. The appeal of these methods stems primarily from their ability to use nonconforming space-time discretizations; SWR methods are consequently well-adapted for coupling models with highly varying spatial and time scales. The efficacy of SWR methods is questionable, however, since in each iteration, one propagates an error across the entire time interval. In this manuscript, we introduce an adaptive pipeline approach wherein one subdivides the computational domain into space-time blocks, and adaptively selects the waveform iterates which should be updated given a fixed number of computational workers. Our method is complementary to existing space and time parallel methods, and can be used to obtain additional speedup when the saturation point is reached for other types of parallelism. We analyze these waveform relaxation with adaptive pipelining (WRAP) methods to show convergence and the theoretical speedup that can be expected. Numerical experiments on solutions to the linear heat equation, the advection-diffusion equation, and a reaction-diffusion equation illustrate features and efficacy of WRAP methods for various transmission conditions.

Key words. waveform relaxation, domain decomposition, adaptivity, parallel computing

AMS subject classifications. 65Y05, 65M20

DOI. 10.1137/17M115311X

1. Introduction.

The parallel numerical solution of time-dependent PDEs has long been the focus of the high performance computing community. The classical approach for leveraging high performance computing clusters is to apply a semi-discretization in time to the time-dependent PDE, and then apply grid partitioning or domain decomposition (DD) in space, for which sophisticated and highly efficient methods exist [34]. For highly refined models however, accuracy or stability constraints often limit the size of the time step. The time stepping process, because of its sequential nature, consequently becomes the bottleneck. Hence, parallelization in the time direction has become an increasingly pressing issue, as attested to by the annual conference series in time-parallelization methods (seventh edition as of 2018; see http://parallel-in-time.org/).

One approach for parallelization in time arises from a different way of using DD, the so-called waveform relaxation (WR) approach. Originally, WR methods were developed by [25] for systems of ordinary differential equations (ODEs) that arise in circuit simulation (see also [36]), and subsequently analyzed and extended by many authors; see, for instance, [28, 29, 33, 3, 24, 23, 21, 2]. This approach has also been adapted by the DD community in order to solve time-dependent PDEs, giving rise to Schwarz waveform relaxation (SWR) methods; see [15, 18, 4, 16, 20] and

*Submitted to the journal's Methods and Algorithms for Scientific Computing section October 20, 2017; accepted for publication (in revised form) November 6, 2018; published electronically January 15, 2019.

http://www.siam.org/journals/sisc/41-1/M115311.html

Funding: The first author was partially supported by grants from the Hong Kong Research Grants Council and the National Natural Science Foundation of China.

†Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (felix_kwok@hkbu.edu.hk).

‡Department of Mathematical Sciences, Michigan Technological University, Houghton, MI, 49931 (ongbw@mtu.edu).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

references therein. The SWR idea is to first decompose in space to obtain a collection of (coupled) space-time subproblems, then iterate while exchanging interface information over the whole time window. In fact, one can formally create WR variants out of any stationary iterative method based on DD. For example, the elliptic Neumann–Neumann (NN) and Dirichlet–Neumann (DN) methods can be adapted to yield the NNWR and DNWR methods [22, 27].[1] SWR formulations provide flexibility for discretizing space and time within each space-time subdomain, especially for problems where the dynamics vary greatly; see [20] for an application on ocean-atmospheric coupling. On the other hand, when the dynamics are uniform and DD is used purely for parallelization purposes, the convergence of SWR methods is typically slower than their elliptic counterparts and deteriorates as the time window length T increases [18, 22]. To address the deterioration in convergence, the convergence rate can be monitored and the time window size reduced adaptively if convergence becomes unacceptably slow; see [6]. Despite the deterioration of convergence rate, WR exposes additional opportunities for parallelization, particularly in the time direction. In [30], we presented the technique known as pipelining, in which different waveform iterations of the SWR method can be made to run simultaneously on different time steps, without affecting the mathematical properties of the algorithm. Pipeline parallelism is also possible for NNWR and DNWR relaxation methods [31]. In [14], the authors show that this can lead to a significant reduction in wall-clock time relative to a purely spatial DD implementation for the same total number of processors. Pipeline parallelism was also a popular technique for gaining parallel solution efficiency for ODEs in the WR community; see, for instance, [17, 35].

Another drawback of the basic SWR method is the issue of *oversolving* in the initial time steps. Consider, for example, an initial value problem (P), posed for $t \in [0, T]$ and discretized using a uniform time step $\Delta t = T/N$. This contains as a subproblem the same PDE, but posed on the shorter time interval $t \in [0, T']$ with $T' = M\Delta t$, where $M < N$. Denoting this subproblem by (P′), we observe that any SWR method for the problem (P) must require at least as many iterations to converge as the same SWR method for (P′), at least if the stopping criterion is in terms of an L^p norm. This is because the iterates for (P′) are simply the restrictions of the iterates for (P) over a smaller time window, so convergence for (P) automatically implies convergence for (P′), but usually not the other way around. For SWR methods applied to parabolic problems in particular, it was shown in [18] that the method converges superlinearly, with a rate that depends even more strongly on T than the generic bound in [29] for ODEs. This means for parabolic problems, the error for SWR in the initial time steps is often several orders of magnitude smaller than the error at the final time. Thus, the method is essentially using valuable computational cycles to oversolve the initial time steps relative to the overall tolerance.

In this paper, we address the oversolving problem by presenting a modified version of the pipelining algorithm in [30]; we call this method waveform relaxation with adaptive pipelining (WRAP), because the time window on which the PDE is actively being integrated changes over the duration of the computation. Initially, the method uses a small time window, whose size is determined by the number of available processors. Once a solution in this time window is solved to sufficient accuracy, we accept the solution and stop iterating; instead, we expand the time horizon and reallocate the

[1] In this paper, we will refer to all DD-based WR methods as SWR methods, even when the underlying DD method is not of the Schwarz type, such as the NNWR and DNWR methods.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

processor to solve for a solution at a later time window. We keep doing this until the final time horizon coincides with the original interval $[0, T]$. We describe this method in more detail in section 2. Note that this method is mathematically different from the original WR method because not every time step is iterated the same number of times starting from the same initial and interface conditions. To analyze the convergence of this method, we introduce an error propagation model in section 3, show that error measures satisfying the error propagation model can be derived for the classical SWR and optimized SWR method, and then study the convergence properties of the error propagation model. We also prove an estimate on the theoretical speedup ratio as a function of the number of available processors P. We will see that the average number of iterations required per time step depends on P, but is independent of the time window size, unlike the original WR method. Finally, in section 4 we present numerical results for a variety of diffusive problems and DD methods. The results confirm our theoretical analysis and show that it is possible for a WRAP method to obtain a speedup of at least 5–6 over a purely spatial DD method with sequential time stepping.

2. Algorithms. We start by considering an equivalent formulation of WR algorithms when the time horizon $[0, T]$ is subdivided into shorter intervals. Suppose that the space-time domain, $\Omega \times [0, T]$, is partitioned into space-time subdomains,

$$\{\Omega_1, \Omega_2, \ldots, \Omega_J\} \otimes \{I_1, I_2, \ldots, I_M\},$$

where the spatial partitioning $\{\Omega_1, \Omega_2, \ldots, \Omega_J\}$ can be overlapping or nonoverlapping, with the interfaces denoted by $\Gamma_j = \partial\Omega_j \setminus \partial\Omega$, and the temporal partitioning is $I_m = [T_{m-1}, T_m], m = 1, \ldots, M$. Let $u_{j,m}^{[k]}(x, t)$ denote the kth waveform iterate in $\Omega_j \times I_m$. Additionally, for ease of notation later, we denote the (spatially) distributed solution as $u_m^{[k]}(x, t)$, where

$$u_m^{[k]}(x, t) = \{u_{j,m}^{[k]}(x, t)\}_{j=1}^{J}. \tag{1}$$

Let `integrate` denote a subroutine that computes a numerical approximation to the spatially distributed solution $u_m^{[k]}(x, t)$. Specifically, the routine

$$[g_m^{[k]}, h_m^{[k]}] \leftarrow \texttt{integrate}(I_m, f, g_{m-1}^{[k]}, h_m^{[k-1]}),$$

takes as its input

- the interval of integration, $I_m = [T_{m-1}, T_m]$;
- boundary conditions for the PDE, f, on $\partial\Omega$;
- the (distributed) solution at the start of the time interval, $g_{m-1}^{[k]} = u_m^{[k]}(x, T_{m-1})$;
- the (time-dependent) coupling conditions, $h_m^{[k-1]}$;

and returns as its output

- the (distributed) solution at the end of the time interval, $g_m^{[k]} = u_m^{[k]}(x, T_m)$;
- the updated (time-dependent) coupling conditions, $h_m^{[k]}$.

For example, in a classical SWR implementation, the coupling conditions, $h_m^{[k]} = \{h_{j,m}^{[k]}\}_{j=1}^{J}$ would be the set of Dirichlet interface conditions required to solve the PDE on $\{\Omega_j\} \times I_m$, i.e., $h_{j,m}^{[k]} = u_{j,m}^{[k]}|_{\Gamma_j \times I_m}$. The classical SWR computation proceeds according to Algorithm 1. In Algorithm 1, we have split the integration over $[0, T]$ into a sequence of shorter integration steps over I_m, $m = 1, \ldots, M$, and computed K

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 1 Classical SWR.

for $k = 1$ to K **do** ▷ for each waveform iterate
  for $m = 1$ to M **do** ▷ for each time block
    if $m = 1$ **then**
      set $g_0^{[k]} = u_0(x)$ ▷ utilize initial condition
    end if
    if $k = 1$ **then**
      specify $h_m^{[0]}$ ▷ guess initial coupling condition
    end if
    $[g_m^{[k]}, h_m^{[k]}] \leftarrow \text{integrate}(I_m, f, g_{m-1}^{[k]}, h_m^{[k-1]})$ ▷ integrate solution over I_m
  end for
end for

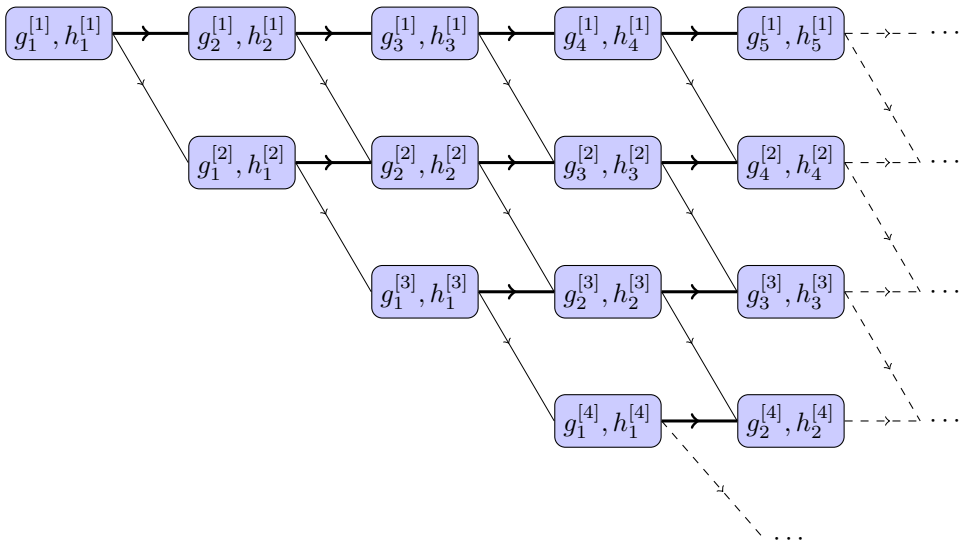

FIG. 1. *Dependency graph for SWR. The variables within the purple boxes denote the outputs of the* `integrate` *routine. The width of the arrows reflects the amount of information that needs to be passed to the newly spawned tasks. If the execution of each task (purple box) takes roughly the same wall time, each column of tasks can be simultaneously computed if sufficient processors are available.*

waveform iterates. Pipeline parallelism is now possible [30], because multiple tasks (i.e., multiple `integrate` routine calls) can be launched if the required input data is available. For example, the completion of

$$[g_1^{[1]}, h_1^{[1]}] \leftarrow \texttt{integrate}(I_1, f, g_0^{[1]}, h_1^{[0]})$$

provides the required input for *two* `integrate` function calls,

$$[g_2^{[1]}, h_2^{[1]}] \leftarrow \texttt{integrate}(I_2, f, g_1^{[1]}, h_2^{[0]}),$$
$$[g_1^{[2]}, h_1^{[2]}] \leftarrow \texttt{integrate}(I_1, f, g_0^{[2]}, h_1^{[1]}).$$

More generally, a dependency graph can be generated to identify tasks that can be run in parallel. In Figure 1, the output of each `integrate` routine is shown in the purple boxes. Tasks belonging to the same column can all be run concurrently, provided enough processors are available. This pipeline works best if the execution of each task (i.e., purple box) takes roughly the same wall time.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Algorithm 2 Pipeline SWR.

```
1: tasklist = {}                                          ▷ initialize tasklist as empty
2: tasklist.append({1,1})                                 ▷ add first task
3: while not_empty(tasklist) do
4:     parfor p = 1 to size(tasklist) do
5:         task = tasklist.get(p)                         ▷ task: pth entry of tasklist
6:         k = task.k; m = task.m;                        ▷ set (k,m) from task
7:         if m = 1 then
8:             set g_0^[k] = u_0(x)                       ▷ utilize initial condition
9:         end if
10:        if k = 1 then
11:            specify h_m^[0]                            ▷ guess initial coupling condition
12:        end if
13:        [g_m^[k], h_m^[k]] ← integrate(I_m, f, g_{m-1}^[k], h_m^[k-1])   ▷ Integrate solution over I_m
14:        tasklist.remove(p)                             ▷ Remove pth entry from tasklist
15:        if k = 1 and m < M then
16:            tasklist.append({k, m+1})                  ▷ advance if not final time block
17:        end if
18:        if k < K then
19:            tasklist.append({k+1, m})                  ▷ compute next waveform iterate
20:        end if
21:    end parfor
22: end while
```

The pipeline parallel SWR computation can be implemented using a `tasklist`, which is a list[2] of tuples (k, m), corresponding to the solution values $(g_m^{[k]}, h_m^{[k]})$ that can presently be computed because the dependencies are satisfied. The pipeline SWR algorithm is given in Algorithm 2. A couple of observations are in order: `integrate` will be called $K \cdot M$ times, similarly to the classical SWR implementation. Second, the order in which tasks in `tasklist` are executed does not matter.

One way to save computation is to *prune* the dependency graph and remove tasks that are either unnecessary or ineffective in reducing the error in the solution. To accomplish this pruning, we propose an adaptive framework that utilizes two key ideas. First, suppose for example, that the error associated with computing $u_1^{[2]}$ satisfies some user prescribed tolerance. Then, one can stop iterating on time interval I_1 and use the converged solution at the end of this interval to spawn any future task involving interval I_2, thereby reducing the total number of tasks within each column. An example of this modified dependency graph is shown in Figure 2. More generally, one can utilize the `integrate` routine to return $(g_m^{[k]}, h_m^{[k]})$ given $(g_{m-1}^{[j]}, h_m^{[k-1]})$, where $j \leq k$, i.e.,

$$[g_m^{[k]}, h_m^{[k]}] \leftarrow \texttt{integrate}(I_m, f, g_{m-1}^{[j]}, h_m^{[k-1]}), \text{ where } j \leq k.$$

Second, if $g_{m-1}^{[k]}$ is so inaccurate that further iteration in $I_m, I_{m+1}, \ldots$ would not lead to a significant reduction in error, then it is advantageous to wait until a more accurate solution $g_{m-1}^{[j]}$, $j > k$, becomes available, and use that as the starting value for further integration. In Figure 3, two iterations are performed in I_2 before we begin

[2]This list can be implemented as a hash map for efficiency.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

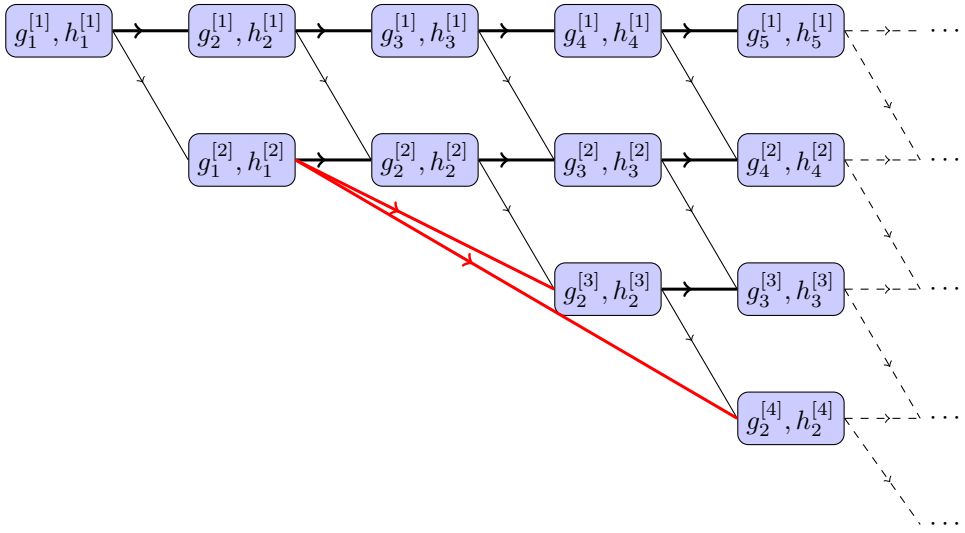

FIG. 2. *Dependency graph for SWR if the error associated with $u_1^{[2]}$ satisfies some user prescribed tolerance. The new dependencies are shown in red.*

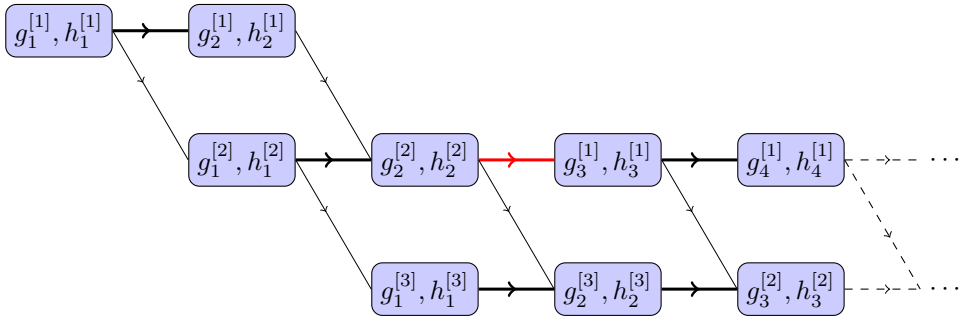

FIG. 3. *Dependency graph for SWR if two iterations are performed in I_2 before we begin iterating in I_3, i.e., $g_2^{[2]}$ is used instead of $g_2^{[1]}$ to compute $g_3^{[1]}$. Each column has only two tasks, i.e., only two time-parallel tasks can be simultaneously computed ($ntasks = 2$).*

iterating in I_3, i.e., $g_2^{[2]}$ is used instead of $g_2^{[1]}$ to compute $g_3^{[1]}$. In other words, we have shifted everything to the right of $(g_2^{[1]}, h_2^{[1]})$ downward and to the right and changed the dependencies, as shown in red in Figure 3. More generally, one can utilize the `integrate` routine to return $(g_m^{[k]}, h_m^{[k]})$ given $(g_{m-1}^{[j]}, h_m^{[k-1]})$, where $j \geq k$, i.e.,

$$[g_m^{[k]}, h_m^{[k]}] \leftarrow \texttt{integrate}(I_m, f, g_{m-1}^{[j]}, h_m^{[k-1]}), \text{ where } j \geq k.$$

This transformation changes the mathematical properties of the WR algorithm, and new convergence estimates must be proved, which we will do in section 3.

We are now ready to present the WRAP method in Algorithm 3. We begin by noting the key differences between Algorithms 2 and 3.

1. In the while-loop block in Algorithm 2, lines 4–21, the pipeline SWR algorithm completes every task in `tasklist`. This corresponds to concurrently executing every purple task in a column of Figure 1. Suppose instead that *ntasks* is the maximum number of "parallel-in-time" tasks that we wish to ex-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

ecute simultaneously by our machine.[3] In Algorithm 3 line 5, at most *ntasks* are performed within each while-loop block. What do we do if there are more than *ntasks* elements in `tasklist`? To choose which tasks in `tasklist` to execute, we use the heuristic that *more accurate initial conditions always lead to faster error reduction*: we select from the list *ntasks* elements with the smallest m, i.e., corresponding to the earliest time intervals. Thus, tasks with larger m will be *delayed* until the solution at earlier time intervals has converged. This is implemented in line 4 of Algorithm 3, where the tasks in `tasklist` are sorted according to m.

2. We only compute additional waveform iterates in I_m if the coupling conditions have not converged, line 19 in Algorithm 3.
3. The most accurate coupling conditions, h_m and initial conditions for each interval, g_m, are used and stored in line 14 of Algorithm 3. This is in contrast to line 13 of Algorithm 2, where specific waveform iterates are used and stored.

Algorithm 3 SWR with adaptive pipelining.

```
1:  tasklist = {}                                          ▷ initialize tasklist as empty
2:  tasklist.append({1,1})                                 ▷ add first task
3:  while not_empty(tasklist) do
4:      tasklist.sort.m;                                   ▷ sort tasklist using the variable m
5:      parfor p = 1 to min(size(tasklist), ntasks) do
6:          task = tasklist.get(p);                        ▷ task: pth entry of tasklist
7:          k = task.k; m = task.m;                        ▷ set (k,m) from task
8:          if m = 1 then
9:              set g_0 ← u_0(x)                           ▷ utilize initial condition
10:         end if
11:         if k = 1 then
12:             specify h_m                                ▷ guess initial coupling condition
13:         end if
14:         [g_m, h_m] ← integrate(I_m, f, g_{m-1}, h_m)   ▷ Integrate solution over I_m
15:         tasklist.remove(p)                             ▷ Remove pth entry from tasklist
16:         if k = 1 and m < M then
17:             tasklist.append({k, m + 1})                ▷ advance if not final time block
18:         end if
19:         if coupling condition not converged then
20:             tasklist.append({k + 1, m})                ▷ compute next waveform iterate
21:         end if
22:     end parfor
23: end while
```

There are two limiting cases of interest in Algorithm 3. If $ntasks = 1$ and the time window $I_m = [T_{m-1}, T_m]$ consists of a single time step, Δt, then the WRAP framework simplifies to a classical DD method, where $g_m^{[k]}$ is iterated to convergence before computing $g_{m+1}^{[1]}$. The second limiting case is when *all* the tasks in `tasklist` are simultaneously computed before a new task list is generated based on the recently

[3] It is envisioned that each parallel-in-time task executes on a spatially distributed solution, (1). If a hybrid MPI-OpenMP framework is used to implement the adaptive WR methods, *ntasks* can be initialized to the number of processing cores available on each socket. Hence, the task-based time parallelism is accomplished using OpenMP and the distributed spatial parallelism using MPI.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

completed tasks. We shall denote this as $ntasks = \infty$, with the understanding that the maximum number of simultaneous tasks that can be computed is limited by the number of time steps used in the discretization. In this case, WRAP produces iterates that are *the same to those of classical SWR*, Algorithms 1 and 2, up to the preset tolerance `TOL`, since the dependency graph is identical.

3. Convergence analysis. To understand the convergence properties of the WRAP method, we first introduce an error propagation model that is valid for both nonadaptive and adaptive SWR methods. Consider again the dependency graph for nonadaptive SWR, shown in Figure 1. Let $G(m,k)$ and $H(m,k)$ be error measures related to the iterates $g_m^{[k]}$ and $h_m^{[k]}$, which must be suitably defined according to the problem and method chosen. In general, one should choose $G(m,k)$ to be the maximum error in $g_m^{[k]}$. Similarly, $H(m,k)$ should be the maximum error in the interface conditions over the time window $[T_{m-1}, T_m]$. In other words, for well-posed problems, $G(m,k)$ and $H(m,k)$ should be chosen so that

$$G(m,k) = H(m,k) = 0 \implies u_{j,m}^{[k]} = u|_{\Omega_j \times I_m}.$$

Our error propogation model will be based on a system of coupled recurrence equations

$$\begin{cases} G(m,k) \le \alpha G(m-1,k) + H(m,k), \\ H(m,k+1) \le G(m-1,k) + \beta H(m,k), \end{cases} \tag{2}$$

where α and β are constants, with $\beta < 1$. The constant α measures amplification or decay of the error in the initial condition for each time window, assuming *no error in the coupling conditions*. This constant can be greater than 1 for unstable problems. The constant β measures the contraction of the error in the interface conditions *when the initial conditions are exact*; this is a property of the Schwarz WR method, and must be less than 1 *in some appropriate norm* if the original method converges. However, the exact error norm that must be chosen in order to achieve $\beta < 1$ depends on both the problem and the method chosen.

The remainder of the section is structured as follows. In subsection 3.1, we illustrate how error measures satisfying (2) can be identified for two representative SWR methods: the classical SWR method with subdomain problems posed in the continuous setting, and an optimized SWR method with Robin interface conditions, using a P^r finite element discretization in space and the theta method in time. For ease of presentation, we present the analysis for the linear heat equation; the techniques are similar for other parabolic problems, but the analysis is more involved. These two methods are chosen to show that the model (2) can accommodate a variety of problems: the system to be solved can be continuous or fully discrete, and the main argument can be based on either the maximum principle or energy estimates. In subsections 3.2 and 3.3, model (2) is used to show that both the nonadaptive and adaptive pipeline methods will converge. In particular, we show that for each space-time subdomain, $G(m,k) \to 0$ and $H(m,k) \to 0$ as $k \to \infty$ for both methods as long as $\beta < 1$. Finally, in subsection 3.4, the theoretical speedup that can be achieved by the WRAP method is derived.

3.1. Error propagation for selected SWR methods. In the next two theorems, we show that for the linear heat equation, the error propagation model (2) is valid for both classical SWR and optimized SWR with Robin transmission conditions,

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

provided we choose the error measures correctly. Since the heat equation is linear, it suffices to consider the homogeneous problem with an arbitrary initial guess along the artificial interfaces.

THEOREM 3.1. *Consider the classical SWR applied to the homogeneous heat equation,*

$$\partial_t u_j^{[k]} - \Delta u_j^{[k]} = 0, \quad u_j^{[k]}\Big|_{t=T_0} = 0,$$

with initial guesses on the artificial interfaces $\partial\Omega_j \setminus \partial\Omega$, $j = 1, \ldots, J$. *Denote the time subintervals by* $I_1, \ldots, I_M$, *where* $I_m = [T_{m-1}, T_m]$, $m = 1, 2, \ldots, M$. *If*

$$G(m,k) = \max_j \|u_j^{[k]}(\cdot, T_m)\|_{L^\infty(\Omega_j)},$$
$$H(m,k) = \max_j \left(\sup_{t \in I_m} \|u_j^{[k]}(\cdot, t)\|_{L^\infty(\partial\Omega_j)} \right),$$

then $\{G(m,k)\}_{k,m\geq 1}$ *and* $\{H(m,k)\}_{k,m\geq 1}$ *satisfy the recurrence* (2) *for some* $0 < \alpha < 1$ *and* $0 < \beta < 1$.

Proof. We consider the solution at the kth iteration inside the space-time subdomain $(x,t) \in \Omega_j \times I_m$. The solution satisfies $\partial_t u_j^{[k]} - \Delta u_j^{[k]} = 0$ with initial and boundary conditions

$$\|u_j^{[k]}(\cdot, T_{m-1})\|_{L^\infty(\Omega_j)} \leq G(m-1,k), \quad \|u_j^{[k]}(\cdot, t)\|_{L^\infty(\partial\Omega_j)} \leq H(m,k) \quad \forall t \in I_m.$$

Since the PDE is linear, it suffices to estimate $G(m,k)$ by first setting $H(m,k) = 0$, then estimating $G(m,k)$ by setting $G(m-1,k) = 0$, and finally adding the two estimates together. The same procedure can be applied to estimate $H(m,k+1)$. Thus, we first consider the subdomain problem with zero interface conditions

$$\partial_t u_j^{[k]} - \Delta u_j^{[k]} = 0, \quad u_j^{[k]}\Big|_{t=T_{m-1}} = 1, \quad u_j^{[k]}\Big|_{\partial\Omega_j} = 0.$$

By the maximum principle, we have

$$0 \leq u_j^{[k]}(x,t) \leq \alpha_j(t) < 1 \quad \forall\, (x,t) \in \Omega_j \times I_m.$$

In anticipation of showing convergence of the solution $u_j^{[k]}(x, T_m)$ at the end of the time interval I_m, we define

$$\alpha_j := \alpha_j(T_m), \quad \alpha := \max_j \alpha_j < 1.$$

Note that although α depends on the length of the time interval I_m and on the diameter of the subdomains, such an α always exists.

Next, if we consider the subdomain problem with zero initial conditions,

$$\partial_t u_j^{[k]} - \Delta u_j^{[k]} = 0, \quad u_j^{[k]}\Big|_{t=T_{m-1}} = 0, \quad u_j^{[k]}\Big|_{\partial\Omega_j} = 1,$$

we get trivially that

$$0 \leq u_j^{[k]}(x,t) \leq 1 \quad \forall (x,t) \in \Omega_j \times I_m.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

However, on a set $\Gamma \subset \Omega_j$ that is at a distance of at least δ away from $\partial\Omega_j$, we in fact have [16, Lemma 3.1]

$$u_j^{[k]}(\cdot,t)\|_{L^\infty(\Gamma)} \le \beta < 1,$$

where β depends on the distance δ. Thus, for the general problem $\partial_t u_j^{[k]} - \Delta u_j^{[k]} = 0$ with

$$|u_j^{[k]}(x,T_{m-1})| \le G(m-1,k) \quad \forall x \in \Omega_j,$$
$$|u_j^{[k]}(x,t)| \le H(m,k) \quad \forall (x,t) \in \partial\Omega_j \times I_m,$$

we have $|u_j^{[k]}(\cdot,t)| \le \alpha_j(t)G(m-1,k) + H(m,k)$, which leads to

$$|u_j^{[k]}(\cdot,T_m)| \le \alpha G(m-1,k) + H(m,k). \tag{3}$$

However, the Dirichlet values transmitted to the neighbors of Ω_j lie in a set Γ at least δ away from $\partial\Omega_j$, so we have the estimate

$$\|u_j^{[k]}(\cdot,t)\|_{L^\infty(\Gamma)} \le G(m-1,k) + \beta H(m,k) \quad \forall t \in I_m. \qquad \square$$

For optimized SWR, we have the following result if we use P^r finite elements for the spatial discretization and the theta method with $\frac{1}{2} \le \theta \le 1$ for discretization in time. For simplicity, we assume that each time block consists of a single time step, and that the spatial decomposition is nonoverlapping with no cross points. We denote by $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$ the interface between Ω_i and Ω_j.

THEOREM 3.2. *Consider the optimized SWR applied to the homogeneous heat equation discretized with the theta method in time and P^r finite elements in space with $r \ge 1$ over a shape regular, quasi-uniform triangulation $\mathcal{T}_h$. More precisely, let $u_{jm}^{[k]} \approx u_j^{[k]}(\cdot,T_m)$ satisfy*

$$\int_{\Omega_j} v\left(\frac{u_{jm}^{[k]} - u_{j,m-1}^{[k]}}{\Delta t_m}\right) + \int_{\Omega_j} \nabla\bar{w}_{jm}^{[k]} \cdot \nabla v + \int_{\partial\Omega_j \setminus \partial\Omega} p\bar{w}_{jm}^{[k]} v = \int_{\partial\Omega_j \setminus \partial\Omega} R_{jm}^{[k]} v \quad \forall v \in V_j^h, \tag{4}$$

$$R_{jm}^{[k+1]}|_{\Gamma_{ij}} = (2p\bar{w}_{im}^{[k]} - R_{im}^{[k]})|_{\Gamma_{ij}}, \tag{5}$$

where $\Delta t_m = T_m - T_{m-1}$, $\bar{w}_{jm}^{[k]} = (1-\theta)u_{j,m-1}^{[k]} + \theta u_{jm}^{[k]}$ with $\frac{1}{2} \le \theta \le 1$, and the initial Robin traces $R_{jm}^{[1]}$ are posed on the artificial interfaces $\partial\Omega_j \setminus \partial\Omega$, $j = 1,\dots,J$; cf. [9]. *If*

$$G(m,k) = \left(\frac{1}{2}\sum_j \|u_{jm}^{[k]}\|^2_{L^2(\Omega_j)}\right)^{1/2}, \quad H(m,k) = \left(\Delta t_m \sum_j \|R_{jm}^{[k]}\|^2_{L^2(\partial\Omega_j\setminus\partial\Omega)}\right)^{1/2},$$

then $\{G(m,k)\}_{k,m\ge1}$ and $\{H(m,k)\}_{k,m\ge1}$ satisfy the recurrence (2) *for $\alpha = 1$ and some $0 < \beta < 1$, where β depends on the length of the time step size Δt_m.*

Proof. Let $v = \bar{w}_{jm}^{[k]}$ in (4) and calculate

$$\frac{1}{2\Delta t_m}\int_{\Omega_j}\left[(u_{jm}^{[k]})^2 - (u_{j,m-1}^{[k]})^2 + (2\theta-1)(u_{jm}^{[k]} - u_{j,m-1}^{[k]})^2\right] + \int_{\Omega_j}|\nabla\bar{w}_{jm}^{[k]}|^2$$
$$= \int_{\partial\Omega_j\setminus\partial\Omega}(R_{jm}^{[k]} - p\bar{w}_{jm}^{[k]})\bar{w}_{jm}^{[k]} = \int_{\partial\Omega_j\setminus\partial\Omega}\left[(R_{jm}^{[k]})^2 - (2p\bar{w}_{jm}^{[k]} - R_{jm}^{[k]})^2\right]. \tag{6}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Using the update formula (5) and the fact that $2\theta - 1 \geq 0$, we obtain, after summing over all subdomains Ω_j, that

$$\frac{1}{2}\sum_j \|u_{jm}^{[k]}\|_{L^2(\Omega_j)}^2 + \Delta t_m \sum_j \|R_{jm}^{[k+1]}\|_{L^2(\partial\Omega_j\setminus\partial\Omega)}^2 \leq \frac{1}{2}\sum_j \|u_{j,m-1}^{[k]}\|_{L^2(\Omega_j)}^2 + \Delta t_m \sum_j \|R_j^{[k]}\|_{L^2(\partial\Omega_j\setminus\partial\Omega)}^2.$$

In other words, we have

$$G(m,k)^2 + H(m,k+1)^2 \leq G(m-1,k)^2 + H(m,k)^2,$$

which immediately implies the recurrence relation (2) with $\alpha = \beta = 1$. To see that β can in fact be chosen to be less than 1, it suffices by linearity to consider the case where $u_{j,m-1}^{[k]} = 0$ for all j and show that $H(m,k+1) \leq \beta H(m,k)$ for some $\beta < 1$.

We proceed by substituting $u_{j,m-1}^{[k]} = 0$ into (4), so that $\bar{w}_{jm}^{[k]} = \theta u_{jm}^{[k]}$:

$$\int_{\Omega_j} \frac{u_{jm}^{[k]} v}{\Delta t_m} + \theta\left(\int_{\Omega_j} \nabla u_{jm}^{[k]} \cdot \nabla v + \int_{\partial\Omega_j\setminus\partial\Omega} p u_{jm}^{[k]} v\right) = \int_{\partial\Omega_j\setminus\partial\Omega} R_{jm}^{[k]} v. \tag{7}$$

By Lemma 4.10 in [34] and Theorem 4.5.11 in [5], there exists a discrete harmonic extension $v \in V_j^h$ of $R_{jm}^{[k]}$, such that $v|_{\partial\Omega_j\setminus\partial\Omega} = R_{jm}^{[k]}$ and

$$\|v\|_{H^1(\Omega_j)} \leq C\|R_{jm}^{[k]}\|_{H^{1/2}(\partial\Omega_j\setminus\partial\Omega)} \leq Ch^{-1/2}\|R_{jm}^{[k]}\|_{L^2(\partial\Omega_j\setminus\partial\Omega)}.$$

Substituting this v into (7) and using the Cauchy–Schwarz inequality on the left, we obtain

$$\begin{aligned}
\int_{\partial\Omega_j\setminus\partial\Omega} (R_{jm}^{[k]})^2 &\leq \frac{1}{\Delta t_m}\|u_{jm}^{[k]}\|_{L^2(\Omega_j)}\,\|v\|_{L^2(\Omega_j)} + \theta|u_{jm}^{[k]}|_{H^1(\Omega_j)}\,|v|_{H^1(\Omega_j)} \\
&\quad + \theta p\|u_{jm}^{[k]}\|_{L^2(\partial\Omega_j\setminus\partial\Omega)}\,\|R_{jm}^{[k]}\|_{L^2(\partial\Omega_j\setminus\partial\Omega)} \\
&\leq \left(\frac{\theta}{\Delta t_m}\|u_{jm}^{[k]}\|_{L^2(\Omega_j)}^2 + \theta^2|u_{jm}^{[k]}|_{H^1(\Omega_j)}^2\right)^{1/2} \\
&\quad \times \left(\frac{1}{\theta\Delta t_m}\|v\|_{L^2(\Omega_j)}^2 + |v|_{H^1(\Omega_j)}^2\right)^{1/2} \\
&\quad + \theta p\|u_{jm}^{[k]}\|_{L^2(\partial\Omega_j\setminus\partial\Omega)}\,\|R_{jm}^{[k]}\|_{L^2(\partial\Omega_j\setminus\partial\Omega)} \\
&\leq \left(\frac{\theta}{\Delta t_m}\|u_{jm}^{[k]}\|_{L^2(\Omega_j)}^2 + \theta^2|u_{jm}^{[k]}|_{H^1(\Omega_j)}^2\right)^{1/2} \\
&\quad \times \left(\frac{C_1}{\sqrt{\theta h\Delta t_m}} + C_2 p\right)\|R_{jm}^{[k]}\|_{L^2(\partial\Omega_j\setminus\partial\Omega)}.
\end{aligned}$$

Dividing both sides by $\|R_{jm}^{[k]}\|_{L^2(\partial\Omega_j\setminus\Omega)}$, we see that

$$\int_{\partial\Omega_j\setminus\partial\Omega} (R_{jm}^{[k]})^2 \leq \bar{C}\left(\frac{\theta}{\Delta t_m}\|u_{jm}^{[k]}\|_{L^2(\Omega_j)}^2 + \theta^2|u_{jm}^{[k]}|_{H^1(\Omega_j)}^2\right),$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

where $\bar{C} > 1$ depends on Δt_m, h, and p. Substituting into (6), and keeping in mind the assumption that $u_{j,m-1}^{[k]} = 0$, we deduce that

$$\int_{\partial\Omega_j \setminus \partial\Omega} \left[(R_{jm}^{[k]})^2 - (2p\bar{w}_{jm}^{[k]} - R_{jm}^{[k]})^2 \right] = \frac{\theta}{\Delta t_m} \int_{\Omega_j} (u_{jm}^{[k]})^2 + \theta^2 \int_{\Omega_j} |\nabla u_{jm}^{[k]}|^2 \geq \bar{C}^{-1} \int_{\partial\Omega_j \setminus \partial\Omega} (R_{jm}^{[k]})^2.$$

We conclude that

$$\int_{\partial\Omega_j \setminus \partial\Omega} (2p\bar{w}_{jm}^{[k]} - R_{jm}^{[k]})^2 \leq (1 - \bar{C}^{-1}) \int_{\partial\Omega_j \setminus \partial\Omega} (R_{jm}^{[k]})^2,$$

so summing over all j shows that $H(m, k+1) \leq \beta H(m,k)$ with $\beta = 1 - \bar{C}^{-1} < 1$, as required. ∎

3.2. The nonadaptive case. We now illustrate how the error propagation model (2) can be used to derive error estimates for the corresponding SWR method. We choose classical SWR as an example; the case of optimized SWR can be derived similarly. Note that this is only a linear estimate and is less sharp than the estimate in [16], but the linear estimate is much more amenable to our later analysis for the adaptive case, when the dependency graph no longer resembles Figure 1.

LEMMA 3.3. *Consider classical SWR with*

$$u_j^{[k]}(x, T_0) = 0 \quad \textit{and} \quad \|u_j^{[1]}(\cdot, t)\|_{L^\infty(\partial\Omega_j)} \leq 1$$

for all j. Let $\xi \geq 1$ and $\eta > \beta > 0$ be constants that satisfy $(\xi - \alpha)(\eta - \beta) = 1$. Then

$$|u_j^{[k]}(x,t)| \leq G(m-1,k) + H(m,k) \quad \textit{on} \quad \Omega_j \times [T_{m-1}, T_m],$$

where the functions $G(m,k)$ and $H(m,k)$ are defined in Theorem 3.1*, and satisfy*

$$H(m,k) \leq \xi^{m-1}\eta^{k-1}, \tag{8}$$

$$G(m,k) \leq (\eta - \beta)\xi^m \eta^{k-1}. \tag{9}$$

Proof. Since $\xi \geq 1$ and $H(m,1) \leq 1$ by definition, we see that (8) holds for $k = 1$. Moreover, since $(\eta - \beta)\xi = 1 + \alpha(\eta - \beta) > 1$, (3) implies

$$G(1,k) \leq H(1,k) \leq \eta^{k-1} \leq (\eta - \beta)\xi\eta^{k-1},$$

which proves (9) for $m = 1$. We now prove (8) and (9) by induction on m and k using the recurrence (2). Indeed, we have

$$H(m,k+1) \leq G(m-1,k) + \beta H(m,k) \leq (\eta - \beta)\xi^{m-1}\eta^{k-1} + \beta\xi^{m-1}\eta^{k-1} = \xi^{m-1}\eta^k.$$

Moreover,

$$G(m,k) \leq \alpha G(m-1,k) + H(m,k) \leq (\alpha(\eta - \beta) + 1)\xi^{m-1}\eta^{k-1} = (\eta - \beta)\xi^m\eta^{k-1},$$

as $1 = (\xi - \alpha)(\eta - \beta)$. We have thus proved (8) and (9) inductively, as required. ∎

Note that there is some flexibility in choosing ξ and η, as long as the constraint $(\xi - \alpha)(\eta - \beta) = 1$ is satisfied. One example is

$$\xi = \frac{1+\alpha}{1-\beta} > 1, \quad \eta = \frac{1+\alpha\beta^2}{1+\alpha\beta} < 1.$$

We see from Lemma 3.3 that $H(m,k)$ converges to zero as $k \to \infty$ for fixed m, but the constant increases with m. One can choose an η arbitrarily close to, but larger than, β, but one must then live with the growth in m that comes from a large ξ.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

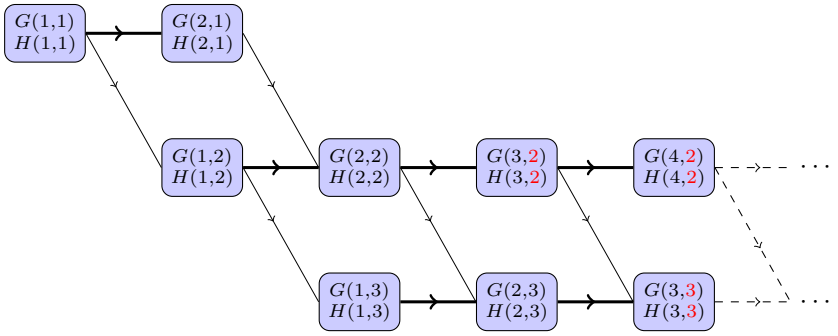

FIG. 4. *This figure gives the dependency graph for the same iterative process previously shown in Figure 3, but with new labels for k, and where $\{G(m,k), H(m,k)\}$ in each task denotes the error measures related to the iterates $g_m^{[k]}$ and $h_m^{[k]}$, respectively. The new labels, k, are related to the old labels, $\tilde{k}$, by the relation $k = \tilde{k} + D_m$, where D_m is the delay in starting the method for the mth time interval because processors are not available to complete this task. In this example, the solution in I_2 is iterated twice before the computation in I_3 is initiated. Hence, we have $D_1 = D_2 = 0$, $D_3 = D_4 = 1$. The new labels are shown in red.*

3.3. Adaptive case. To analyze the adaptive case, we start by referring to the dependency graph in Figure 3. To facilitate the analysis, it is more convenient to label each task in a row with the same iteration number k; thus, from now on we redefine the iteration number k as in Figure 4.

Let $\tilde{k}$ be the old label. The old and new labels are related by $k = \tilde{k} + D_m$, where D_m is the *delay* in starting the method for the mth time interval because processors are not available to compute the mth interval. This delay does not include the "burn-in" time, i.e., the amount of time waiting for appropriate initial or boundary conditions to begin the computation on the mth time interval. For the adaptive SWR, for instance, we have

$$G(m,k) = G(m, \tilde{k} + D_m) = \max_j \|g_{j,m}^{[k]}\|_{L^\infty(\Omega_j)}.$$

For convenience, we will let $P = ntasks$, the number of time-parallel tasks that can be executed simultaneously. The delay D_m has the following properties:

- $D_m \le D_{m+1}$ for all m;
- if $P \ge 1$ time-parallel tasks can be run simultaneously, then $D_1 = \cdots = D_P = 0$. This is because the first P time intervals always have priority over later times in the task list.

With the new numbering, our computational model (2) becomes

$$\begin{cases} G(m,k) \le \alpha G(m-1,k) + H(m,k), & k > D_m, \\ H(m,k+1) \le G(m-1,k) + \beta H(m,k), & k > D_m, \\ H(m,k) \le 1, & k \le D_m. \end{cases} \tag{10}$$

The last condition simply indicates that there can be no reduction of error in the interface conditions until the method starts iterating on the interval I_m. To solve (10), we need the following lemma, whose proof is identical to that of Lemma 3.3.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

LEMMA 3.4. *Let $\xi \geq 1$ and $\eta > \beta > 0$ be constants that satisfy $(\xi-\alpha)(\eta-\beta) = 1$. Let $A_r = A_r(\xi,\eta)$, $r \geq 1$, be any nonnegative function of ξ and η such that*

$$\xi^{m-1}\eta^{D_m}\sum_{r=1}^{m} A_r(\xi,\eta) \geq 1. \tag{11}$$

If $G(m,k)$ and $H(m,k)$ satisfy (10) for all $m,k \geq 1$, then

$$H(m,k) \leq \xi^{m-1}\eta^{k-1}\sum_{r=1}^{m} A_r, \quad G(m,k) \leq (\eta-\beta)\xi^m\eta^{k-1}\sum_{r=1}^{m} A_r.$$

We are now going to choose the A_r so that condition (11) is satisfied.

LEMMA 3.5. *Let $\xi \geq 1$ and $\eta > \beta > 0$ be constants that satisfy $(\xi-\alpha)(\eta-\beta) = 1$. For each $m \geq 1$, define*

$$A_m = \xi^{1-m}\eta^{-D_m}\max\left(0, 1 - \textstyle\sum_{r=1}^{m-1} A_r\xi^{m-1}\eta^{D_m}\right).$$

Then

$$\xi^{m-1}\eta^{k-1}\sum_{r=1}^{m} A_r = \max_{1\leq j\leq m} \xi^{m-j}\eta^{k-1-D_j}. \tag{12}$$

Therefore, if $G(m,k)$ and $H(m,k)$ satisfy (10) for all $m,k \geq 1$, then

$$H(m,k) \leq \max_{1\leq j\leq m} \xi^{m-j}\eta^{k-1-D_j}, \quad G(m,k) \leq (\eta-\beta)\max_{1\leq j\leq m} \xi^{m-j+1}\eta^{k-1-D_j}.$$

Proof. We will use induction on m. The base case $m = 1$ reads

$$\eta^{k-1}A_1 = \eta^{k-1}\eta^{-D_m} = \max_{1\leq j\leq m} \eta^{k-1-D_j}.$$

Assume inductively that (12) holds for m. Then for $m+1$, we have

$$\begin{aligned}
\xi^m\eta^{k-1}\sum_{r=1}^{m+1} A_r &= \xi \max_{1\leq j\leq m} \xi^{m-j}\eta^{k-1-D_j} + \xi^m\eta^{k-1}A_{m+1} \\
&= \max_{1\leq j\leq m} \xi^{m+1-j}\eta^{k-1-D_j} + \max\left(0, \eta^{k-1-D_{m+1}} - \textstyle\sum_{r=1}^{m} A_r\xi^m\eta^{k-1}\right) \\
&= \max_{1\leq j\leq m} \xi^{m+1-j}\eta^{k-1-D_j} + \max\left(0, \eta^{k-D_{m+1}} - \xi \max_{1\leq j\leq m} \xi^{m-j}\eta^{k-1-D_j}\right).
\end{aligned}$$

Thus,

$$\xi^m\eta^{k-1}\sum_{r=1}^{m+1} A_r = \begin{cases} \eta^{k-1-D_{m+1}} & \text{if } \eta^{k-1-D_{m+1}} \geq \max\limits_{1\leq j\leq m} \xi^{m+1-j}\eta^{k-1-D_j}, \\ \max\limits_{1\leq j\leq m} \xi^{m+1-j}\eta^{k-1-D_j} & \text{otherwise.} \end{cases}$$

It follows that

$$\xi^m\eta^{k-1}\sum_{r=1}^{m+1} A_r = \max_{1\leq j\leq m+1} \xi^{m+1-j}\eta^{k-1-D_j},$$

which completes the induction. The corresponding bounds on $H(m,k)$ and $G(m,k)$ now follow from Lemma 3.4. ∎

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

3.4. Theoretical speedup. We are now ready to estimate the theoretical speedup of WRAP when $P < \infty$ time-parallel tasks can be executed simultaneously. By construction, one cannot start iterating on the time interval I_m until the iteration on I_{m-P} has converged. Define E_m to be the *ending time* for the mth time interval, i.e., the smallest k such that $H(m, k+1) \leq \epsilon$, where ϵ is some predefined tolerance. Then, by definition, we have $E_m = k$, where

$$H(m, k+1) \leq \epsilon \leq H(m, k) \leq \max_{1 \leq j \leq m} \xi^{m-j} \eta^{k-1-D_j}.$$

Suppose the maximum on the right-hand side of the above equation is achieved for $j = j^*$. Then taking logarithms yields

$$(m - j^*) \log \xi - (E_m - D_{j^*} - 1) |\log \eta| \geq -|\log \epsilon|$$

or

$$E_m \leq 1 + D_{j^*} + \frac{|\log \epsilon|}{|\log \eta|} + (m - j^*) \frac{\log \xi}{|\log \eta|}. \tag{13}$$

Moreover, since j^* maximizes $\xi^{m-j} \eta^{k-1-D_j}$, we see that

$$(m - j^*) \log \xi - (k - 1 - D_{j^*}) |\log \eta| \geq (m - j) \log \xi - (k - 1 - D_j) |\log \eta|$$

for all $1 \leq j \leq m$. In other words, we have

$$D_{j^*} - j^* \frac{\log \xi}{|\log \eta|} \geq D_j - j \frac{\log \xi}{|\log \eta|}, \quad j = 1, \ldots, m.$$

This function will be important later, so let us define

$$F_m := D_m - m(\log \xi / |\log \eta|). \tag{14}$$

We can then rewrite (13) as

$$E_m - D_m \leq 1 + \frac{|\log \epsilon|}{|\log \eta|} + \max_{1 \leq j \leq m} F_j - F_m. \tag{15}$$

Note that the left-hand side is the number of iterations required for convergence in the mth time window.

The term $(1 + \frac{\log \epsilon}{\log \eta})$, on the right-hand side of (15), is comparable to the iteration count for a classical Schwarz (non-WR) method on the corresponding elliptic problem, which is bounded by $(1 + \frac{\log \epsilon}{\log \beta})$. The remaining terms measure the additional iterations required because of the adaptive WR. If $\max_{1 \leq j \leq m} F_j - F_m$ were bounded by a constant, then we will have proven that the iteration count is independent of the time horizon. This is a difficult task, in general, because the error estimate in our computational model is only an upper bound; however, we will be able to bound E_m as a constant times m.

Bounding E_m when $m \leq P$ is trivial. Recall that $D_m = 0$ for $m = 1, \ldots, P$, because the first P time intervals have priority over later time intervals. (14) simplifies to

$$F_m = -m \frac{\log \xi}{|\log \eta|}, \quad m = 1, \ldots, P. \tag{16}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Hence, (15) for $m = 1, 2, \ldots, P$ gives

$$E_m \le 1 + \frac{|\log \epsilon|}{|\log \eta|} + \max_{1 \le j \le m} F_j - F_m = 1 + \frac{|\log \epsilon|}{|\log \eta|},$$

which is close to the iteration count for a nonadaptive SWR method on these time blocks when $\eta \approx \beta$. If $m > P$, D_m is no longer zero, and we need to resort to the following recurrence relation to derive an equation for the delay,

$$D_{m+P} = E_m - P, \quad m = 1, 2, \ldots. \tag{17}$$

From (14), we have

$$\begin{aligned} F_{m+P} &= D_{m+P} - (m+P)\frac{\log \xi}{|\log \eta|} \\ &= (E_m - P) - (m+P)\frac{\log \xi}{|\log \eta|} \\ &\le 1 + \frac{|\log \epsilon|}{|\log \eta|} + \max_{1 \le j \le m} F_j - D_m - P\frac{\log \xi}{|\log \eta|} - P \end{aligned}$$

or, equivalently,

$$F_m \le 1 + \frac{|\log \epsilon|}{|\log \eta|} + \max_{1 \le j \le (m-P)} F_j - D_{m-P} - P\frac{\log \xi}{|\log \eta|} - P. \tag{18}$$

Since $D_m = 0$ for $m = 1, \ldots, P$, it will be convenient to simplify $\max_{1 \le j \le m} F_m$ iteratively for $\ell P < m \le (\ell+1)P$. Consider the case $\ell = 1$, i.e., $P < m \le 2P$. Using (16), (18) simplifies to

$$F_m \le 1 + \frac{|\log \epsilon|}{|\log \eta|} - (P+1)\frac{\log \xi}{|\log \eta|} - P.$$

If

$$\Delta := 1 + \frac{|\log \epsilon|}{|\log \eta|} - P\left(1 + \frac{\log \xi}{|\log \eta|}\right)$$

is positive, then

$$\max_{1 \le m \le 2P} F_m \le -\frac{\log \xi}{|\log \eta|} + \Delta,$$

otherwise it is just bounded by $-\log \xi / |\log \eta|$. Repeating this argument for $\ell = 2, 3, \ldots$, we see that for $\ell P < m \le (\ell+1)P$,

$$F_m \le \begin{cases} -\dfrac{\log \xi}{|\log \eta|} + \ell\Delta, & \Delta > 0, \\[2ex] -\dfrac{\log \xi}{|\log \eta|} + \Delta, & \Delta \le 0. \end{cases}$$

By substituting the above into (18), we obtain the following theorem.

THEOREM 3.6. *Consider a WRAP method that satisfies the model* (10)*, and assume that ξ and η satisfy $(\xi - \alpha)(\eta - \beta) = 1$. Let E_m be the time to convergence for the mth time window, i.e., the smallest k such that $H(m, k+1) \le \epsilon$, where ϵ is a predefined tolerance. Let ℓ be an integer such that $\ell P < m \le (\ell+1)P$. Then*

$$E_m \le 1 + \frac{|\log \epsilon|}{|\log \eta|} + (m-1)\frac{\log \xi}{|\log \eta|} + \ell \cdot \max\left\{0, 1 + \frac{|\log \epsilon|}{|\log \eta|} - P\left(1 + \frac{\log \xi}{|\log \eta|}\right)\right\}.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

To estimate the wall time needed to complete the integration, we introduce the concept of *effective parallel linear solves* (EPLS), which is defined as the number of columns in the dependency graph, assuming that all tasks in a column are simultaneously computed. For the standard time stepping algorithm, the number of EPLS is estimated by

$$M\left(1+\frac{|\log\epsilon|}{|\log\beta|}\right) =: k_{\text{std}},$$

where $\beta < \eta$ is the actual contraction rate when we have exact initial conditions, and $(1+\frac{|\log\epsilon|}{|\log\beta|})$ is the number of iterations required for convergence on a single time interval. For the WRAP algorithm, the EPLS is given by $E_M + (M-1)$, where the extra $M-1$ solves arise because the task involving time interval I_M can only appear in the task list after $M-1$ updates, even if there is no delay in execution. Letting $M-1 = \ell P + r$, where $0 \le r < P$, we have

$$\text{EPLS} \le \begin{cases} (\ell+1)\left(1+\frac{|\log\epsilon|}{|\log\eta|}\right) + r\left(1+\frac{\log\xi}{|\log\eta|}\right), & P < \left(1+\frac{|\log\epsilon|}{|\log\eta|}\right) / \left(1+\frac{\log\xi}{|\log\eta|}\right), \\ 1+\frac{|\log\epsilon|}{|\log\eta|} + (M-1)\left(1+\frac{\log\xi}{|\log\eta|}\right), & \text{otherwise.} \end{cases}$$

We see that the ratio

$$P^* = \left(1+\frac{|\log\epsilon|}{|\log\eta|}\right) \Big/ \left(1+\frac{\log\xi}{|\log\eta|}\right)$$

determines the optimal number of processors per subdomain. In fact, if $P < P^*$, then we have

$$\text{EPLS} \le \left(1+\frac{|\log\epsilon|}{|\log\eta|}\right)(\ell+1+r/P^*) \le \frac{M+P-1}{P}\left(1+\frac{|\log\epsilon|}{|\log\eta|}\right).$$

If we have $\eta \approx \beta$, then the theoretical speedup becomes

$$\text{Speedup} = \frac{k_{\text{std}}}{\text{EPLS}} \gtrapprox P\left(1+\frac{P-1}{M}\right)^{-1},$$

meaning the speedup approaches P as the number of time intervals becomes large. Thus, we get perfect speedup in the limit. On the other hand, if $P \ge P^*$, then

$$\text{EPLS} \gtrapprox (M+P^*-1)\left(1+\frac{\log\xi}{|\log\eta|}\right),$$

so the speedup is bounded above by

$$\text{Speedup} \le \frac{MP^*}{M+P^*-1} \to P^* \quad \text{as } M \to \infty.$$

Remark. If we assume (15) is a reasonable approximation of the actual iteration count, i.e., if

$$k_m \approx 1 + \frac{|\log\epsilon|}{|\log\eta|} + \max_{1\le j\le m} F_j - F_m,$$

then a straightforward substitution yields

$$k_m \approx \begin{cases} \frac{|\log\epsilon|}{|\log\eta|} + (m-1)\frac{\log\xi}{|\log\eta|}, & 1 \le m \le P, \\ \max\left(P(1+\frac{\log\xi}{|\log\eta|}), \frac{|\log\epsilon|}{|\log\eta|}\right), & m > P. \end{cases}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Thus, for the initial time intervals, we need to take additional iterations to offset the growth of the error as m increases. The same thing happens with the nonadaptive WR method. Beyond the first P intervals, however, the number of iterations is essentially constant, but the constant depends on the number of processors P. For small P, we take the same number of iterations as the sequential method, but for large P, the constant is proportional to P. This is in agreement with our numerical experiments; see section 4.

Remark. The maximum possible speedup when $P = M$ has been previously studied for the nonadaptive WR method [30]. Specifically, EPLS $= M + K$, where K is the number of waveform iterations computed for the nonadaptive WR method. To compute the maximum possible speedup when $P = M$ for the adaptive WR method, we first let k_m be the number of iterations required by a Schwarz iteration in time block I_m (i.e., the adaptive WR method with $P = 1$). Denote $k_{\text{tot}} = \sum_{m=1}^{M} k_m$. Let $\tilde{k}_m$ be the number of iterations required in time block I_m for the adaptive WR method with $P = M$, and denote $\tilde{k}_{\max} = \max_{1 \le m \le M} \tilde{k}_m$. Then the maximum possible speedup for the adaptive WR method with $P = M$ is

$$\frac{k_{\text{tot}}}{M + \tilde{k}_{\max}}. \tag{19}$$

This speedup can be estimated by realizing that $k_{\text{tot}} = M k_{\text{avg}}$, and the ratio $\frac{M}{M+\tilde{k}_{\max}}$ is bounded above by one. Hence, the maximum possible speedup is bounded by k_{avg}.

4. Numerical experiments. In this section, we perform several experiments that illustrate the behavior of the WRAP framework applied to different DD methods and problems. In subsection 4.1, we solve the heat equation using three DD methods, namely, the classical and optimized SWR methods, as well as the NNWR method. In subsection 4.2, we briefly survey other parallel-in-time approaches to highlight the difficulty of time parallelism and to frame our contributions in the broader picture. In subsection 4.3, we consider an advection-diffusion equation that is advection dominated; this is an interesting case because the performance of other time-parallel methods such as parareal [26], deteriorates as the equation becomes more and more dominated by advection. Finally in subsection 4.4, we present a nonlinear PDE system that models an idealized autocatalytic reaction.

4.1. Linear heat equation. We begin by using the adaptive classical SWR approach to solve the linear heat equation in one dimension,

$$u_t = u_{xx}, \quad x \in [0,1], \quad t \in [0,1],$$
$$u(0,x) = \sin(\pi x).$$

We discretize the system using backward Euler in time and central differences in space, with $\Delta x = 1/1024$ and $\Delta t = 0.01$. The spatial domain is subdivided into four overlapping subdomains; the width of the overlap region is chosen to be $\frac{1}{16}$th of the subdomain width, requiring the classical SWR method to take *many* iterations to converge to the monodomain solution. One hundred time blocks, each consisting of one time step, are used. For a tolerance of 10^{-6}, the number of waveform iterates required at each time step for various *ntasks* values are shown in Figure 5.

From Figure 5, several observations should be made. First, consider the total number of iterations (tasks) required for each implementation with *ntasks*, i.e., the area under each curve in Figure 5. The implementation requiring the fewest to-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

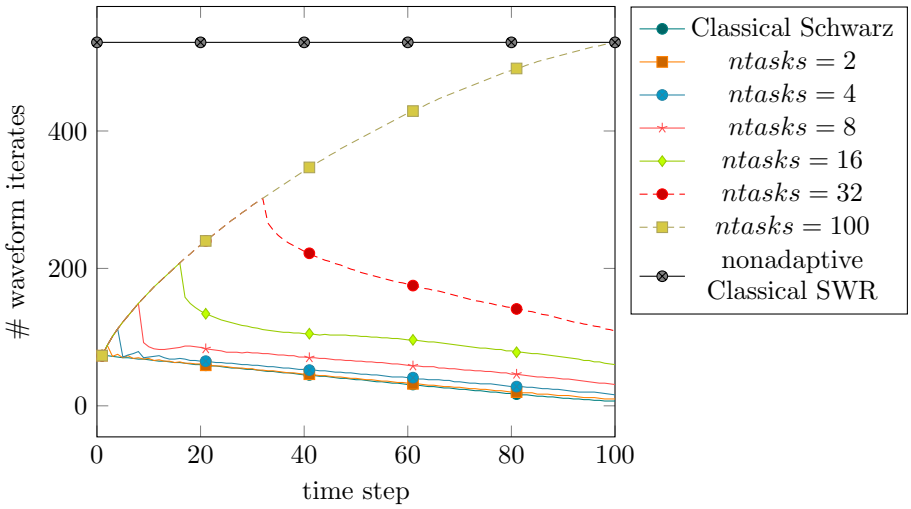

FIG. 5. *Classical Schwarz coupling conditions: Number of waveform iterates at each time step required to reach the same final tolerance for varying numbers of simultaneous tasks.*

tal number of iterations is $ntasks = 1$, corresponding to the classical Schwarz DD method. This is unsurprising since we are iterating each time step until convergence, so later time steps do not need to spend extra iterations to eliminate the error propagated from earlier time steps. Second, the total number of waveform iterates for the adaptive WR approach is significantly lower than for the nonadaptive classical SWR approach. Last, as $ntasks$ is increased, the total number of waveform iterates required increases.

Figure 5 does not address the speedup that is possible using adaptive pipelining, however. In Figure 6, we depict the computation of the waveform iterates for each time step (x-axis) relative to when they are computed in the simulation (y-axis) for the case $ntasks = 8$. (Figure 6 can be viewed as the silhouette of the dependency graph, rotated by 90 degrees.) Observe that the height of the bar corresponds to the number of iterations required at each time step. The WRAP algorithm does more iterations initially, consistent with the analysis. Also observe that each horizontal slice of the plot in Figure 6 will have at most eight markers because the maximum number of tasks that are simultaneously computed in this example is $ntasks = 8$. Finally, we see that the WRAP algorithm has a preference for iterating earlier time steps to convergence; later time steps are not started until the earlier time steps are iterated to convergence.

Table 1 shows the speedup that can be expected using the adaptive pipeline WR approach with classical Schwarz transmission conditions. Columns 2–3 display the EPLS and speedup for $M = 100$ time intervals; columns 4–5 show the same for a repeated experiment with $M = 1000$. The theoretical speedup is computed by taking the ratio of the the number of EPLS using the adaptive pipeline WR framework against that of the classical Schwarz DD method ($ntasks = 1$). For $M = 100$, the speedup increases monotonically with $ntasks$, but saturates at approximately 6, even when $ntasks = 100$. The observed saturated speedup is in agreement with (19). Specifically, we have $\tilde{k}_{\max} = 529$ for the adaptive WR method with $ntasks = 100$. Since $M = 100$ and $k_{\text{tot}} = 3841$ (note: $k_{\text{tot}} =$ EPLS for $ntasks = 1$), (19) gives a theoretical maximum speedup of 6.1.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

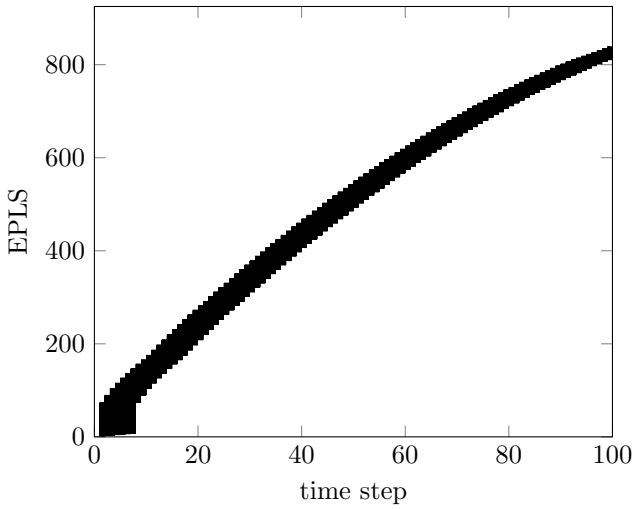

FIG. 6. *Classical Schwarz coupling conditions: bars denote computation of the waveform iterates for each time step (x-axis) relative to when they are computed in the simulation (y-axis) for the case* $ntasks = 8$. *Here, the wall time unit is the amount of time it would take to compute one parallel solve. This WRAP method using* $ntasks = 8$ *requires* 841 *EPLS.*

TABLE 1

Heat equation in one dimension using classical Schwarz coupling conditions. Reported: theoretical speedup using the adaptive pipeline WR approaches for various ntasks (number of time-parallel tasks), with $M = 100$ *time blocks (columns* 2–3*) and* $M = 1000$ *(columns* 4–5*). The EPLS is defined in subsection* 3.4.

ntasks	$M = 100$		$M = 1000$	
	EPLS	Speedup	EPLS	Speedup
1	3841	–	9902	–
2	2017	1.90	5182	1.91
4	1195	3.21	3101	3.19
8	841	4.57	2183	4.54
16	687	5.59	1748	5.66
32	629	6.11	1537	6.44
100	628	6.12	1388	7.13

Speedup can be potentially improved when more time blocks are used, since the processors can then march in a pipe for a larger number of tasks, as shown in columns 4–5 of Table 1. For $M = 1000$, the speedup saturates at around 7, which is better than before, but only marginally. The reason is that the problem has become easier as Δt becomes smaller: for $ntasks = 1$, i.e., the standard time stepping method only requires an average of 9.9 EPLS per time step, instead of 38.4 EPLS per time step when $\Delta t = 0.01$. Also note that WRAP now only takes 1388 effective solves (with $ntasks = 100$) to complete a 1000-step integration, i.e., about 1.4 EPLS per step. With such a low EPLS per step, it is unlikely that further speedup can be obtained by adding processors in the time direction. Nevertheless, this speedup comes *on top of any spatial parallelism*, so an extra multiplicative factor of 5 to 7 in the speedup is nontrivial.

Next, we report the results when different coupling conditions are used. In columns 2–3 of Table 2, we report the EPLS and speedup when optimized transition conditions are imposed between subdomains, with optimized parameter $p = \frac{1}{\sqrt{\Delta t}}$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 2
Heat equation in one dimension using the WRAP method for Optimized SWR and Neumann–Neumann WR (NNWR). Here, four non-overlapping spatial domains were used along with $M = 100$ time blocks, and a tolerance of 10^{-12}.

ntasks	Optimized SWR		NNWR	
	EPLS	Speedup	EPLS	Speedup
1	865	–	1358	–
2	436	1.98	681	1.99
4	228	3.79	350	3.88
8	176	4.91	206	6.59
16	165	5.24	170	7.99
100	165	5.24	164	8.28

TABLE 3
Heat equation in two dimensions using classical Schwarz coupling conditions.

ntasks	EPLS	Speedup
1	19042	–
2	9572	1.99
4	4948	3.85
8	2800	6.80
16	1838	10.36
32	1400	13.60
64	1205	15.80

The time horizon is divided into $M = 100$ time blocks, with each time block consisting of a single time step. Four nonoverlapping spatial domains were used, with $\Delta x = 1/1024$. Even with a smaller tolerance of 10^{-12}, modest parallel speedup numbers are observed. This can be explained by the low number of EPLS per time step, which went from 8.65 for $ntasks = 1$ to 1.65 for $ntasks \geq 16$, since more effective coupling conditions were used. In columns 4–5, we report the EPLS and speedup for a non-Schwarz variant: an adaptive pipeline parallel implementation for NNWR methods [31]. The NNWR method performs a two-step iteration consisting of first solving a "Dirichlet" subproblem on each space–time domain, followed by solving an auxiliary "Neumann" subproblem.

Last, we solve the linear heat equation in two dimensions to illustrate the speedup that can be expected when the EPLS per step increases,

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

Using $\Delta x = \frac{1}{40}, \Delta y = \frac{1}{60}$, we split the spatial domain into 4×3 subdomains with an overlap of $2\Delta x$ or $2\Delta y$. For the time integration, we take $M = 400$ time blocks with $\Delta t = \frac{1}{400}$. The EPLS and speedup are reported in Table 3. The average EPLS per time step is 47 for $ntasks = 1$; the average EPLS per time step for $ntasks = 16$ is five.

4.2. Other parallel-in-time approaches. Before we continue with more numerical experiments, we digress briefly to compare our numbers against published speedup results obtained for other time-parallel methods such as revisionist integral deferred correction (RIDC) methods [8], parareal [26], multigrid-in-time (MGRIT) [11], and parallel exponential integrators [12]. This comparison is not intended to promote any specific method or approach, as many of the underlying problems, how speedup is evaluated, and the underlying computing hardware may differ. Rather, the

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

purpose of this discussion is to highlight the difficulty of time parallelism and situate our contribution in the broader picture.[4]

- We begin our discussion with RIDC, which uses a predictor and correctors in parallel to generate high-order time integrators. Although only capable of small-scale parallelism, RIDC is able to achieve 8× speedup using eight time-parallel tasks to solve the linear heat equation and the Brusselator [7].
- The most widely studied parallel-in-time algorithm is the parareal algorithm. Selecting the coarse and fine propagators is an art but, often, the coarse grid correction is the bottleneck of the parareal algorithm. A recent attempt to accelerate coarse grid correction has shown that a speedup of 5–6× can be expected with a parallel coarse grid correction [37] when there are 100 time-parallel tasks.
- We next turn our attention to XBraid [1], a software package that implements MGRIT. In [19], XBraid was used to solve a model problem that mimics unsteady flow at low Reynolds number. Using 256 cores, XBraid was able to achieve a speed up of 5–6× over a serial computation.
- The original parallel exponential integrator was generalized for nonlinear problems by using a rational approximation. Dubbed REXI (rational exponential integrators), the idea is to approximate the computationally expensive approach of exponential integrators while adding additional degrees of parallelization. A recent manuscript [32] explores scalability for REXI applied to linear oscillatory problems. For their time parallelization results, the authors get a performance improvement of 118× using 3584 cores as compared against a sequential Runge–Kutta 4 integrator, approximately 3% efficiency.

4.3. Advection-diffusion. Next, we solve the advection-diffusion equation

$$u_t = \nu u_{xx} + u_x, \quad x \in [0,2], \quad t \in [0,4],$$

with periodic boundary conditions

$$u(0,t) = u(2,t), \quad u_x(0,t) = u_x(2,t), \qquad t \in (0,T),$$

and with initial conditions $u(x,0) = e^{-20(x-1)^2}$ for $x \in (0,2)$. We discretize the system using backward Euler in time and first-order upwind in space, with $\Delta x = 1/512$ and $\Delta t = 0.01$. As $\nu \to 0$, the problem becomes more and more advection dominated. It has been shown in [13] that the convergence of the parareal method deteriorates for small ν, and speedup suffers as a result. We show our results for $\nu = 0.05$ and $\nu = 0.005$ in Table 4. Four overlapping subdomains and Dirichlet transmission conditions are used in both cases. We see that our speedup remains reasonable even for these highly-advection-dominated cases. In fact, the less favorable speedup for $\nu = 0.005$ is due to the problem being *easier*: serial time stepping only requires 2400 EPLS, or 6 EPLS per time step, instead of 4589 EPLS (or 11.5 EPLS per time step) in the more diffusive case. For $\nu = 0.005$, Figure 7 shows only a few waveform iterates are required to reach the same final tolerance if the fewer *ntasks* are used.

4.4. Brusselator. In the last experiment, we consider an idealized autocatalytic reaction, the Brusselator system, which can be modeled by the following reaction-

[4] A notable omission from this brief survey is the parallel full approximation scheme in space and time PFASST [10] because the spatial and time parallelism is tightly coupled there.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

TABLE 4
Theoretical speedup using the WRAP framework applied to the advection-diffusion problem with various ν's and $M = 400$ time blocks.

ntasks	$\nu = 0.05$		$\nu = 0.005$	
	EPLS	Speedup	EPLS	Speedup
1	4859	–	2400	–
2	2437	1.99	1201	2.00
4	1250	3.89	604	3.97
8	754	6.44	422	5.69
16	562	8.65	418	5.74
32	489	9.94	418	5.74
100	487	10.00	418	5.74

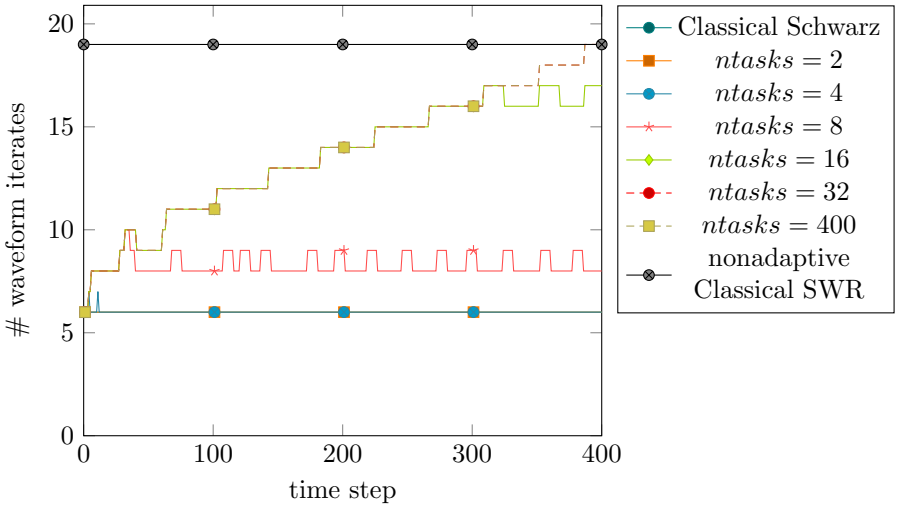

FIG. 7. *WRAP for advection-diffusion equation with $\nu = 0.005$: plot shows the number of waveform iterates at each time step required to reach the same final tolerance for varying numbers of simultaneous tasks.*

diffusion system,

$$u_t = A + u^2\, v - (B+1)u + \alpha\, u_{xx},$$
$$v_t = B\, u - u^2\, v + \alpha\, v_{xx}.$$

Here, $A = 1$ and $B = 3$ are rate constants, and $\alpha = \frac{1}{50}$ is the diffusion constant. The initial and boundary conditions are

$$u(0,t) = u(1,t) = 1, \qquad v(0,t) = v(1,t) = 3,$$
$$u(x,0) = 1 + \sin 2\pi\, x, \qquad v(x,0) = 0.$$

This reaction system is nonlinear, and stiff due to the diffusion. We discretize the system using an implicit-explicit scheme: the reaction term is handled explicitly using the explicit Euler integrator, and the diffusion term is handled implicitly using the implicit Euler integrator. A centered finite difference approximation is used to approximate the diffusion term. The spatial domain is subdivided into four overlapping subdomains; the width of the overlap region is again chosen to be $\frac{1}{16}$th of the subdomain width. One hundred time blocks, each consisting of one time step, are used.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

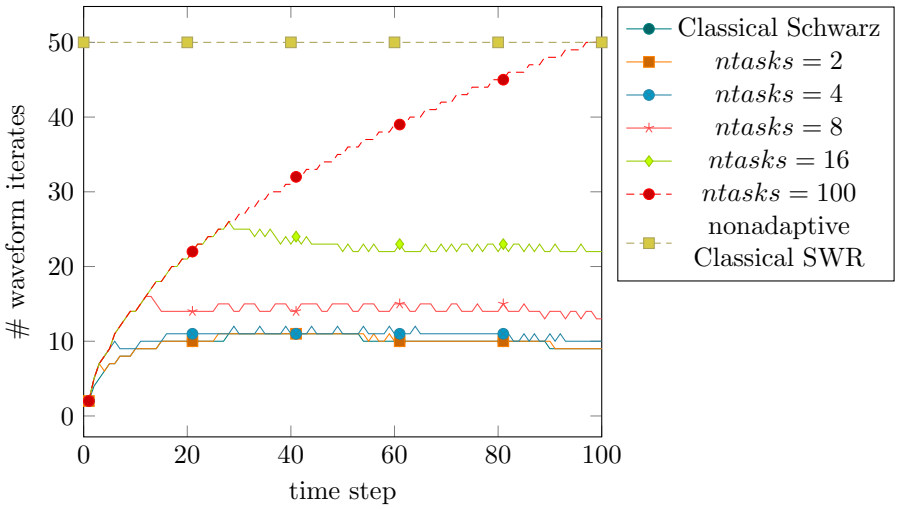

FIG. 8. *Solving the Brusselator equation using the WRAP framework with Dirichlet transmission conditions. Here, we plot the number of waveform iterates at each time step required to reach the same final tolerance for varying numbers of simultaneous tasks.*

TABLE 5
Theoretical speedup for solving the Brusselator system using the WRAP framework with Dirichlet transmission condition and $M = 100$ *time blocks.*

ntasks	EPLS	Speedup
1	975	–
2	495	1.97
4	270	3.61
8	184	5.30
16	154	6.33
100	149	6.54

Similar observations to the first numerical experiment can be made. For a tolerance of 10^{-6}, the number of waveform iterates required at each time step for various *ntasks* values is shown in Figure 8. The theoretical speedup is summarized in Table 5.

5. Conclusions. Adaptive pipelining is introduced to efficiently utilize a fixed number of computational workers for WR methods. In this method, we address two main issues of WR methods, namely, convergence degradation for long-time integration, and oversolving in the initial time steps. We do so by keeping the effective window of integration small, and reassigning workers from converged time steps in order to grow the time horizon. The new WRAP methods are analyzed to show the theoretical speedup that can be expected. The WRAP framework has several desirable properties. First, one limiting case recovers Schwarz DD methods, allowing a direct comparison with classical DD methods. Another limiting case recovers classical WR methods. The numerical experiments show that parallel speedup with moderate efficiency over classical DD methods can be expected with the WRAP framework. Second, although the parallel speedup saturates as the number of tasks (i.e., number of waveform iterates computed in parallel) increases, the speedup appears as a multiplicative factor when used in combination with other temporal or spatial parallelism. In fact, this method can be used within parareal itself in order to accelerate the fine

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

integration steps. Thus, our method is complementary to existing space and time parallel methods.

Acknowledgment. We thank the anonymous referees for their valuable comments and suggestions, which helped us improve the quality of the paper.

REFERENCES

[1] *XBraid: Parallel Time Integration with Multigrid*, http://llnl.gov/casc/xbraid.
[2] M. Al-Khaleel, M. J. Gander, and A. E. Ruehli, *A mathematical analysis of optimized waveform relaxation for a small RC circuit*, Appl. Numer. Math., 75 (2014), pp. 61–76, https://doi.org/10.1016/j.apnum.2012.12.005.
[3] A. Bellen and M. Zennaro, *The use of Runge-Kutta formulae in waveform relaxation methods*, Appl. Numer. Math., 11 (1993), pp. 95–114, https://doi.org/10.1016/0168-9274(93)90042-P.
[4] D. Bennequin, M. J. Gander, and L. Halpern, *A homographic best approximation problem with application to optimized Schwarz waveform relaxation*, Math. Comp., 78 (2009), pp. 185–223.
[5] S. Brenner and R. Scott, *The Mathematical Theory of Finite Element Methods*, Texts Appl. Math. 15, Springer, New York, 2008.
[6] K. Burrage, C. Dyke, and B. Pohl, *On the performance of parallel waveform relaxations for differential systems*, Appl. Numer. Math., 20 (1996), pp. 39–55.
[7] A. J. Christlieb, R. D. Haynes, and B. W. Ong, *A parallel space-time algorithm*, SIAM J. Sci. Comput., 34 (2012), pp. C233–C248, https://doi.org/10.1137/110843484.
[8] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput., 32 (2010), pp. 818–835, https://doi.org/10.1137/09075740X.
[9] V. Dolean, P. Jolivet, and F. Nataf, *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*, Other Titles Appl. Math. 144, SIAM, Philadelphia, 2015.
[10] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132, https://doi.org/10.2140/camcos.2012.7.105.
[11] R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM J. Sci. Comput., 36 (2014), pp. C635–C661.
[12] M. J. Gander and S. Güttel, *PARAEXP: A parallel integrator for linear initial-value problems*, SIAM J. Sci. Comput., 35 (2013), pp. C123–C142, https://doi.org/10.1137/110856137.
[13] M. J. Gander, *Five Decades of Time Parallel Time Integration, and a Note on the Degradation of the Performance of the Parareal Algorithm as a Function of the Reynolds Number*, Oberwolfach Report, 2017.
[14] M. J. Gander, F. Kwok, and B. C. Mandal, *Dirichlet–Neumann and Neumann-Neumann waveform relaxation algorithms for parabolic problems*, Electron. Trans. Numer. Anal., 45 (2016), pp. 424–456.
[15] 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.
[16] M. J. Gander and H. Zhao, *Overlapping Schwarz waveform relaxation for the heat equation in n dimensions*, BIT, 42 (2002), pp. 779–795.
[17] C. Gear, *Waveform methods for space and time parallelism*, J. Comput. Appl. Math., 38 (1991), pp. 137–147.
[18] E. Giladi and H. B. Keller, *Space-time domain decomposition for parabolic problems*, Numer. Math., 93 (2002), pp. 279–313.
[19] S. Günther, N. R. Gauger, and J. B. Schroder, *A non-intrusive parallel-in-time adjoint solver with the XBraid library*, Comput. Vis. Sci., 19 (2018), pp. 85–95.
[20] L. Halpern, C. Japhet, and J. Szeftel, *Optimized Schwarz waveform relaxation and discontinuous Galerkin time stepping for heterogeneous problems*, SIAM J. Numer. Anal., 50 (2012), pp. 2588–2611.
[21] R. Jeltsch and B. Pohl, *Waveform relaxation with overlapping splittings*, SIAM J. Sci. Comput., 16 (1995), pp. 40–49, https://doi.org/10.1137/0916004.
[22] F. Kwok, *Neumann-Neumann waveform relaxation for the time-dependent heat equation*, in Domain Decomposition in Science and Engineering XXI, J. Erhel, M. J. Gander,

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

L. Halpern, G. Pichot, T. Sassi, and O. B. Widlund, eds., Lect. Notes Comput. Sci. Eng. 98, Springer, Cham, Switzerland, 2014, pp. 189–198.

[23] B. Leimkuhler, *Estimating waveform relaxation convergence*, SIAM J. Sci. Comput., 14 (1993), pp. 872–889, https://doi.org/10.1137/0914054.

[24] B. Leimkuhler and A. Ruehli, *Rapid convergence of waveform relaxation*, Appl. Numer. Math., 11 (1993), pp. 211–224, https://doi.org/10.1016/0168-9274(93)90049-W.

[25] E. Lelarasmee, A. Ruehli, and A. Sangiovanni-Vincentelli, *The waveform relaxation method for time-domain analysis of large scale integrated circuits*, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., 1 (1982), pp. 131–145.

[26] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps (pararéel)*, C. R. Acad. Sci. Ser. I Math., 332 (2001), pp. 661–668, https://doi.org/10.1016/S0764-4442(00)01793-6.

[27] B. C. Mandal, *A time-dependent Dirichlet-Neumann method for the heat equation*, in Domain Decomposition in Science and Engineering XXI, J. Erhel, M. J. Gander, L. Halpern, G. Pichot, T. Sassi, and O. B. Widlund, eds., Lect. Notes Comput. Sci. Eng. 98, Springer, Cham, Switzerland, 2014, pp. 467–475.

[28] U. Miekkala and O. Nevanlinna, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 459–482.

[29] O. Nevanlinna, *Remarks on Picard-Lindelöf iteration*, BIT, 29 (1989), pp. 535–562.

[30] B. W. Ong, S. High, and F. Kwok, *Pipeline Schwarz waveform relaxation*, in Domain Decomposition Methods in Science and Engineering XXII, T. Dickopf, M. J. Gander, L. Halpern, R. Krause, and L. Pavarino, eds., Lect. Notes Comput. Sci. Eng. 104, Springer, Cham, Switzerland, 2016, pp. 179–187, https://doi.org/10.1007/978-3-319-18827-0_36.

[31] B. W. Ong and B. C. Mandal, *Pipeline implementations of Neumann–Neumann and Dirichlet–Neumann waveform relaxation methods*, Numer. Algorithms, 78 (2017), pp. 1–20, https://doi.org/10.1007/s11075-017-0364-3.

[32] M. Schreiber, P. S. Peixoto, T. Haut, and B. Wingate, *Beyond spatial scalability limitations with a massively parallel method for linear oscillatory problems*, Int. J. High Perform. Comput. Appl., 32 (2018), pp. 913–933, https://doi.org/10.1177/1094342016687625.

[33] R. D. Skeel, *Waveform iteration and the shifted Picard splitting*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 756–776, https://doi.org/10.1137/0910046.

[34] A. Toselli and O. B. Widlund, *Domain Decomposition Methods: Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer, Berlin, 2005.

[35] S. Vandewalle and E. Van de Velde, *Space-time concurrent multigrid waveform relaxation*, Ann. Numer. Math., 1 (1994), pp. 335–346.

[36] J. White and A. Sangiovanni-Vincentelli, *Partitioning algorithms and parallel implementations of waveform relaxation algorithms for circuit simulation*, in IEEE Procedings of the International Symposium on Circuits and Systems (ISCAS), IEEE, New York, 1985, pp. 1069–1072.

[37] S.-L. Wu, *Toward parallel coarse grid correction for the parareal algorithm*, SIAM J. Sci. Comput., 40 (2018), pp. A1446–A1472, https://doi.org/10.1137/17M1141102.

Downloaded 02/14/19 to 141.219.44.85. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.