Pipeline Schwarz Waveform Relaxation

Benjamin Ong[1], Scott High[2], and Felix Kwok[3]

Abstract To leverage the computational capability of modern supercomputers, existing algorithms need to be reformulated in a manner that allows for many concurrent operations. In this paper, we outline a framework that reformulates classical Schwarz waveform relaxation so that successive waveform iterates can be computed in a parallel pipeline fashion after an initial start-up cost. The communication costs for various implementations are discussed, and numerical scaling results are presented.

Key words: Schwarz waveform relaxation, pipeline parallelism, domain decomposition, distributed computing

1 Introduction

Schwarz Waveform Relaxation (SWR) introduced in [2] has been analyzed for a wide range of time-dependent problems, including the parabolic heat equation [7], wave equation and advection-diffusion equations [6, 8], Maxwell's equations [4], and the porous medium equation [9]. In contrast to classical Schwarz iterations, where the time-dependent PDE is discretized in time and domain-decomposition is applied to the sequence of steady-state problems, SWR solves *time-dependent* sub-problems; this relaxes synchronization of the sub-problems and provides a means to couple disparate solvers applied to individual sub-problems, for example [10]. SWR has also been shown in [8, 1] to have superlinear convergence for small time windows. This paper outlines a framework that reformulates SWR so that successive waveform iterates can be computed in a pipeline fashion, allowing for increased concurrency and hence, increased scalability for SWR-type algorithms. In §2, we review the SWR algorithm before introducing and comparing several Pipeline Schwarz Waveform Relaxation algorithms (PSWR) in §3. Numerical scaling results for the linear heat equation are presented in §4.

[1] Michigan State University, Institute for Cyber-Enabled Research, e-mail: ongbw@msu.edu ·
[2] Michigan State University, Department of Mathematics, e-mail: highscot@msu.edu ·
[3] Université de Genève, Switzerland, Section de Mathématiques e-mail: felix.kwok@unige.ch

2 Schwarz Waveform Relaxation

Denote the PDE of interest as

$$\begin{aligned} u_t &= \mathscr{L}(t,u), \quad (x,t) \in \Omega \times [0,T] \\ u(x,0) &= f(x), \quad x \in \Omega \\ u(z,t) &= g(z,t), \quad z \in \partial\Omega. \end{aligned} \tag{1}$$

Consider a partitioning of the domain, $\Omega = \cup_i \Omega_i$. The domains in the partition may be overlapping or non-overlapping. Let u_i denote the solution on sub-domain Ω_i. Then, equation (1) can be decomposed into a coupled system of equations,

$$\begin{aligned} (u_i)_t &= \mathscr{L}(t,u_i), \quad (x,t) \in \Omega_i \times [0,T] \\ u_i(x,0) &= f(x), \quad x \in \Omega_i \\ u_i(z,t) &= g(z,t), \quad z \in \partial\Omega_i \cap \partial\Omega, \\ \mathscr{T}_{ij}(u_i(z,t)) &= \mathscr{T}_{ij}(u_j(z,t)), \quad z \in \partial\Omega_i \cap \partial\Omega_j. \end{aligned} \tag{2}$$

where T are transmission operators appropriate to the equation (1). SWR decouples the system of PDEs in equation (2). Let $u_i^{[k]}$ denote the k-th waveform iterate on sub-domain Ω_i. After specifying an initial estimate for the sub-domain solution on the interfaces, $u_i^{[0]}(z,t), z \in \partial\Omega_i \setminus \partial\Omega$, the SWR algorithm iteratively solves PDEs (3) for waveform iterates $k = 1,2,\ldots$ until convergence,

$$\begin{aligned} (u_i^{[k]})_t &= \mathscr{L}(t,u_i^{[k]}), \quad (x,t) \in \Omega_i \times [0,T] \\ u_i^{[k]}(x,0) &= f(x), \quad x \in \Omega_i \\ u_i^{[k]}(z,t) &= g(z,t), \quad z \in \partial\Omega_i \cap \partial\Omega, \\ \mathscr{T}_{ij}(u_i^{[k]}(z,t)) &= \mathscr{T}_{ij}(u_j^{[k-1]}(z,t)), \quad z \in \partial\Omega_i \cap \partial\Omega_j. \end{aligned} \tag{3}$$

A pseudo-code for the algorithm is presented on the next page. Observe that SWR allows for each sub-domain to independently compute time-dependent solutions on their respective sub-domains (lines 9-11) During each waveform iteration, transmission data on each sub-domain is aggregated for the entire computational time interval before boundary data is exchanged between neighboring sub-domains (lines 12-14).

3 Pipeline Schwarz Waveform Relaxation

Using a similar approach described in [3], the relaxation framework can be rewritten so that after initial start-up costs, multiple waveform iterations can be computed in a pipeline-parallel fashion. A graphical example of the PSWR algorithm for two

Schwarz Waveform Relaxation Algorithm

```
1. MPI Initialization
2. parallel for  $i = 1 \dots N$ (Sub-domain)
3.     for $t = \Delta t \dots T$
4.         Guess $u_i^{[0]}(z,t), \quad z \in \partial\Omega_i \cap \partial\Omega_j$
5.     end
6. end
7. for  $k = 1 \dots K$ (Waveform iteration)
8.     parallel for  $i = 1 \dots N$ (Sub-domain)
9.         for $t = \Delta t \dots T$
10.            Solve for  $u_i^{[k]}(t,x)$
11.        end
12.        for $t = \Delta t \dots T$
13.            Exchange transmission data $\mathscr{T}(u_i^{[k]}(t,z))$
14.        end
15.        Check convergence
16.    end
17. end
```
.

subdomains is shown in Figure 1. To simplify the presentation, we first present the

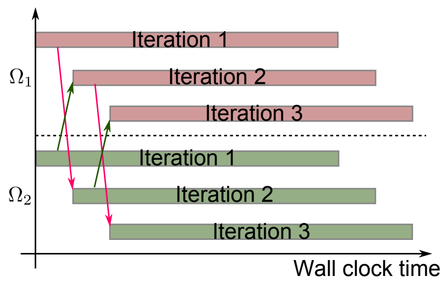

Fig. 1 The proposed PSWR algorithm allows for multiple Schwarz waveform iterations to be simultaneously computed. After an initial start-up cost, multiple iterates are computed in a pipeline fashion.

algorithm for the simplified case where the *same* time discretization is used for all sub-problems (Pipeline Schwarz Waveform Relaxation Algorithm 1).

Several observations should be made about the proposed PSWR algorithm. First, a Schwarz iteration can only proceed if boundary data (i.e. transmission conditions) from the previous iterate are available; this condition (part of the start-up cost before the PSWR algorithm can be run in a pipeline fashion) is checked by the `if` statement in line 12. Secondly, transmission data is exchanged after every time step to facilitate the pipeline parallellism. This added synchronization can be relaxed at the expense of increasing the start-up cost needed to run this algorithm in a pipeline fashion. This pipeline parallelism allows for $N \cdot K$ concurrent processes in the PSWR algorithm

Pipeline Schwarz Waveform Relaxation Algorithm 1

```
1. MPI Initialization
2. parallel for  $i = 1 \dots N$ (Sub-domain)
3.     for $t = \Delta t \dots T$
4.         Guess $u_i^{[0]}(z,t), \quad z \in \partial\Omega_i \cap \partial\Omega_j$
5.     end
6.     Set $t^{[0]} = T$
7. end
8. parallel for  $k = 1 \dots K$ (Waveform iteration)
9.     parallel for  $i = 1 \dots N$ (Sub-domain)
10.        set $t^{[k]} = \Delta t$
11.        While $t^{[k]} \leq T$
12.            If  $t^{[k]} < t^{[k-1]}$
13.                Solve for  $u_i^{[k]}(t^{[k]},x)$
14.                Exchange transmission data $\mathscr{T}(u_i^{[k]}(t^{[k]},z))$
15.                $t^{[k]} \leftarrow t^{[k]} + \Delta t$
16.            end
17.        end
18.        Check convergence
19.    end
20. end
```
.

with efficiency $\frac{N_t}{K+N_t}$ (accounting for start-up costs), where N_t is the number of time steps used to discretize the time domain $[0,T]$, N is the number of sub-domains, K is the number of waveform iterates. This contrasts with the SWR algorithm, which can only utilize N concurrent processes corresponding to the N sub-domains. This increased concurrency in PSWR comes with the overhead of an increased number of messages and synchronization.

For the SWR algorithm, one needs to send $O(K-1)$ message of size $O(N_t)$. If $N \cdot K$ processors are used in a pipeline parallel fashion as described in Pipeline Schwarz Waveform Relaxation Algorithm 1, $O((K-1) \cdot N_t)$ messages of size $O(1)$ are needed. More generally, if $N \cdot p$ processors are used in the PSWR algorithm, where $p < K$ is a multiple of K, then $O((p-1)/p \cdot K \cdot N_t)$ messages of size $O(1)$, and $O(K/p-1)$ messages of size $O(N_t)$, are needed. We note that the PSWR algorithm can also be implemented using a framework the naturally reduces the number of messages in a system. Assuming a heterogenous compute platform (where each socket has multiple cores), one can use the MPI-3 framework [11] or the OpenMP protocol in the outer “parallel for” statement in line 8, to aggregate transmission data from line 14 naturally before exchanging transmission data with neighboring nodes. Alternatively, because nodes working on waveform iterate k only needs to communicate with waveform iterates $k-1$, the PSWR algorithm allows for a natural grouping of nodes so that one can (in principle) use multiple overlapping communicators to leverage data/network-topology and software defined networking advances [5] to add scalability.

Generalizations to allow for disparate time discretizations in each sub-problem are possible. We list the algorithm without implementation. Unlike PSWR Algorithm 1, it is not possible to keep the “pipe” full, i.e. domain i might necessarily need to wait for it’s neighbouring domains to provide adequete boundary data. Additionally, solving for $u_i^{[k]}(t_i^{[k]},x)$ in line 14 requires an interpolation algorithm to correctly obtain the correct transmission condition to be used in the solution of (3). Lastly, an implementation decision has to be made on how to collect and store the data from neighboring domains before the interpolation is used to obtain the transmission conditions for an update in line 14.

Pipeline Schwarz Waveform Relaxation Algorithm 2

```
1. MPI Initialization
2. parallel for  $i = \dots 1..N$ (Sub-domain)
3.     for $t_i = \Delta t_i \dots T$
4.         Guess $u_i^{[0]}(z,t), \quad z \in \partial\Omega_i \cap \partial\Omega_j$
5.     end
6.     Set $t_i^{[0]} = T$
7. end
8. parallel for  $k = 1 \dots K$ (Waveform iteration)
9.     parallel for  $i = 1 \dots N$ (Sub-domain)
10.        initialize $\Delta t_i^{[k]}$
11.        set $t_i^{[k]} = \Delta t_i^{[k]}$
12.        While $t_i^{[k]} \leq T$
13.            If  $t_i^{[k]} < t_j^{[k-1]}$ for all neighbors $j$
14.                Solve for  $u_i^{[k]}(t_i^{[k]},x)$
15.                Send transmission data $\mathscr{T}(u_i^{[k]}(t_i^{[k]},z))$ to neighbor nodes
16.                $t_i^{[k]} \leftarrow t_i^{[k]} + \Delta t_i^{[k]}$
17.            end
18.        end
19.        Check convergence
20.    end
21. end
```
.

4 Numerical Experiments

We present results from strong scaling studies, which vary the number of computational cores used to compute the PSWR algorithm while keeping total discretized problem size constant. The diffusion equation $u_t = k(u_{xx} + u_{yy})$ is solved in $\mathbb{R}^2$ using a centered five point finite-difference approximation in space, and a backward Euler time integrator. In our first scaling study, 400x400 grid points are decomposed into 4x4 non-overlapping domains for 400 total time steps. Optimized robin transmission conditions of the form

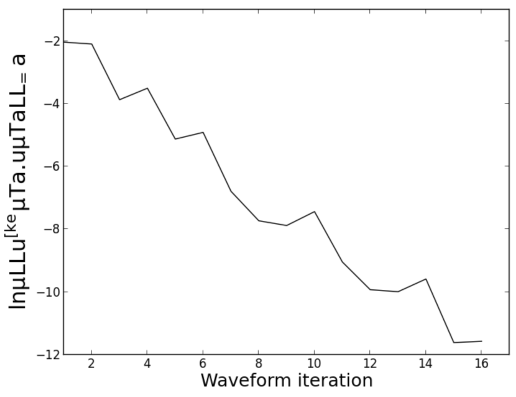

Fig. 2 The error of the waveform iterates at time T is computed relative to monodomain solution for a 4×4 decomposition of the problem using optimized transmission conditions. The convergence behvaior of the PSWR algorithm is identical to the convergence behavior of the SWR algorithm.

$$\mathscr{T}_{ij}[\cdot] = \left(\frac{d}{d\hat{n}} + p\right)[\cdot], \quad \mathscr{T}_{ji}[\cdot] = \left(\frac{d}{d\hat{n}} - p\right)[\cdot],$$

are used, where $\frac{d}{d\hat{n}}$ is the derivative in the normal direction, and $p = 1$. (A recurrsion formula is used to compute the transmission condition in lieu of discretizing the derivative in the normal direction). In each experiment a total of 16 full waveform iterations are completed. Timing results are obtained using the stampede supercomputer at the Texas Advanced Computing Center. Good parallel efficiency and speedup is observed in spite of the increase in the number of messages required by the PSWR algorithm. Note that the $4\times 4\times 1$ case is identically the SWR algorithm.

$N_x \times N_y \times N_k$	# cores	walltime	speedup	efficiency
$4\times 4\times 1$	16	293.02 seconds	1.00 ×	1.00
$4\times 4\times 2$	32	149.92 seconds	1.95 ×	0.98
$4\times 4\times 4$	64	75.48 seconds	3.89 ×	0.97
$4\times 4\times 8$	128	38.71 seconds	7.57 ×	0.95
$4\times 4\times 16$	256	23.90 seconds	12.26 ×	0.77

In our second scaling study, 1600x1600 grid points are decomposed into 16x16 non-overlapping domains domains for 400 total time steps. Again, a centered five point finite difference stencil, a backward Euler time integrator, and optimized transmission conditions are used. Good parallel efficiency and speedup is observed even with the increased synchronization/number of messages in the system.

$N_x \times N_y \times N_k$	# cores	walltime	speedup	efficiency
$16 \times 16 \times 1$	256	295.86 seconds	1.00 $\times$	1.00
$16 \times 16 \times 2$	512	155.98 seconds	1.90 $\times$	0.95
$16 \times 16 \times 4$	1024	77.10 seconds	3.84 $\times$	0.96
$16 \times 16 \times 8$	2048	43.20 seconds	6.85 $\times$	0.86
$16 \times 16 \times 16$	4096	26.65 seconds	11.10 $\times$	0.69

In the above computations, a linear solve on a sub-domain takes $O(10^{-2})$ seconds. This relatively small problems size was chosen (100×100 on each subdomain) so that communications would play a substantial role in the timing studies. The presented efficiencies can be improved by partitioning the problem to be more computationally expensive (i.e. more time is spent in the linear solve).

5 Conclusions

In this paper, we have reformulated classical Schwarz waveform relaxation to allow for pipeline-parallel computation of the waveform iterates, after an initial startup cost. Theoretical estimates for the parallel speedup and communication overhead are presented, along with strong scaling studies to show the effectiveness of the pipeline Schwarz waveform relaxation algorithm.

Acknowledgements This work was supported by in part by Michigan State University through computational resources provided by the Institute for Cyber-Enabled Research and AFOSR Grant FA9550-12-1-0455. This work also used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

References

1. Bennequin, D., Gander, M.J., Halpern, L.: A homographic best approximation problem with application to optimized Schwarz waveform relaxation. Math. Comp. **78**(265), 185–223 (2009). DOI 10.1090/S0025-5718-08-02145-5. URL `http://dx.doi.org/10.1090/S0025-5718-08-02145-5`
2. Bjørhus, M.: On domain decomposition, subdomain iteration and waveform relaxation. Ph.D. thesis, PhD thesis, University of Trondheim, Norway (1995)
3. Christlieb, A., Macdonald, C., Ong, B.: Parallel high-order integrators. SIAM J. Sci. Comput. **32**(2), 818–835 (2010)
4. Courvoisier, Y., Gander, M.J.: Time domain maxwell equations solved with schwarz waveform relaxation methods. In: Domain Decomposition Methods in Science and Engineering XX, pp. 263–270. Springer (2013)
5. Feamster, N., Balakrishnan, H., Rexford, J., Shaikh, A., van der Merwe, J.: The case for separating routing from routers. In: Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture, FDNA '04, pp. 5–12. ACM, New York, NY, USA (2004). DOI 10.1145/1016707.1016709. URL `http://doi.acm.org/10.1145/1016707.1016709`

6. Gander, M.J., Halpern, L., Nataf, F., et al.: Optimal convergence for overlapping and non-overlapping schwarz waveform relaxation. In: the Eleventh International Conference on Domain Decomposition Methods, CH. Lai, P. Bjørstad, M. Cross, and O. Widlund, eds, pp. 27–36. Citeseer (1999)
7. Gander, M.J., Stuart, A.M.: Space-time continuous analysis of waveform relaxation for the heat equation. SIAM J. Sci. Comput. **19**(6), 2014–2031 (1998). DOI 10.1137/S1064827596305337
8. Giladi, E., Keller, H.B.: Space-time domain decomposition for parabolic problems. Numer. Math. **93**(2), 279–313 (2002). DOI 10.1007/s002110100345
9. Japhet, C., Omnes, P.: Optimized schwarz waveform relaxation for porous media applications. In: Domain Decomposition Methods in Science and Engineering XX, pp. 585–592. Springer (2013)
10. Lemarié, F., Patrick, M., Debreu, L., Blayo, E.: Sensitivity of an Ocean-Atmosphere Coupled Model to the Coupling Method : Study of Tropical Cyclone Erica. URL `http://hal.inria.fr/hal-00872496`
11. Tipparaju, V., Gropp, W., Ritzdorf, H., Thakur, R., Traff, J.: Investigating high performance rma interfaces for the mpi-3 standard. In: Parallel Processing, 2009. ICPP '09. International Conference on, pp. 293–300 (2009). DOI 10.1109/ICPP.2009.54