Reduced Communication for Faster and Energy-Efficient Parallel Computing To Solve Partial Differential Equations [⋆]

Soumyadip Ghosh [a], Kamal K. Saha [b], Vijay Gupta [a], Gretar Tryggvason [c]

[a] *Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA*

[b] *Center for Research Computing, University of Notre Dame, Notre Dame, IN 46556, USA*

[c] *Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218, USA*

Abstract

One of the many uses of parallel computing is to numerically solve partial differential equations. Such numerical simulations involve communication of data among the parallel processing elements. A typical implementation requires such communication at every iteration of the numerical algorithm. As the number of processors increases, the time and energy required for communication turns out to be a major portion of the overall simulation time and energy. Developing strategies to reduce communication without compromising on the quality of the solution are, thus, an important research area. In this paper, we cast the parallel numerical solution as a problem of reaching consensus in a multi-agent system. Consequently, we propose two "relaxed communication" schemes inspired from consensus in multi-agent systems – periodic and event-triggered – to reduce communication and, thus, save on simulation time and energy while guaranteeing convergence to the same solution. We model the system as a switched dynamical system and analyze properties such as stability and rate of convergence of the resulting numerical algorithm. The reduction in simulation time and communication energy due to reduced communication is shown through numerical experiments.

Key words: Multi-agent consensus; Numerical Algorithms; Parallel Computing

1 INTRODUCTION

A popular application of parallel computing is the numerical solution of partial differential equations (PDEs). In most implementations, the spatial domain is usually divided into many processing elements (PEs). In order to compute the space derivatives of the PDE at the boundaries of a PE, the boundary values of the PE which have been assigned neighboring sub-domains need to be communicated. Further, all the PEs need to exchange information among each other to guarantee that the stopping criterion for convergence of the numerical solver has been met across the entire domain. Both these forms of communication – *local* communication of boundary values among *neighboring* PEs and *global* communication of convergence criterion among all the PEs – are typically associated with a synchronization operation which makes a PE wait for values that are arriving from other PEs before proceeding with the next iteration. Especially as the number of PEs increase, time spent on such communication and synchronization typically becomes a major portion of the overall simulation time, e.g, Bergman et al. (2008). Motivated by this observation, it is of great importance to find ways to reduce communication and the associated synchronization requirements in implementations of numerical algorithms.

Among the parallel computing community, researchers have focused on two approaches to tackle this problem. The first approach is that of *asynchronous* algorithms. These algorithms still carry out local communication at every iteration, but relax the synchronization requirement for local communication. This means that the PEs still send data at every iteration resulting in local communication, but do not wait for the updated values to be received from other PEs before starting their next iteration. Instead, they proceed with the last values that were communicated to it. For a survey of asynchronous methods, the reader is referred to Frommer & Szyld (2000), Bertsekas & Tsitsiklis (1989). Of relevance to this pa-

[⋆] Some of the results in this paper appeared in a preliminary form in Ghosh et al. (2018a,b, 2019).

Email addresses: sghosh2@nd.edu (Soumyadip Ghosh), ksaha@nd.edu (Kamal K. Saha), vgupta2@nd.edu (Vijay Gupta), gtryggv1@jhu.edu (Gretar Tryggvason).

per is the application of asynchronous methods to finite difference schemes to solve PDEs by Donzis & Aditya (2014), Aditya & Donzis (2017), Lee et al. (2015).

While asynchronous schemes save simulation time, they still assume that the PEs communicate at every iteration. A second class of algorithms have been proposed that relaxes the requirement of communication at every iteration, see Demmel et al. (2008). These algorithms, popularly known as *communication-avoiding* (CA) algorithms, not only save on time but also on energy spent on communication. Due to reduction in communication, the synchronization associated with communication is also naturally reduced.

In this paper, we cast the parallel numerical solution of PDEs as a problem of reaching consensus in a multi-agent system. There has been a lot of work focused on reducing the communication between agents in such problems, see Nowzari et al. (2019) for a survey. Motivated by these techniques, we design new algorithms for reduced communication in parallel computing. While the algorithms we propose, also lie in the general class of CA algorithms, we propose novel algorithms by borrowing the techniques of periodic and event-triggered communication from consensus literature to design newer CA schemes that are amenable to analysis of stability, convergence rate and numerical error. We focus on the Jacobi method for our analysis and experiments, however the techniques we propose apply more generally. More specifically,

- We first propose a periodic communication algorithm where the local communication for boundary values and global communication for exchange of convergence criterion occurs periodically. Using techniques from control literature such as lifting, we study the stability and convergence of the algorithm.
- We then propose an event-triggered communication algorithm where the local communication happens in an event-triggered fashion. We choose a threshold-based discrete event-triggered communication scheme, e.g. Demirel et al. (2017), where an event for communication happens if the value to be communicated has changed from the last communicated value by some threshold. With the assumption of a bound on the outdated values from the neighbor PEs, we show that a numerical solver with event-triggered communication is stable and converges to the same solution as that with regular communication.

While an event-triggered communication scheme seems entirely new in the context of parallel computing, we note Hoemmen (2010) and Lee & Bhattacharya (2016) as relevant and pioneering works towards the use of periodic communication in this area. However, our results are quite different from these works. For instance, Hoemmen (2010) develops s-step Krylov subspace solvers where s is the period of communication but concludes that the algorithm can be unstable for high values of s. We show that our Jacobi solver with periodic communication is always stable irrespective of the period of communication. On the other hand, Lee & Bhattacharya (2016) design iterative solvers for quadratic optimization using periodic communication. They are able to prove that the solver with periodic communication converges only for very high values of period of communication, while our proof of convergence holds for any period of communication. We also note that a direct computation of values from one period to another without iterating for intermediate values between periods as considered in Lee & Bhattacharya (2016) may not be tractable for large-scale PDE solvers. On the contrary, we explicitly consider intermediate iterations and take them into account in our results on simulation time.

Part of our work on periodic communication has been presented in Ghosh et al. (2018a). We extend that work here by providing a more general proof of asymptotic stability and extensive numerical experiments. Initial results of our work on event-triggered communication was presented in Ghosh et al. (2019) with implementation details in Ghosh et al. (2018b). In this paper, we develop the idea further and illustrate it with more numerical experiments.

The paper is organized as follows. We begin in Section 2 by formulating the PDE problem that we use for our analysis and experiments. Section 3 describes the two main algorithms proposed in this paper with further analysis in Sections 4 and 5. Section 6 demonstrates the efficiency of the proposed algorithms through numerical experiments. Section 7 concludes the paper with scope of future work.

Notation: The symbol $||.||_\infty$ stands for the infinity norm of a vector or matrix, as will be clear from the context. I denotes an identity matrix of appropriate order. $\mathbb{R}^n$ denotes the n-dimensional real space. $\rho(M)$ and $\det(M)$ denotes the spectral radius and the determinant of a matrix M respectively.

2 PROBLEM FORMULATION

PDEs and their discretization: Our focus is on PDEs that can be expressed in the form

$$\rho S(x) + \xi \frac{\partial u}{\partial t} = \sum_{d=1}^{D} \alpha_d(x) \frac{\partial^d u}{\partial x^d}, \tag{1}$$

on a bounded domain with Dirichlet boundary conditions, where $u(x,t) \in \mathbb{R}$ is the dependent variable, x and t denote continuous space and time respectively, D is the highest order of the derivative, and $S(x)$ is a spatially varying function. Dirichlet boundary conditions specify the fixed values that the solution to the PDE needs to take along the boundary of the domain.

Remark 1 *The analysis described in this paper for the type of PDE in* (1) *can be easily extended to multiple dimensions* (x, y, z) *with minor modifications; hence, we focus on the 1-D case for notational ease.*

The PDE in (1) is a specific but widely studied class of PDEs. Note that choosing $\xi = 1$ in (1) yields a time-dependent PDE while $\xi = 0$ describes a time-independent PDE. An example of time-dependent PDE is obtained from (1) by setting $D = 2$, $\alpha_1 = 0$, $\alpha_2 \neq 0$, $\xi = 1$ and $\rho = 0$ which yields the diffusion (or heat) equation. Similarly, an example of time-independent PDE is obtained by setting $D = 2$, $\alpha_1 = 0$, $\alpha_2 \neq 0$, $\xi = 0$ and $\rho = 1$ in (1), yielding the Poisson equation.

We assume that the PDE in (1) is solved using the finite difference method with a forward difference in time and central difference in space (FTCS) scheme. The resolution of discretization in time and space are chosen to be Δt and Δx respectively. Denote by $u_i(k)$ the value of the dependent variable at the k-th discrete time step ($k = 0, 1, \cdots$) and i-th grid space point ($i = 1, \cdots, L$). For convenience, define $\mu = \lceil \frac{D}{2} \rceil$. To enforce Dirichlet boundary conditions, let $u_i(k)$ be constant values at every time k for $i = 1, \cdots, \mu$ and $i = L - \mu + 1, \cdots, L$. Then, (1) can be discretized as

$$\rho S_i + \xi \frac{u_i(k+1) - u_i(k)}{\Delta t} = \alpha_1 \left(\frac{u_{i+1}(k) - u_{i-1}(k)}{2\Delta x} \right) + \alpha_2 \left(\frac{u_{i+1}(k) - 2u_i(k) + u_{i-1}(k)}{\Delta x^2} \right) + \cdots, \quad (2)$$

for $i = \mu + 1, \cdots, L - \mu$ and $k \geq 0$, where S_i is the value of $S(x)$ at the i-th spatial grid point.

For time-dependent PDEs where $\xi = 1$, we group the coefficients in the discretized equation (2) and obtain

$$u_i(k+1) = \gamma_1 u_{i-\mu}(k) + \cdots + \gamma_{\mu+1} u_i(k) + \cdots + \gamma_{D+1} u_{i+\mu}(k) + \gamma_S S_i, \quad (3)$$

for suitably defined coefficients $\{\gamma_i\}'$s that are functions of $\{\alpha_j\}'$s, Δt and Δx. Note that the FTCS discretization of the time-dependent PDE leads to an explicit numerical scheme (Thomas (2013)). For time-independent PDEs where $\xi = 0$, a similar procedure of grouping coefficients in (2) yields an equation of the form

$$S_i = \gamma_1 u_{i-\mu} + \cdots + \gamma_{\mu+1} u_i + \cdots + \gamma_{D+1} u_{i+\mu}, \quad (4)$$

for appropriately defined coefficients $\{\gamma_i\}'$s.

Matrix-vector formulation: We now formulate the iterations (3) and (4) in a matrix form.

- For the time-dependent PDE in (3), we stack the variables $u_i(k)$ for $i = \mu + 1, \cdots, L - \mu$ to define a state vector $U(k) \in \mathbb{R}^{(L-2\mu)}$. Then (3) can be written in short as
$$U(k+1) = \bar{A} U(k) + \bar{b}, \quad (5)$$
where $\bar{A} \in \mathbb{R}^{(L-2\mu)\times(L-2\mu)}$ is the iteration matrix and $\bar{b} \in \mathbb{R}^{(L-2\mu)}$ is a constant that contains terms corresponding to $u_i(k)$ for $i = 1, \cdots, \mu$ and $L - \mu + 1, \cdots, L$ (i.e., the boundary conditions) and the source S_i. The initial condition for this time-dependent problem is represented by $U(0)$.
- For the time-independent PDE in (4), we can similarly define U by stacking the variables u_i for $i = \mu + 1, \cdots, L - \mu$ and rewrite (4) as
$$MU = v, \quad (6)$$
for a suitably defined matrix M and vector v. Often the equation (6) is solved using an iterative solver. A popular class of iterative solvers are relaxation-type iterative solvers such as Jacobi, Gauss-Seidel or Successive Over-relaxation (Strang & Aarikka (1986)). In this paper, we assume that the Jacobi method is being used. In this case, we write $M = D + R$ where D is a diagonal matrix and R is a matrix with zeroes along the diagonal. If we define $\hat{A} = -D^{-1}R$ and $\hat{b} = D^{-1}v$, then (6) can be solved iteratively as
$$U(k+1) = \hat{A} U(k) + \hat{b}. \quad (7)$$

Given the similarity of iterations (5) for time-dependent PDEs and (7) for time-independent PDEs, from now on we will consider the system described by

$$U(k+1) = AU(k) + b. \quad (8)$$

An interesting property to note is that A has a banded Toeplitz structure.

Stability of numerical algorithm: The numerical algorithm in (8) is said to be asymptotically stable if the values of $U(k)$ converge to a stationary point U^* as $k \to \infty$.

Assumption 1 *The spectral radius of* A *in* (8), *denoted by* $\rho(A)$, *is less than* 1.

Assumption 1 also implies that A is Schur diagonally stable (Kaszkurewicz & Bhaya (2012)). From the theory of linear time-invariant dynamical systems, we know that Assumption 1 is a necessary and sufficient condition for asymptotic stability of (8).

Parallelization: For solving (8) using parallel computing, we decompose the spatial grid among the N PEs. For simplicity and without loss of generality, we assume that each PE has an identical number n of consecutive grid points. We find it useful to denote the value of the dependent variable on the i-th grid point ($1 \leq i \leq n$) in the I-th PE ($1 \leq I \leq N$) by u_i^I. However, the grid points for $i = 1, \ldots, \mu$ for 1st PE and for $i = L - \mu, \ldots, L$ for N-th PE are not considered owing to the specification of the Dirichlet boundary conditions. Further, denote the vector of the values of the dependent variable for every grid point in the I-th PE by $U_I(k) \triangleq [u_1^I(k), u_2^I(k), \ldots, u_n^I(k)]^\top \in \mathbb{R}^{n\times 1}$ with suitable modifications for the I-th and N-th PEs. Finally, the entire vector of the dependent variable values for every grid point $U(k)$ can be expressed as $[U_1(k)^\top, U_2(k)^\top, ..., U_N(k)^\top]^\top$. Since computation of the space derivatives at the PE boundaries requires values from PEs assigned neighboring sub-domains, communication of these values have to take place before the next iteration begins. In typical parallel computation, this communication happens at every iteration k. Let $\mathcal{B}_{IJ}$ denote the set of all boundary points which the I-th PE receives from J-th PE. This exchange of values with neighboring PEs results in *local* communication. We define the *state* of a PE as the values of the grid points in its sub-domain. In other words, the values of $U_I(k)$ are considered to be the state of the I-th PE at iteration k. Further, we find it convenient to decompose A into blocks as $A = [A_{IJ}]_{1\leq I\leq N, 1\leq J\leq N}$ where A_{II} corresponds to the independent state of the I-th PE and A_{IJ} is the state of I-th PE that is dependent on the J-th PE. Then, (8) can be rewritten as

$$U_I(k+1) = A_{II}U_I(k) + \sum_{J\in\mathcal{N}_I} A_{IJ}U_J(k) + b_I, \quad (9)$$

for all $1 \leq I \leq N$ where b_I is the block of b corresponding to the I-th PE and $\mathcal{N}_I$ is the set of neighbors of the I-th PE. We find it useful to collect all the diagonal entries of A into a separate matrix A_1 and define $A_2 = A - A_1$.

For any numerical solver, some metric has to be monitored as a stopping criterion. When this metric decreases to a specified value called tolerance, the iterations are stopped and the numerical solver is considered to have converged. Two popular choices of this convergence metric are the change in iteration defined as $\mathbb{E}(k) = U(k+1) - U(k)$ and residual defined as $R(k) = MU(k) - v$. Every PE can calculate only the local convergence metric for the sub-domain allocated to it. In order to ensure that the specified level of tolerance has been met uniformly throughout the entire domain, all the PEs need to exchange the local convergence information among themselves to come to a consensus regarding global convergence information. Such an operation, which is typically done at every iteration, results in *global* communication. Since the PEs cannot proceed to the next iteration before checking the global convergence criterion, global communication also keeps the iterations synchronized, ensuring that all PEs execute the same iteration at the same time. Now we state the pseudo code of a typical parallel algorithm in the I-th PE in Algorithm A. The evolution of the values at various grid points described by this parallel algorithm follows the same dynamical system as described for the centralized algorithm in (8). Subsequently, if the centralized algorithm converges to a value, this parallel algorithm will converge to the same value in the same number of iterations. We note that two forms of communication, local and global, take place at every iteration of the numerical solver. We call this the *regular* algorithm because it involves a regular pattern of communication at every iteration.

Algorithm A: Regular Communication

```
do
    Compute $U_I(k+1)$ as in (9)
    Communicate boundary values to neighbors
    Calculate local convergence criterion
    Global convergence criterion
        $\leftarrow \max_{1\leq I\leq N}$(local convergence criterion)
while Global convergence criterion > tolerance
```

Analogy to Consensus: We note that our problem is analogous to the problem of consensus in multi-agent systems. From Olfati-Saber et al. (2007), consider the formulation $U_I(k+1) = U_I(k) + R_I(k)$ where $U_I(k)$ is the state of agent I and $R_I(k) = -\sum_{j\in\mathcal{N}_I}(U_I(k) - U_J(k))$ is the state-feedback input required to reach consensus. Then the dynamics of each agent is obtained as $U_I(k+1) = \tilde{P}_{II}U_I(k) + \sum_{J\in\mathcal{N}_I}\tilde{P}_{IJ}U_J(k)$ where $\tilde{P}$ is a suitably defined matrix. This is very similar in form to (9). We can similarly draw an analogy between the objective of reaching the average of the initial states of all agents in consensus problems and to reaching the tolerance value of the convergence metric in our problem.

It has been observed that both the local communication for exchange of boundary values and global communication for determination of convergence criterion usually turn out to be more expensive in terms of time than computation in a parallel computing system with many PEs. This means that the solver spends a lot of time communicating data among the PEs rather than doing useful computation. We propose two methods of alleviating this problem by considering the following analogy.

3 ALGORITHM DESCRIPTION

Having noted the analogy to consensus as above, we can then also take motivation from literature on consensus under communication constraints (Nowzari et al. (2019)) to relax the requirement of communication at every iteration. Instead, we propose to initiate communication based on a criterion. When the criterion is not satisfied, computation proceeds with the last communicated values. This corresponds to a zero-order hold at the receiver. Because the criterion for communication is not satisfied at every iteration, the sender PE does not send values at every iteration. Consequently, the receiver PE also does not receive a value at every iteration. As a result of this relaxation in communication, the update equation in (9) is modified as

$$U_I(k+1) = A_{II}U_I(k) + \sum_{J\in\mathcal{N}_I} A_{IJ}U_J(\tau_{IJ}(k)) + b_I, \quad (10)$$

where $\tau_{IJ}(k)$ denotes the last iteration before or including iteration k at which PE I receives a value from PE J. The value $\tau_{IJ}(k)$ is dependent on the criterion for communication which can be quite general. In this paper, we consider two types of criteria described below.

3.1 Periodic Communication

Our first algorithm is one in which both local and global communication happens periodically with a period r. In between the iterations involving communication, the receiving PE keeps using the value that was last communicated to it. Since both the sender and the receiver are aware of the instants at which communication happens, this algorithm can be implemented using standard MPI two-sided communication (Gropp et al. (1999)).

Formally, in this algorithm for every iteration k, $\tau_{I,J}(k) = k - (k \mod r)$. Note that $\tau_{IJ}(k)$ is independent of I or J in this algorithm. The regular update equation in (8) can thus be written as

$$U(k+1) = A_1 U(k) + A_2 U\left(k - (k \mod r)\right) + b. \quad (11)$$

As stated above, we make the global communication also periodic with the same period r. As a result, the iterations in all the PEs are synchronized at the iterations involving communication. This means that even if a PE is faster and completes more iterations than the others, it has to wait at the point of global communication for the other PEs to catch up to the same iteration. The pseudo code for the algorithm with periodic communication is shown in Algorithm B. Henceforth we refer to this algorithm with periodic communication as the *periodic* algorithm for convenience. Note that the regular algorithm in (8) corresponds to $r = 1$.

Algorithm B: Periodic Communication

```
do
    Compute $U_I(k+1)$ as in (11)
    if $k$ mod $r$ == 0 then
        Communicate boundary values to neighbors
        Calculate local convergence criterion
        Global convergence criterion
            $\leftarrow \max_{1 \le I \le N}$(local convergence criterion)
    end if
while Global convergence criterion > tolerance
```

3.2 Event-Triggered Communication

Our second algorithm triggers communication based on the state of a PE. While there can be many choices on how to choose the event triggering the communication, we design it based on the change in values (from the last communicated values) of the points on the boundary of the sub-domain allocated to the sending PE. When this change exceeds some user specified threshold δ, the values are communicated to the PEs that have been assigned the neighboring sub-domain; otherwise the intended receiver PE keeps using the values last communicated from the sender PE. Since occurrence of an event that triggers a communication is known only to the sending PE but not the receiving PE, this algorithm requires an advanced MPI implementation, called MPI one-sided (Gropp et al. (2014)). This implementation often leads to a situation when a message sent from the J-th PE at iteration k may be received at the I-th PE at an iteration later than k. The sequence $\tau_{IJ}(k)$ used in (10) is identified as the sequence received by the I-th PE. We also define a separate sending sequence $\lambda_{IJ}(k)$ where $\lambda_{IJ}(k)$ denotes the last iteration before or including iteration k at which PE J sends a value to PE I. Due to the delay in receiving a message, $\tau_{IJ}(k)$ may be different from $\lambda_{IJ}(k)$. Further, this delay is usually stochastic and varies across different iterations and different PE boundaries. As a result of this stochastic delay, the updates in the numerical scheme becomes stochastic. As a simplification, we consider the following assumption about this delay:

Assumption 2 *The delay in receiving a message is bounded by a finite non-negative integer d.*

The event-triggering condition determines the sending sequence $\lambda_{IJ}(k)$. Formally, we define

$$\lambda_{IJ}(k) = \begin{cases} k, & \text{if}\,(u_{\mathcal{B}_{IJ}}(k) - u_{\mathcal{B}_{IJ}}(\lambda_{IJ}(k-1))) \ge \delta \\ \lambda_{IJ}(k-1), & \text{otherwise,} \end{cases} \quad (12)$$

where δ is a designer specified threshold. The condition (12) means that when the change in the boundary value exceeds a threshold, the current boundary value is sent to the neighboring PE.

Note that with this rule, when the boundary grid points have almost reached their steady state, no more events for communication will be triggered. In other words, the change in boundary values near the steady state does not exceed the threshold δ. Fig 1 illustrates this phenomenon. In order to fix this phenomenon and ensure that the messages are sent infinitely often as needed for asymptotic convergence, we enforce a lower bound on the last received message. To this effect, we make the following assumption:

Assumption 3 *To ensure that $\lambda_{IJ}(k)$ is lower bounded by a positive constant, we superimpose periodic communication with period r.*

With this assumption, the event-triggering criterion in (12) can be rewritten as:

$$\lambda_{IJ}(k) = \begin{cases} k, & \text{if}\,(u_{\mathcal{B}_{IJ}}(k) - u_{\mathcal{B}_{IJ}}(\lambda_{IJ}(k-1))) \ge \delta \\ & \text{OR}\, k \bmod r == 0 \\ \lambda_{IJ}(k-1), & \text{otherwise,} \end{cases} \quad (13)$$

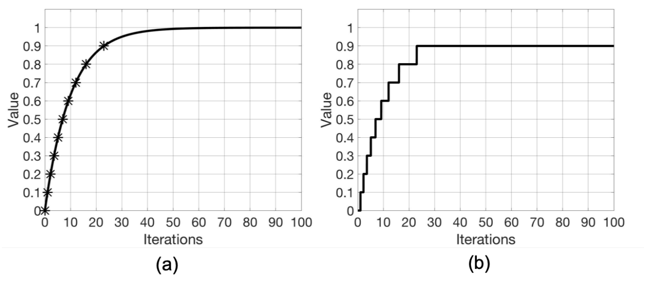

Fig. 1. The basic idea showing the first attempt of event-triggered communication. (a) shows the evolution of the boundary values over iterations with the asterisks showing the points where the threshold criterion triggers the communication (threshold of 0.1 shown here). (b) shows the values used by the receiving PE. Note that after the value crosses 0.9 in (a), an event for communication fails to trigger because the value never crosses the threshold of 0.1. As a result, the receiver does not receive updated values after 0.9.

This criterion states that an event for communication happens when the boundary changes by δ or when the periodic condition is satisfied. The corresponding message is then received at iteration $\tau_{IJ}(k)$.

Assumptions 2 and 3 ensure that $\tau_{IJ}(k)$ is lower bounded by $k-r-d+1$. This intuitively means that the boundary values from neighbor PEs continue to be received at least once within $r+d$ steps even if the event triggering condition is not satisfied. Such an assumption of periodic communication on top of event-triggered communication has been made in the control literature to safeguard against faulty components (Demirel et al. (2017)). Note that imposing the periodic assumption on top of the event-triggered scheme is different from periodic event-triggered control which evaluates the event-triggering condition periodically (Heemels et al. (2013)).

We note that $\lambda_{IJ}(k)$ varies with I, J, k, except when the condition $k \mod r == 0$ is satisfied. In other words, when $k \mod r == 0$, $\lambda_{IJ}(k) = k$ for all I, J in the domain, meaning that all PEs perform local communication to exchange boundary information. Since $k \mod r == 0$ is thus a global event for local communication in the entire domain, we consider the global communication to also take place at these iterations. The pseudo code for this algorithm is given in Algorithm C. We refer to the algorithm with event-triggered communication as the *event-triggered* algorithm for the sake of convenience. Note that the regular algorithm in (9) is given by this same pseudo code with $r = 1$ and $\delta = 0$.

4 CONVERGENCE ANALYSIS OF PERIODIC ALGORITHM

In this section, we present the convergence properties of Algorithm B.

Algorithm C: Event-Triggered Communication

```
do
    Compute $U_I(k+1)$ as in (10)
    if $(u_{\mathcal{B}_{IJ}}(k) - u_{\mathcal{B}_{IJ}}(\lambda_{IJ}(k-1)) \geq \delta || k \mod r == 0)$
then
        Communicate boundary values to neighbors
    end if
    if $k \mod r == 0$ then
        Calculate local convergence criterion
        Global convergence criterion
            $\leftarrow \max_{1\leq J\leq N}$(local convergence criterion)
    end if
while Global convergence criterion > tolerance
```

4.1 Formulation as a Linear Periodic System

Following the usual treatment of linear periodic systems (Bittanti & Colaneri (2009)), define the augmented state $X(k) \in \mathbb{R}^{(L-2\mu)r} \triangleq [U(k)^\top, U(k-1)^\top, ..., U(k-r+1)^\top]^\top$. Also define $B \in \mathbb{R}^{(L-2\mu)r} \triangleq [b, 0, \cdots, 0]^\top$. We can then define r matrices $H_1, H_2, \ldots, H_r \in \mathbb{R}^{(L-2\mu)r\times(L-2\mu)r}$, such that the system (11) evolves as

$$X(k+1) = H_j X(k) + B, \quad j = (k \mod r) + 1. \quad (14)$$

Note that the augmented state involves the terms $U(-1), \cdots, U(-r+1)$ which are assumed to be equal to $U(0)$. For simplicity, we define $m \geq 0$ as the index of the iterations involving local and global communication. For example, if $r = 3$, communication happens at iterations $k = 0, 3, 6, 9, \ldots$ which correspond to $m = 0, 1, 2, 3, \ldots$ respectively. The evolution of the state for every r iterations can thus be obtained as

$$X((m+1)r) = WX(mr) + VB. \quad (15)$$

where $W \triangleq H_r \ldots H_2 H_1$ and $V \triangleq \sum_{j=2}^{r-1} H_r \cdots H_j + I$. (15) is a time-invariant system that is stable if and only if the original periodic system (11) is stable. Since this is an easier system to analyze, we concentrate on the stability properties of (15) from here on. Before proceeding further, we note the following properties of the matrices H_i and W.

Lemma 1 *Consider the matrices $H_1, H_2, \ldots, H_r$ and W as defined above. Then, it holds that*

- $\|H_j\|_\infty = 1, \ \forall\, j = 1, \cdots, r.$
- $\rho(W) \leq 1.$

PROOF. The first claim follows by inspection of the matrices H_j's. The second claim follows from the sub-multiplicativity property of the infinity-norm as $\rho(W) \leq ||W||_\infty = ||H_r \ldots H_1||_\infty \leq ||H_r||_\infty \cdots ||H_1||_\infty = 1.$ □

4.2 Asymptotic Stability

Now we examine the asymptotic stability of (15). We begin with the following result that simplifies the system we need to study.

Lemma 2 *Define* $C \triangleq A_1^r + \sum_{j=0}^{r-1} A_1^j A_2$ *and* $\tilde{b} \triangleq \sum_{j=0}^{r-1} A_1^j b$ *The system* (15) *is equivalent in asymptotic stability to the system*

$$U(m+1) = CU(m) + \tilde{b}. \tag{16}$$

PROOF. The proof follows by expanding (15) and is omitted for space constraints. □

Lemma 3 *The matrices* $(I - A_1)$ *and* $(I - A_1^r)$ *commute.*

PROOF. Note that $(I - A_1)(I - A_1^r) = I - A_1^r - A_1 + A_1^{r+1}$ and $(I - A_1^r)(I - A_1) = I - A_1 - A_1^r + A_1^{r+1}$. Since the two products are same, the matrices commute. □

Theorem 1 *Consider the time invariant system given in* (16)*. The system converges to the same value as the regular algorithm in* (8)*.*

PROOF. For the regular case when $r = 1$, $C = A_2 + A_1 = A$ and $\tilde{b} = b$. Hence the fixed point value is given by $U^* = (I - A)^{-1}b$. For period r, the fixed point value is given by

$$\begin{aligned}
U &= (I - C)^{-1}\tilde{b} \\
&= \left\{ I - \left(A_1^r + \sum_{i=0}^{r-1} A_1^i A_2 \right) \right\}^{-1} \sum_{i=0}^{r-1} A_1^i b \\
&= \left\{ \left(\sum_{i=0}^{r-1} A_1^i \right)^{-1} - \left(\sum_{i=0}^{r-1} A_1^i \right)^{-1} A_1^r - A_2 \right\}^{-1} b \\
&= \left\{ \left(\sum_{i=0}^{r-1} A_1^i \right)^{-1} (I - A_1^r) - A_2 \right\}^{-1} b \\
&= \left\{ \{(I - A_1)^{-1}(I - A_1^r)\}^{-1}(I - A_1^r) - A_2 \right\}^{-1} b \\
&= \left\{ (I - A_1^r)^{-1}(I - A_1)(I - A_1^r) - A_2 \right\}^{-1} b \\
&\overset{(a)}{=} \left\{ (I - A_1^r)^{-1}(I - A_1^r)(I - A_1) - A_2 \right\}^{-1} b \\
&= \left(I - A_1 - A_2 \right)^{-1} b \\
&= \left(I - A \right)^{-1} b = U^*.
\end{aligned}$$

□

where (a) follows from Lemma 3. The above proof is a general version of the proof specified in our previous work Ghosh et al. (2018a) which had an assumption on linearly independent eigenvectors. We also note that Lee & Bhattacharya (2016) provides a similar proof which only holds for large enough r. Our proof holds for any value of r.

4.3 Marginal Stability

The time invariant system in (16) models the behavior of the periodic algorithm at the points of communication. However, it does not provide information about the iterations in between points of communication. This motivates us to check if the periodic communication has any effect on stability during the iterations in between points of communication. To this effect, we use the augmented steady state error for our analysis. Define $X^* = [U^*, \cdots, U^*]$ as the augmented steady state. Also define the augmented steady state error of (14) $G(k) \in \mathbb{R}^{(L-2\mu)r}$ as

$$G(k) = X(k) - X^*. \tag{17}$$

Lemma 4 *The augmented steady state error system* $G(k)$ *follows the dynamics*

$$G(k+1) = H_j G(k) \quad \forall \quad j = 1, \cdots, r. \tag{18}$$

PROOF. First we note that by definition of steady state, we obtain $X^* = H_j X^* + B \quad \forall \quad j = 1, \ldots, r$. Therefore,

$$\begin{aligned}
G(k+1) &= X(k+1) - X^* \\
&= H_j X(k) + B - H_j X^* - B \\
&= H_j (X(k) - X^*) \\
&= H_j G(k).
\end{aligned}$$

□

Theorem 2 *The system* (18) *is marginally stable.*

PROOF. We use the sub-multiplicative property of the infinity-norm to obtain $||G(k+1)||_\infty = ||H_j G(k)||_\infty \leq ||H_j||_\infty ||G(k)||_\infty$. Using first result in Lemma 1, we obtain $\frac{||G(k+1)||_\infty}{||G(k)||_\infty} \leq 1$, which implies that the system is marginally stable. □

4.4 Rate of convergence

In this subsection, we study the rate of convergence of the periodic algorithm. While we have established asymptotic convergence for the periodic algorithm to the same final solution irrespective of the value of r, the rate of convergence does vary with r. For this analysis, first we define the error w.r.t the steady state for the system in (8) as

$$E(k) = U(k) - U^*. \tag{19}$$

Lemma 5 *For the regular Algorithm A,* $E(k)$ *follows the dynamics*

$$E(k+1) = AE(k). \tag{20}$$

PROOF. The proof is similar to that of Lemma 4 and is omitted for space constraints. □

For the periodic Algorithm B, $E(k)$ evolves as $E(k+1) = A_1 E(k) + A_2 E\left(k - (k \mod r)\right)$. Note that the augmented version of the error $E(k)$ is $G(k)$ as defined in (17).

Corollary 1 (to Lemma 5) *The evolution of error system between the points of communication is given by $E(m+1) = CE(m)$ where m is the index of the iterations involving communication defined before.*

A measure of the rate of convergence of the state in the periodic Algorithm B is the spectral radius of C in (16). The rate of convergence influences the number of iterations taken to converge to a specified level of tolerance. In order to consider that, we have to estimate when the convergence criterion reaches the tolerance ϵ. In Section 2, we have already mentioned two possible convergence criterion, namely, the change in iteration $\mathbb{E}(k)$ and the residual $R(k)$. Since we check the convergence criterion only at the iterations involving communication given by m in Algorithm B and Algorithm C, we modify the convergence criteria to the average change in iteration $\mathbb{E}(m) = \frac{U(m+1)-U(m)}{r}$ and the residual $R(m) = MU(m) - v$. We have the following result for the evaluation of these quantities.

Lemma 6 *The metric $\mathbb{E}(m)$ evolves as*

$$\mathbb{E}(m+1) = C\mathbb{E}(m).$$

PROOF. By definition, we can write

$$\begin{aligned}\mathbb{E}(m+1) &= \frac{U(m+2)-U(m+1)}{r}\\ &= \frac{E(m+2)+U^*-E(m+1)-U^*}{r}\\ &= \frac{(C-I)E(m+1)}{r} = \frac{(C-I)CE(m)}{r}.\end{aligned}$$

Similarly, we can write $\mathbb{E}(m) = \frac{(C-I)E(m)}{r}$. Combining $\mathbb{E}(m+1)$ and $\mathbb{E}(m)$, we obtain the desired expression. □

The residual $R(m)$ evolves as $R(m+1) = CR(m)$. The proof is similar and omitted due to space constraints.

When the convergence criterion reaches the tolerance ϵ, the numerical scheme is considered to have converged. We can estimate the number of iterations the numerical solver takes to converge. Let p be the number of iterations involving communication. If $\mathbb{E}(m)$ is used as the convergence criterion, we can write $||\mathbb{E}(p)||_\infty = \epsilon \implies ||C^p\mathbb{E}(0)||_\infty = \epsilon$. Then we can write $C^p \approx C_\zeta|\zeta|^p$ where $C_\zeta \in \mathbb{R}^{(L-2\mu)\times(L-2\mu)}$ is independent of p. Thus, we obtain $p \approx \frac{\ln\left(\frac{\epsilon}{||C_\zeta\mathbb{E}(0)||_\infty}\right)}{\ln\,|\zeta|}$. Similarly, when the convergence criterion is the residual, the development is the same as above and the final expression of p is $p \approx \frac{\ln\left(\frac{\epsilon}{||C_\zeta R(0)||_\infty}\right)}{\ln\,|\zeta|}$.

Since the convergence criterion is checked only at the iterations when communication happens, the total number of iterations till convergence is approximated by pr.

Remark 2 *The total simulation time taken by the numerical solver with periodic algorithm can be estimated. Denote the time taken for an iteration at which communication among the PEs happens as t_c. For iterations not involving communication, we assume that the time taken per iteration is qt_c where $0 < q < 1$. Both t_c and q can be approximately characterized for a given parallel computing set up. Now, p iterations involve communication among the PEs and $p(r-1)$ are communication avoiding iterations. Thus, the overall time for the solver is approximately given by $T = pt_c\,(1 + q(r-1))$.*

5 CONVERGENCE ANALYSIS OF THE EVENT-TRIGGERED ALGORITHM

We now analyze the event-triggered Algorithm C. Once again, we utilize the error system $E(k)$ defined in (19). It is straight forward to see that the $E(k)$ evolves as

$$E_I(k+1) = A_{II}E_I(k) + \sum_{J\in\mathcal{N}_I} A_{IJ}E_J(\tau_{IJ}(k)), \quad (21)$$

where $\tau_{IJ}(k)$ is given by (13).

5.1 Asymptotic Stability

The dynamical system in (21) is a discrete-time dynamical system that is time-varying. We analyze the stability of this time-varying system using a diagonal-type Lyapunov function specified in Kaszkurewicz & Bhaya (2012). Before our main stability proof, we need the following lemma:

Lemma 7 *There exists N positive real numbers W_I, $I = 1,\dots,N$ such that* $\max_I\{W_I^{-1}\sum_{J=1}^N W_J A_{IJ}\} < 1$.

PROOF. Since A is Schur diagonally stable, we obtain

$$\begin{aligned}&W_I(1-A_{II}) > \sum_{J\neq I} W_J A_{IJ}\\ &\implies W_{II} + \sum_{J\neq I}\frac{W_J}{W_I}A_{IJ} < 1 \implies \sum_{J=1}^N \frac{W_J}{W_I}A_{IJ} < 1\\ &\implies \max_I\{W_I^{-1}\sum_{J=1}^N W_J A_{IJ}\} < 1. \qquad \square\end{aligned}$$

Due to assumptions 2 and 3, we know that $\tau_{IJ}(k)$ belongs to the set $\{k, k-1, \dots, k-(r+d-1)\}$. For the sake of convenience, we define $E_I(k)$ as the error w.r.t the steady state for the I-th PE. We denote delayed versions of $E_I(k)$ as $E_{I,l}(k) = E_I(k-l)$ for all $l = 0,\dots,r+d-1$ and $I = 1,\dots,N$. Define $G_I(k) = [E_I(k)^\top, E_{I,1}(k)^\top, \dots, E_{I,r+d-2}(k)^\top, E_{I,r+d-1}(k)^\top]^\top$ for $I = 1,\dots,N$. Note that $G_I(k)$ is a subset of $G(k)$ defined in (17) that corresponds to values for the I-th PE. Further, we define functions $\Psi_{IJ}(G_J(k))$ that take the value of exactly one of the variables in the subset $G_J(k)$ for $J = 1,\dots,N$. Now we can represent (21) in a standard state space representation as follows:

$$\begin{aligned} E_I(k+1) &= \sum_{J=1}^{N} A_{IJ}\Psi_{IJ}(G_J(k)), \\ E_{I,1}(k+1) &= E_I(k), \\ E_{I,2}(k+1) &= E_{I,1}(k), \\ &\vdots \\ E_{I,r+d-1}(k+1) &= E_{I,r+d-2}(k). \end{aligned} \tag{22}$$

Theorem 3 *The system of equations in* (22) *is globally asymptotically stable for all* $I = 1, \ldots, N$.

PROOF. We consider the Lyapunov function $V(k) = \max_{I,l}\{W_I^{-1}||E_I(k)||_\infty, W_I^{-1}||E_{I,l}(k)||_\infty\}$. Then, from (22),

$$\begin{aligned} &V(k+1) \\ &= \max_{I,l}\left\{W_I^{-1}\left\|\sum_{J=1}^{N} A_{IJ}\Psi_{IJ}(G_J(k))\right\|_\infty, W_I^{-1}||E_{I,l}(k)||_\infty\right\} \\ &\le \max_{I,l}\left\{\sum_{J=1}^{N} W_J W_I^{-1} A_{IJ} W_J^{-1}\|\Psi_{IJ}(G_J(k))\|_\infty, \right. \\ &\qquad\qquad \left. W_I^{-1}||E_{I,l}(k)||_\infty\right\} \\ &\le \max_{I,l}\left\{\max_I\left\{W_I^{-1}\sum_{J=1}^{N} W_J A_{IJ}\right\}\right. \\ &\qquad \left.\times \max_J\left\{W_J^{-1}||\Psi_{IJ}(G_J(k))||_\infty\right\}, W_I^{-1}||E_{I,l}(k)||_\infty\right\} \\ &\overset{(b)}{<} \max_{I,l}\left\{\max_J\left\{W_J^{-1}||\Psi_{IJ}(G_J(k))||_\infty\right\}, W_I^{-1}||E_{I,l}(k)||_\infty\right\} \\ &\le \max_{I,l}\left\{W_I^{-1}||E_I(k)||_\infty, W_I^{-1}||E_{I,l}(k)||_\infty\right\} \\ &= V(k). \end{aligned}$$

where (b) follows from Lemma 7. Since $V(k+1) < V(k)$, the system is globally asymptotically stable. □

5.2 Rate of Convergence

To determine the rate of convergence of (21), the frequency of communication at the PE boundaries needs to be taken into account. Since every PE will trigger communication at different times depending on the threshold, it seems intractable to find a closed form for the rate of convergence and we leave it for future work. Instead, we point to a lower bound for the rate of convergence based on the period r in Assumption 3.

Proposition 1 *The rate of convergence of the system in* (21) *where* $\tau_{IJ}(k)$ *follows the event-triggered communication rule with threshold* δ *and period* r *is lower bounded by the reciprocal of the spectral radius of the time-invariant system matrix* C *corresponding to the same period* r *for the periodic algorithm specified in Section 4.*

6 SIMULATIONS

In this section, we demonstrate our proposed algorithms with numerical experiments on a PDE. We choose a 2D Poisson PDE which is used to express the electric potential resulting from a distribution of electric charges. The PDE is given by

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = -\frac{\rho}{\epsilon}, \tag{23}$$

where V is the electric potential, ρ is the electric charge and ϵ is the permittivity of free space equal to $8.854 \times 10^{-12} m^{-3} kg^{-1} s^4 A^2$. After discretizing the PDE by finite difference, we solve it numerically using Jacobi method on a parallel computing cluster. It can be shown that the iteration matrix A as in (8) for this PDE has spectral radius less than 1, thus satisfying Assumption 1. We follow a 1D domain decomposition, meaning that the domain is decomposed along one dimension among the multiple PEs. We use a cluster of nodes with each node having 2 CPU Sockets of AMD's EPYC 24-core 2.3 GHz processor and 128 GB RAM per node. The cluster uses Mellanox EDR interconnect. The MPI library chosen is mvapich2 built with Intel compiler. The length of the 2D domain is considered to be 1 unit in each dimension. The convergence criterion is chosen to be the maximum residual and the tolerance is taken to be 10^{-5}.

We choose a grid resolution of 960 in both dimensions and demonstrate the efficiency of our schemes. Fig 2 shows the reduction in overall simulation time of the periodic algorithm over various periods of communication. As the period is increased, there is reduction in simulation time due to time saved on communication. However, in order to compensate for the reduced communication, the solver takes more iterations to converge to the correct solution. When the effect of increased iterations starts dominating the effect of reduced time due to lesser communication, the simulation time starts increasing as is seen in Fig 2 for periods above 16.

The simulation time results for the event-triggered algorithm versus various thresholds for communication are shown in Fig 3. Note that in addition to the threshold, there is a parameter for the period in this algorithm. Period 1 corresponds to the benchmark solution where communication happens at every iteration, hence it takes the same simulation time across various thresholds. We note that time decreases initially with increasing threshold, implying lesser time spent on communication. A similar trend on reduction in time is noted when the period is initially increased. However, for higher periods and higher thresholds, messages are rarely exchanged, leading to a significant increase in number of iterations which results in higher simulation time. For instance, this effect is seen for period 64 above threshold

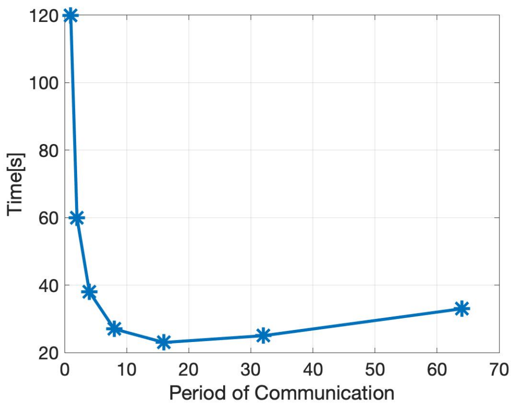

Fig. 2. Reduction in time using periodic algorithm running on 48 PEs. The data points for periods of {1,2,4,8,16,32,64} are shown here.

10^{-14}. A moderate value of threshold equal to 10^{-2} and period equal to 8 leads to the lowest simulation time.

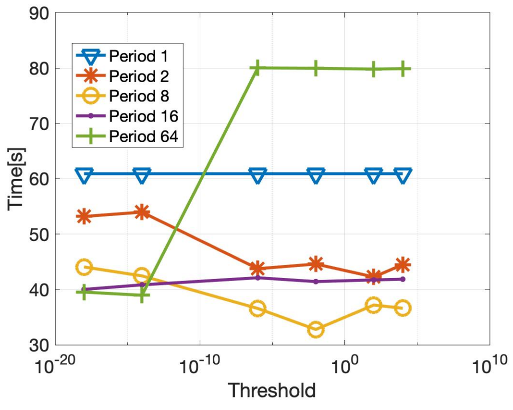

Fig. 3. Reduction in time using event-triggered algorithm running on 48 PEs. The data points for threshold of $\{10^{-18}, 10^{-14}, 10^{-6}, 10^{-2}, 10^{2}, 10^{4}\}$ are shown here.

Both the algorithms are also effective in saving energy. Firstly, they save running energy costs of the parallel computing cluster by completing simulations faster. This includes the energy for powering the cluster and auxiliary costs such as cooling. Further, they also save on energy spent due to communication. It is estimated that moving data between PEs connected by a network require around 1 to 3 pJ per bit (Bergman et al. (2008)). This means that a numerical scheme involving a significant number of iterations running on many PEs can save a significant amount of energy with this class of algorithms. We would like to quantify this savings of the energy spent due to communication. To the best of our knowledge, there is no direct way to measure the energy taken by the interconnect between nodes in a parallel cluster for measuring the energy spent in communication. Instead we look at an indirect measurement for quantifying savings in energy. Specifically we find the communication ratio for a specified algorithm as

$$\frac{\text{Number of messages exchanged with specified algorithm}}{\text{Number of messages exchanged with regular algorithm}}.$$

The lesser the communication ratio, the lesser the energy spent in communication. The communication ratios for both the periodic and event-triggered algorithms across various periods are shown in Table 1 in this example. For the same period, the periodic algorithm involves lesser communication than the event-triggered algorithm and hence saves more energy.

Table 1
Factor of energy savings with periodic and event-triggered algorithms running on 48 PEs. The event-triggered algorithm is run with a threshold of 10^{-2}.

Period	Periodic	Event-Triggered
1	1	1
2	0.45	0.52
8	0.14	0.16
16	0.09	0.10
64	0.04	0.05

Now we provide an experimental verification of Proposition 1 in Section 5. Specifically, we look at the evolution of the residual (which is our convergence criterion of interest) over iterations for the event-triggered algorithm (having threshold δ and period r) and the periodic algorithm (with same period r) in Fig 4. The faster the residual decreases, the faster the numerical algorithm is considered to be converging to the desired tolerance, hence higher the rate of convergence. In Fig 4, we see that the residual for the algorithm with periodic communication decays slower than that of the event-triggered communication, establishing that the former is a lower bound on the rate of convergence of the latter.

We are also interested in the strong scaling of our algorithms. Strong scaling studies how the solution time decreases with increased number of PEs for a fixed problem size (Kumar (2002)). Strong scaling is said to be ideal if the solution time halves when the program is run on double the number of PEs. In other words, the ideal speedup of the program with double the number of PEs is exactly 2 times. We run strong scaling tests for our algorithms from 48 PEs to a significantly high value of 384 PEs. This requires us to choose a large grid resolution of 7680 in both dimensions for these tests. This is in contrast to the smaller grid resolution of 960 considered in Fig 2, Fig 3 and Fig 4 and also Table 1. The strong scaling speedup plots using grid resolution of 7680 are shown for the periodic algorithm in Fig 5 and for the event-triggered algorithm in Fig 6. It is seen that the strong scaling results for the event-triggered algorithm in Fig 6 are closer to the ideal speedup line than the periodic algorithm in Fig 5. This means that the event-triggered

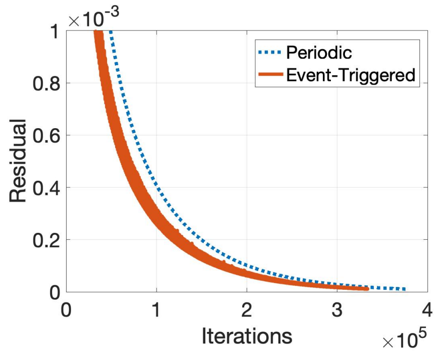

Fig. 4. Figure depicting the bound of rate of convergence. The red solid line corresponds to the residual of the event-triggered algorithm run with threshold $\delta = 10^{-2}$ and period $r = 8$ while the blue dotted line corresponds to the bound in terms of the periodic algorithm run with period $r = 8$. The y-axis of the plot has been truncated at 0.001 to show the region of interest. The red solid line for event-triggered algorithm has oscillations.

algorithm scales much better upto higher number of PEs than the periodic algorithm.

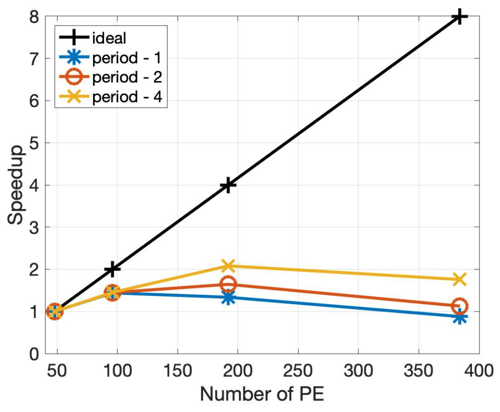

Fig. 5. Strong Scaling of the periodic algorithm with period $r = 1, 2, 4$. Since we run our simulations on multiples of 48 PEs from 48 to 384, the ideal scaling ranges from 1 to 8.

7 CONCLUSIONS

We present reduced communication schemes for parallel computing in this paper inspired by ideas from consensus literature. We showed that these methods of periodic and event-triggered communication have potential to reduce time and energy involved in parallel numerical solution of PDEs. We model the algorithms as discrete-time linear dynamical systems and provide proofs for asymptotic stability with insights on rate of convergence.

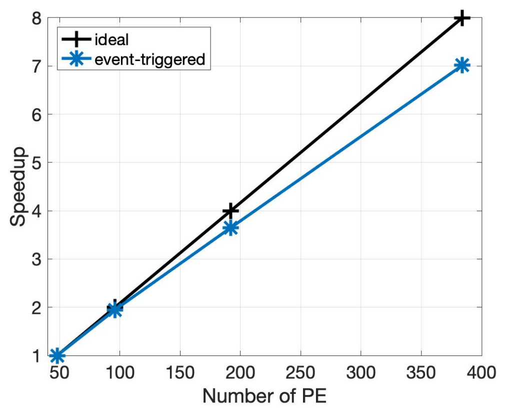

Fig. 6. Strong Scaling of event-triggered algorithm with threshold $\delta = 10^{-16}$ and period $r = 20$. Since we run our simulations on multiples of 48 PEs from 48 to 384, the ideal scaling ranges from 1 to 8.

References

Aditya, K. & Donzis, D. A. (2017), 'High-order asynchrony-tolerant finite difference schemes for partial differential equations', *Journal of Computational Physics* **350**, 550–572.

Bergman, K. et al. (2008), 'Exascale computing study: Technology challenges in achieving exascale systems', *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep* **15**.

Bertsekas, D. P. & Tsitsiklis, J. N. (1989), *Parallel and distributed computation: numerical methods*, Vol. 23, Prentice hall Englewood Cliffs, NJ.

Bittanti, S. & Colaneri, P. (2009), *Periodic systems: filtering and control*, Vol. 5108985, Springer Science & Business Media.

Demirel, B., Gupta, V., Quevedo, D. E. & Johansson, M. (2017), 'On the trade-off between communication and control cost in event-triggered dead-beat control', *IEEE Transactions on Automatic Control* **62**(6), 2973–2980.

Demmel, J., Hoemmen, M., Mohiyuddin, M. & Yelick, K. (2008), Avoiding communication in sparse matrix computations, *in* '2008 IEEE International Symposium on Parallel and Distributed Processing', IEEE, pp. 1–12.

Donzis, D. A. & Aditya, K. (2014), 'Asynchronous finite-difference schemes for partial differential equations', *Journal of Computational Physics* **274**, 370–392.

Frommer, A. & Szyld, D. B. (2000), 'On asynchronous iterations', *Journal of computational and applied mathematics* **123**(1-2), 201–216.

Ghosh, S., Lu, J., Gupta, V. & Tryggvason, G. (2018a), Fast parallel computation using periodic synchronization, *in* '2018 Annual American Control Conference (ACC)', IEEE, pp. 1659–1664.

Ghosh, S., Saha, K. K., Gupta, V. & Tryggvason, G. (2018b), Event-triggered communication in parallel computing, *in* '2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA)', IEEE, pp. 1–8.

Ghosh, S., Saha, K. K., Gupta, V. & Tryggvason, G. (2019), Parallel computation using event-triggered communication, *in* '2019 Annual American Control Conference (ACC)', IEEE. `http://sites.nd.edu/sghosh/files/2019/03/ACC_2019.pdf`.

Gropp, W. D., Gropp, W., Lusk, E. & Skjellum, A. (1999), *Using MPI: portable parallel programming with the message-passing interface*, Vol. 1, MIT press.

Gropp, W., Hoefler, T., Thakur, R. & Lusk, E. (2014), *Using advanced MPI: Modern features of the message-passing interface*, MIT Press.

Heemels, W. H., Donkers, M. & Teel, A. R. (2013), 'Periodic event-triggered control for linear systems', *IEEE Transactions on Automatic Control* **58**(4), 847–861.

Hoemmen, M. (2010), Communication-avoiding Krylov subspace methods, PhD thesis, UC Berkeley.

Kaszkurewicz, E. & Bhaya, A. (2012), *Matrix diagonal stability in systems and computation*, Springer Science & Business Media.

Kumar, V. (2002), *Introduction to parallel computing*, Addison-Wesley Longman Publishing Co., Inc.

Lee, K. & Bhattacharya, R. (2016), On the relaxed synchronization for massively parallel numerical algorithms, *in* '2016 American Control Conference (ACC)', IEEE, pp. 3334–3339.

Lee, K., Bhattacharya, R. & Gupta, V. (2015), A switched dynamical system framework for analysis of massively parallel asynchronous numerical algorithms, *in* '2015 American Control Conference (ACC)', IEEE, pp. 1095–1100.

Nowzari, C., Garcia, E. & Cortés, J. (2019), 'Event-triggered communication and control of networked systems for multi-agent consensus', *Automatica* **105**, 1–27.

Olfati-Saber, R., Fax, J. A. & Murray, R. M. (2007), 'Consensus and cooperation in networked multi-agent systems', *Proceedings of the IEEE* **95**(1), 215–233.

Strang, G. & Aarikka, K. (1986), *Introduction to applied mathematics*, Vol. 16, Wellesley-Cambridge Press Wellesley, MA.

Thomas, J. W. (2013), *Numerical partial differential equations: finite difference methods*, Vol. 22, Springer Science & Business Media.