Journal of Computational Physics 350 (2017) 550–572

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

High-order asynchrony-tolerant finite difference schemes for partial differential equations

Konduri Aditya *,[1], Diego A. Donzis

Department of Aerospace Engineering, Texas A&M University, College Station, TX 77843, United States

ARTICLE INFO

Article history:
Received 24 January 2017
Received in revised form 5 July 2017
Accepted 19 August 2017
Available online 1 September 2017

Keywords:
Asynchrony-tolerant schemes
Partial differential equations
Massive computations
Asynchronous computing

ABSTRACT

Synchronizations of processing elements (PEs) in massively parallel simulations, which arise due to communication or load imbalances between PEs, significantly affect the scalability of scientific applications. We have recently proposed a method based on finite-difference schemes to solve partial differential equations in an asynchronous fashion – synchronization between PEs is relaxed at a mathematical level. While standard schemes can maintain their stability in the presence of asynchrony, their accuracy is drastically affected. In this work, we present a general methodology to derive asynchrony-tolerant (AT) finite difference schemes of arbitrary order of accuracy, which can maintain their accuracy when synchronizations are relaxed. We show that there are several choices available in selecting a stencil to derive these schemes and discuss their effect on numerical and computational performance. We provide a simple classification of schemes based on the stencil and derive schemes that are representative of different classes. Their numerical error is rigorously analyzed within a statistical framework to obtain the overall accuracy of the solution. Results from numerical experiments are used to validate the performance of the schemes.

1. Introduction

Numerical simulations are an important tool in understanding complex problems in physics and engineering systems. Many of these phenomena are multi-scale in nature, and are governed by nonlinear partial differential equations (PDEs). With a wide range of scales at realistic conditions, like turbulence phenomena in fluid flows, the numerical solution of these equations becomes computationally very expensive. Advances in computing technology have made it possible to carry out intensive simulations on massively parallel computers. Currently, state-of-the-art simulations are routinely being done on tens or hundreds of thousands of processing elements (PEs) [1–4].

It is known, at extreme scale, that data communication as well as synchronization between PEs pose a major challenge in the scalability of scientific applications [5]. In the case of PDE solvers, where the parallelism is typically realized by decomposing the computational domain among PEs, communications that affect the scalability arise due to the computation of spatial derivatives in order to propagate the physical information across the domain. The problem becomes more acute in simulations of transient phenomena, where spatial derivatives are evaluated at each time step over an integration of large number of steps. Another issue concerning the scalability is related to the performance variations across the PEs in

* Corresponding author.
E-mail addresses: konduri.adi@gmail.com (K. Aditya), donzis@tamu.edu (D.A. Donzis).
[1] Current affiliation: Combustion Research Facility, Sandia National Laboratories, Livermore, CA 94550, United States.

http://dx.doi.org/10.1016/j.jcp.2017.08.037
0021-9991/

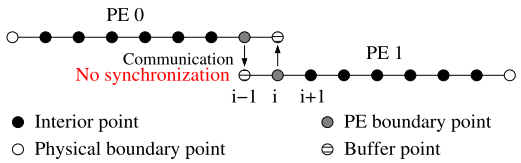

Fig. 1. Discretized one-dimensional domain decomposed into two PEs ($P = 2$).

a parallel system. In this case, sub-optimal performance of even a few PEs may lead to idling of others, as dictated by the data dependencies involved in the computations. It is likely that in future Exascale computing systems, which will have an extremely large PE count, communication and synchronization will be a major bottleneck. It is thus not surprising that there is a substantial increased interest in developing numerical methods that minimize communications and relax data synchronizations at the mathematical level [6,7].

An early effort in solving PDEs in an asynchronous fashion has been presented in [8,9]. Their method is based on finite-difference schemes and is restricted to the solution of parabolic PDEs with at most second order accuracy. More recent work [10,11], again based on finite-difference method, has suggested that due to the randomness in the arrival of messages at different PEs, the resulting algebraic difference equations are stochastic in nature. In that work, a statistical framework to analyze such systems was developed to study the numerical properties of commonly used schemes in the presence of asynchrony. Furthermore, they show that though the stability and consistency of the schemes can be maintained, their accuracy is significantly degraded. They also proposed the possibility of deriving schemes that are tolerant to communication data asynchrony. A follow up of this work to a simple specific equation and numerical scheme has be presented in [12]. Although the authors were able to maintain second order accuracy for their chosen scheme when asynchrony is present, one can show using Taylor series that they are severely limited to low order of accuracy. However, as mentioned earlier, a number of natural and engineering systems are multi-scale in nature and will require higher order accurate schemes. In this work, we present a general methodology to generate different classes of high-order asynchrony-tolerant (AT) schemes. This is the main objective of this paper.

The rest of the paper is organized as follows. We first briefly review the concept of asynchronous computing for PDEs in section 2. A general method to derive AT schemes, the choices in stencil available in arriving at these schemes and their classification are presented in section 3. In section 4, we show a statistical framework to analyze the overall accuracy of a numerical method when AT schemes are used. Numerical experiments to validate the performance of AT schemes are shown in section 5. Conclusions and further discussions are presented in section 6.

2. Concept

Let $u(x,t)$ be a function of spatial coordinate x and time t, which is governed by a time-dependent PDE in a one-dimensional domain. Fig. 1 illustrates the discretized domain which is decomposed into P number of PEs. Let i and n represent an arbitrary grid point in the domain and time level such that $u(x_i,t_n) = u_i^n$. For clarity in the exposition, we assume that the grid points are uniformly distributed in the domain with a spacing Δx. A finite-difference to approximate a spatial derivative at point i and time level n can be expressed, in the most general case, as

$$\left.\frac{\partial^d u}{\partial x^d}\right|_i^n = \sum_{j=-J_1}^{J_2} c_j u_{i+j}^n + \mathcal{O}(\Delta x^a), \tag{1}$$

where d is the order of the derivative, J_1 and J_2 are the number of points to the left and right of point i in the stencil, and c_j is the appropriate coefficient or weight of u_{i+j}^n such that the scheme is accurate to an order a in space. The term $\mathcal{O}(\Delta x^a)$ represents the truncation error of the scheme.

Usually, the numerical solution of a time-dependent PDE is obtained by advancing an initial condition according to an algebraic finite-difference equation in small steps of time Δt. During each time advancement, say, marching from a time level n to $n+1$, spatial derivatives are computed at each grid point using Eq. (1). In general, these computations are trivial to implement in a serial code, as the value of the function at all the grid points will be locally available in the memory of the PE. However, if the domain is decomposed into multiple PEs, computations at points near PE boundaries may need values of the function at stencil points that are computed in the neighboring PEs. Such values are commonly communicated into buffer or ghost points, as shown in Fig. 1. Note that the number of values communicated across the left and right PE boundaries is equal to J_1 and J_2, respectively.

Let I represent the set of physical grid points in the domain and B represent the set of buffer points. For convenience we divide the set I further such that $I = I_I \cup I_B$. The set of grid points near PE boundaries whose computations need data from neighboring PEs will be denoted by I_B. The complementary set of interior points, whose computations are independent of communication between PEs is denoted by I_I. In commonly used parallel algorithms, computations at a point $i \in I_B$