HIGHLY SCALABLE ASYNCHRONOUS COMPUTING METHOD FOR PARTIAL DIFFERENTIAL EQUATIONS: A PATH TOWARDS EXASCALE

A Dissertation

by

ADITYA KONDURI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2016

Major Subject: Aerospace Engineering

ABSTRACT

Many natural and engineering systems are governed by nonlinear partial differential equations (PDEs) which result in a multiscale phenomena, e.g. turbulent flows. Numerical simulations of these problems are computationally very expensive and demand for extreme levels of parallelism. At realistic conditions, simulations are being carried out on massively parallel computers with hundreds of thousands of processing elements (PEs). It has been observed that communication between PEs as well as their synchronization at these extreme scales take up a significant portion of the total simulation time and result in poor scalability of codes. This issue is likely to pose a bottleneck in scalability of codes on future Exascale systems. In this work, we propose an asynchronous computing algorithm based on widely used finite difference methods to solve PDEs in which synchronization between PEs due to communication is relaxed at a mathematical level. We show that while stability is conserved when schemes are used asynchronously, accuracy is greatly degraded. Since message arrivals at PEs are random processes, so is the behavior of the error. We propose a new statistical framework in which we show that average errors drop always to first-order regardless of the original scheme. We propose new asynchrony-tolerant schemes that maintain accuracy when synchronization is relaxed. The quality of the solution is shown to depend, not only on the physical phenomena and numerical schemes, but also on the characteristics of the computing machine. A novel algorithm using remote memory access communications has been developed to demonstrate excellent scalability of the method for large-scale computing. Finally, we present a path to extend this method in solving complex multi-scale problems on Exascale machines.

To my mother and father

Konduri Vijaya Lakshmi and Dr. Konduri Satya Prasad,

ACKNOWLEDGEMENTS

This work has been supported by the National Science Foundation (grant numbers OCI-1054966 and CCF-1349100, CCF-1439145). The computational resources of National Energy Research Scientific Computing Center (NERSC), NICS (University of Tennessee) and Extreme Science and Engineering Discovery Environment (XSEDE) have been used to perform our simulations.

First, I would like to express my thanks and gratitude to my advisor, Prof. Donzis, whose guidance and support has made this work possible. He has been a great inspiration and a *guru*. I have not just learnt the tools of trade, but also a rational approach towards problem solving with meticulous detail. I will always cherish our stimulating discussions which, typically, ended in exploring new ideas in research and teaching. I thank him for encouraging my ideas in several aspects of this work. I am also thankful to him for giving me plenty of opportunities to interact and collaborate with some of the eminent scholars.

I am thankful to Drs. Bowersox, Girimaji, Rauchwerger and Amato for being a part of my dissertation committee. I am especially thankful to Dr. Girimaji for teaching insightful courses on turbulence and occasional discussions on my work. I thank Drs. Bhattacharya and Hoefler for the collaborations in this work. I am also grateful to Dr. Hoefler for supporting my visit to ETH Zurich. I would also like to thank Dr. Narasimha, JNCASR, India for his encouragement and helpful discussions.

Many thanks to our department staff, Ms. Leatherman, Ms. Knabe and Ms.

Marianno for their warm support over the last four years.

A big thanks to my lab-mates and dear friends Agustin and Shriram who have always been there in support and made my doctoral journey a joyful experience. I thank my other lab-mates Chang-Hsin, Sualeh, Bryan and John for their help. My special thanks to Vidisha, Shyam and Sujan who have made my living in College Station memorable.

Finally, I thank my family - my parents, Dr. K. Satya Prasad and Vijaya Lakshmi, my wife, Haripriya, my sister and brother-in-law, Aruna and Sasidhar, and my little nephew, Vedanth, for their boundless love and emotional support. I am grateful to my parents who are my strength and who have always encouraged me in climbing my educational ladder.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

I.A. Overview

Many natural and engineered systems and processes can be accurately described by partial differential equations (PDEs). This includes fluid mechanics, electromagnetism, quantum mechanics as well as common simple processes in continuum media such as diffusion and wave propagation. In aerospace engineering, PDEs are widely used in understanding problems in aerodynamics, propulsion, structural design and control systems. Due to the complexity of the governing equations as well as the geometrical aspects of the problems, analytical solutions are not known in practical applications. A common example from aerodynamics and propulsion is the turbulence phenomena, which is governed by the celebrated Navier-Stokes equations. These equations are highly non-linear nature and result in a multi-scale phenomena, as depicted in Fig. I.1.

Advances in computing technology have made numerical simulations an indispensable research tool in understanding problems at great detail. At realistic conditions, the complexities involved in the above mentioned phenomena and systems typically require massive computational resources. In the last few decades, this computational power has been realized through increasing levels of parallelism. Current state-of-the-art simulations are being done routinely on hundreds of thousands of processing elements (PEs) with Petascale supercomputers (e.g. Jagannathan & Donzis,

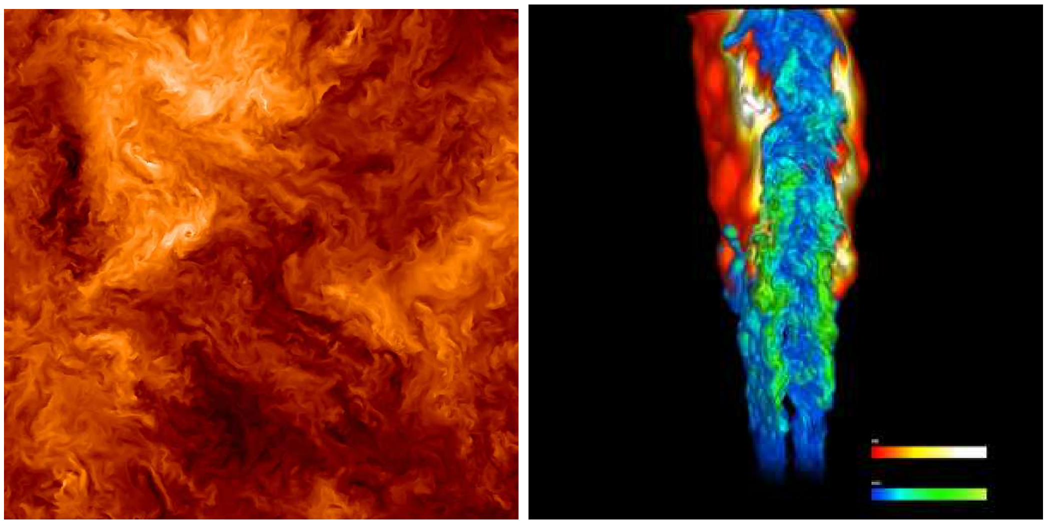


Figure I.1. Illustration of multi-scale nature of turbulence. Left: Velocity magnitude contour in an incompressible isotropic turbulent flow (Maqui & Donzis, 2011). Right: Iso-surfaces of chemical species in a turbulent jet flame (Yoo *et al.*, 2011).

2012; Lee *et al.*, 2013).

I.B. Literature review

In parallel simulations, when a problem is decomposed into a number of PEs, solving a PDE typically requires communication between PEs to compute spatial derivatives. As the number of PEs increases, this communication becomes more challenging (e.g. Jagannathan & Donzis, 2012; Lee *et al.*, 2013) and affects the scalability of codes. In fact, this may well be a major bottleneck at the next generation of computing systems (Dongarra *et al.*, 2011) which may comprise an extremely large number of PEs. At those extreme levels of parallelism, even small imbalances due to noise (Hoefler *et al.*, 2010) in otherwise perfectly balanced codes can represent enormous penalties as PEs idle waiting to receive data from other PEs. This is especially critical when, as commonly done, a global synchronization is imposed at each time step to finalize all communications as well as to obtain information to determine the time-step size in unsteady calculations subjected to a so-called Courant-Friedrichs-Lewy condition. Thus, in order to take advantage of computational systems at extreme levels of parallelism, relaxing all (especially global) synchronizations is of prime necessity (Dongarra *et al.*, 2011).

To understand the evolution of state-of-the-art turbulence simulations over the latest several decades, we plot graphs of the peak computational power and the highest Reynolds number in the simulations of incompressible homogeneous isotropic turbulence. It is interesting to note that the increasing Reynolds number has been realized by exploiting the parallelism and there has not been any change in the numerical method used in these simulations (Rogallo, 1981; Ishihara *et al.*, 2009; Donzis & Sreenivasan, 2010).

Currently, several efforts are in place to relax synchronizations at both hardware as well as software levels (Dongarra *et al.*, 2011). In terms of communications between PEs, messages can now be sent and received using asynchronous non-blocking algorithms. At a mathematical level substantial efforts have been devoted to asynchronous algorithms in different contexts such as linear systems of equations or more general fixed-point formulations Bertsekas & Tsitsiklis (1989); Frommer & Szyld (2000). Asynchronous linear solvers have been used in the context of PDEs to solve linear systems required in the computations but global synchronizations are typically still required. Due to the importance of increasing parallelism, other approaches have also been investigated such as explicit-implicit methods (e.g. Tavakoli & Davami,

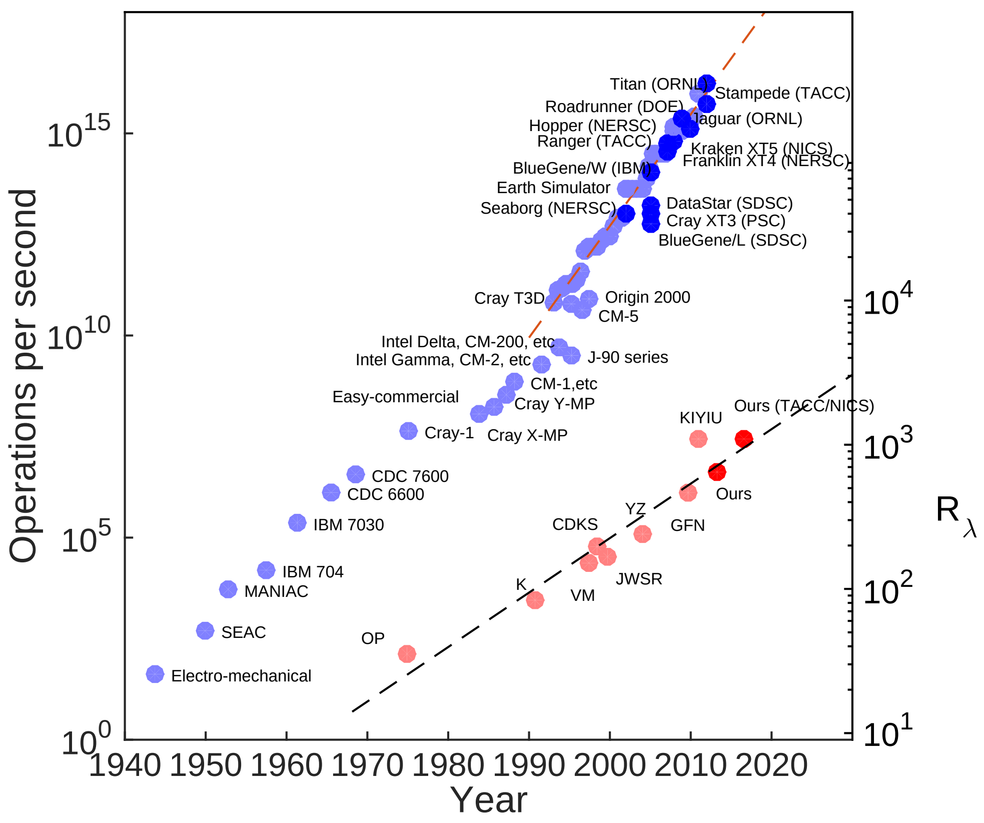


Figure I.2. Evolution of the computational power (left axis and blue circles) and the highest Taylor Reynolds number (right axis and blue circles)that has been acheived in the simulations of incompressible homogeneous isotropic turbulence.

2006) though, again, some type of synchronization at some point during the calculations is typically unavoidable or so-called discrete event-driven simulations (e.g. Karimabadi *et al.*, 2005). Work exploiting asynchrony to solve directly problems governed by time-dependent PDEs, however, has been more limited. For example, some studies that focused on PDEs (Amitai *et al.*, 1992, 1994) were limited to a particular class of PDE (heat equation) and the order of accuracy of the resulting schemes remained low. Further extensions to higher orders has also been limited (e.g. (Amitai *et al.*, 1996) for second order schemes).

I.C. Objective of the present work

The objectives of the current work are to:

1. investigate the effect of asynchrony on the numerical properties of standard finite difference schemes

2. devise a statistical framework to analyze numerical properties of schemes

3. present an asynchronous computing method based on finite difference schemes, which relaxes synchronization processes at a mathematical level, to solve PDEs

4. provide a novel methodology to derive asynchrony-tolerant schemes of arbitrary accuracy

5. design implementation algorithms and demonstrate the numerical and computational performance in practical applications

CHAPTER II

ASYNCHRONOUS COMPUTING*

II.A. Concept

Our interest is in the general linear PDE:

$$\frac{\partial u}{\partial t} = \sum_{d=1,D} \beta_d \frac{\partial^d u}{\partial x^d} \tag{2.1}$$

with D being the highest derivative in the PDE and the constants β_d's determine the characteristic of the different physical processes represented by the different terms. Particular cases of interest are the wave equation ($D = 1$ with $\beta_1 \neq 0$), the heat equation ($D = 2$ with $\beta_1 = 0$ and $\beta_2 \neq 0$), and the advection-diffusion equation ($D = 2$ with $\beta_1 \neq 0$ and $\beta_2 \neq 0$).

For illustration purposes consider the unsteady one-dimensional heat (or diffusion) equation

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \tag{2.2}$$

where $u(x,t)$ is the temperature at a spatial location $x \in [0,l]$ and time t and α is the thermal diffusivity of the medium. With N uniformly distributed grid points, Eq. (2.2) can be discretized using a second-order central difference in space and first-order forward difference in time to obtain a numerical scheme with well-know

*Parts of this chapter have been used from "Asynchronous finite-difference schemes for partial differential equations", Diego A. Donzis and Konduri Aditya, Volume 274, 1 October 2014, Pages 370-392, with permission from Elsevier under license number 3842150790850.

characteristics (Tannehill *et al.*, 1997):

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \mathcal{O}(\Delta t, \Delta x^2) \tag{2.3}$$

where u_i^n is the temperature at a point $x = x_i$ and time level n.

Here $x_i = i\Delta x$ with $\Delta x = l/N$ being the grid spacing and $i = 1, \ldots, N$. We will assume, unless explicitly mentioned, periodic boundary conditions. The time step size is Δt. The last term represents the order of the truncation error in time and space for this approximation.

The scheme in Eq. (2.3) can be rewritten in the following form

$$u_i^{n+1} = u_i^n + \frac{\alpha \Delta t}{\Delta x^2} \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n \right) \tag{2.4}$$

which shows that to advance the solution from time level n to $n+1$, one needs the value of the function at neighboring points at time level n. This is trivially implemented in a serial code where all the values u_i^n are available in the PE's memory (Fig. II.1(a)).

Consider, however, the case where the discretized domain is divided among a specified number of PEs (say 2, as in Fig. II.1(b)). Computations at interior points remain trivial as the required information is available locally to the PEs. Updating the values at grid points close to PE boundaries, however, require values from other PEs, that is, either u_{i-1}^n or u_{i+1}^n from the corresponding neighboring PEs. These values are typically communicated over the network into buffer (or "ghost") arrays. Computations are halted until all PEs receive data in these so-called halo exchanges.

For later use, we define set I which contains all the physical grid points. The buffer points in the grid belong to set B. The set of points I, where computations are carried out, is further divided into two subsets. I_I is the set of all interior grid points; that is, if $i \in I_I$ then computing derivatives at x_i does not require data from other PEs. If, on the other hand, computing the derivative at x_i does make use of data from other PEs, then i is a boundary point and we have $i \in I_B$. Obviously $I_I \cap I_B = \emptyset$. The number of elements in I_I and I_B will be denoted by N_I and N_B respectively.

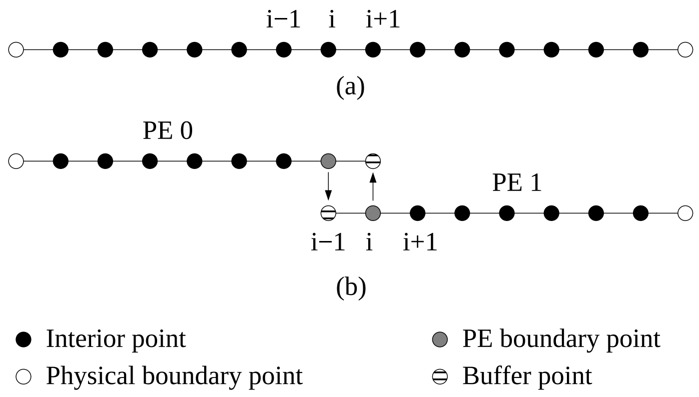


Figure II.1. Discretized one-dimensional domain. (a) Domain in serial codes. (b) Same domain decomposed into two PEs.

In applications, especially at extreme scales, the computing time may be much smaller than the communication time. This may result in a significant waiting time in large computing systems and have a profound effect in the scalability of the code. Furthermore, even if computation time is comparable to or larger than communication time, a global synchronization such as that described above, could cause substantial idling time if additional tasks (even if small) are required by one or a subset of PEs forces all PEs to wait for a single (the slowest) PE to finish its computations. This may also be an issue with respect to system noise at very large scales (Hoefler *et al.*, 2010).

These problems can be avoided by relaxing these synchronizations and allowing all PEs to continue calculations regardless of the status of the messages that are to be received by the corresponding PEs. In the context of Fig. II.1, PE 1 is not required to wait for the most updated value u^n_{i-1}, but instead, can compute derivatives using $u^{\tilde{n}}_{i-1}$ where $\tilde{n}$ is the latest time level available to PE 1 at x_{i-1}. This modifies the finite difference equation Eq. (2.4) for points close to PEs boundaries. In particular, the expressions for the leftmost and rightmost grid points in each PE are, respectively,

$$u_i^{n+1} = u_i^n + \frac{\alpha \Delta t}{\Delta x^2}\left(u^n_{i+1} - 2u^n_i + u^{\tilde{n}}_{i-1}\right) \tag{2.5}$$

$$u_i^{n+1} = u_i^n + \frac{\alpha \Delta t}{\Delta x^2}\left(u^{\tilde{n}}_{i+1} - 2u^n_i + u^{n}_{i-1}\right)\ . \tag{2.6}$$

In the most general case, $\tilde{n}$ could be n, $(n-1)$, $(n-2)$, etc., which, can further vary for different buffer points and time levels. The occurrence of a particular level for $\tilde{n}$ depends on how fast the communications take place, which in turn depends on a number of factors like hardware, network topology, network traffic, message size, etc. some of which may be unpredictable and turn the process into a random one.

Since $\tilde{n}$ will be essentially a random variable[†], we associate the occurrence of each time level with a probability. While in principle $\tilde{n}$ could take any value, it is convenient for the analysis (and necessary in terms of accuracy as we show below) to limit the number of past time levels the scheme could use. If the number of allowable time levels is L, then $\tilde{n} \in \{n, n-1, ..., n-L+1\}$. Let $p_{k[i]}$ be the probability of having $\tilde{n} = n - \tilde{k}_i$ at a grid point i, where the random delay $\tilde{k}_i$ can take the values

[†]In order to distinguish random from deterministic variables, we will use a tilde ($\tilde{}$) over the variable for the former.

Parameter	Symbol
Grid resolution	N
Number of PEs	P
Maximum number of time levels	L
Probability of $\tilde{n} = (n-k)$ at x_i	$p_{k[i]}$
Number of points in stencil	$2S+1$

Table II.1. Parameters used in the study of the properties of asynchronous schemes. N: number of grid points; P: number of processing elements; L: maximum level allowed for delays; $p_{k[i]}$: probability of observing a delay of k at grid point i; S: number of grid points on each side of a grid point used to compute a derivative. For a symmetric finite difference the number of grid points in the stencil is $2S+1$;

$\tilde{k}_i = 0, 1, ..., L-1$, then the probabilities at any grid point x_i are obviously related by

$$\sum_{k[i]=0}^{L-1} p_{k[i]} = 1. \tag{2.7}$$

The probability of having asynchronous computations at a buffer point i is thus $(1 - p_{0[i]})$.

Such asynchronous numerical schemes will be a viable option only if they are shown to be stable, consistent and accurate. Interestingly, the computed solution will not only depend on grid resolution and timestep, but also on the parameters that influence $\tilde{n}$. Table II.1 lists the parameters we will use to study the properties of finite differencing schemes under asynchronous conditions.

II.B. Stability

An obvious requirement for any scheme to be usable in practice is that it has to be stable. For deterministic schemes well-known techniques such as von Neumann analysis have been used extensively to prove stability (Hirsch, 1994). The main idea is to take advantage of the linear nature of the governing PDE in which case errors due to, for example, finite precision arithmetic, evolve according to the same PDE. By using a Fourier series to represent this error, one can easily determine under what conditions a particular mode (characterized by its wavenumber) will grow unbounded in time in which case the scheme is said to be unstable. If all the modes are either damped or conserve their amplitude in time, the scheme is said to be stable. The important parameter here is the amplification factor G which is defined as the ratio of the error between successive iterations. In the case of Eq. (2.2) discretized as in Eq. (2.4), it is readily shown (Hirsch, 1994) that the condition for stability is $r_\alpha \leq 1/2$ where $r_\alpha \equiv \alpha \Delta t / \Delta x^2$.

When asynchrony is allowed across PE boundaries, however, there are two difficulties that prevent us from utilizing von Neumann stability analysis. First, von Neumann analysis requires the discretized equation to be the same across the entire domain. This is not the case when $\tilde{k}_i \neq 0$ at some $i \in I_B$. Second, the specific value of $\tilde{k}_i$ (and therefore $\tilde{n}$) at those points is essentially a random variable which complicates further the applicability of von Neumann analysis.

The so-called matrix formulation, on the other hand, provides the flexibility needed to incorporate these complexities in a unified framework that allows us to compare stability characteristics with the original synchronous scheme. This is the approach we will follow in this work.

II.B.1. Synchronous schemes

Consider again Eq. (2.6) written in the following form

$$u_i^{n+1} = r_\alpha u_{i-1}^n + (1-2r_\alpha)u_i^n + r_\alpha u_{i+1}^{\tilde{n}}. \tag{2.8}$$

If the scheme is completely synchronous (i.e. $\tilde{n} = n$ always) and we assume a periodic domain, we can write

$$\boldsymbol{V}^{n+1} = \boldsymbol{A}^n \boldsymbol{V}^n. \tag{2.9}$$

where $\boldsymbol{V}^n = [u_1^n \; u_2^n \; \ldots \; u_i^n \; \ldots \; u_N^n]^T$ (superscript T stands for transpose) is a vector of size N and the $N \times N$ (cyclic tridiagonal) matrix $\boldsymbol{A}^n$ is given by

$$\boldsymbol{A}^n = \begin{bmatrix} (1-2r_\alpha) & r_\alpha & 0 & . & . & r_\alpha \\ r_\alpha & (1-2r_\alpha) & r_\alpha & 0 & . & . \\ 0 & r_\alpha & (1-2r_\alpha) & r_\alpha & 0 & . \\ . & . & \ddots & \ddots & \ddots & . \\ 0 & . & . & r_\alpha & (1-2r_\alpha) & r_\alpha \\ r_\alpha & 0 & . & . & r_\alpha & (1-2r_\alpha) \end{bmatrix} \tag{2.10}$$

Although the elements of this array are clearly independent of the value of n, the superscript in $\boldsymbol{A}^n$ is maintained for later analysis.

It is well known from Hirsch (1994) that the stability of such a scheme is determined by the spectral characteristics of the matrix $\boldsymbol{A}^n$ (in particular its largest eigenvalue). However, as will be clear momentarily a more general treatment is needed for our asynchronous schemes. We, thus proceed by noting that the evolution in time can be represented by

$$\boldsymbol{V}^{n+1} = \boldsymbol{A}^n\boldsymbol{V}^n = \boldsymbol{A}^n\boldsymbol{A}^{n-1}\boldsymbol{V}^{n-1} = \cdots = \boldsymbol{A}^n\boldsymbol{A}^{n-1}\ldots\boldsymbol{A}^0\boldsymbol{V}^0 \tag{2.11}$$

where $\boldsymbol{V}^0$ is the initial condition. In order for the scheme to be stable one requires a bounded solution with no amplification of perturbations. Due to the linear nature of the governing equation, the error evolves according to Eq. (2.11) as well and, thus, the system is stable when

$$||\boldsymbol{V}^{n+1}||/||\boldsymbol{V}^0|| \leq 1 \ . \tag{2.12}$$

where $||\cdot||$ is an appropriate norm. Using the property that $||\boldsymbol{AB}|| \leq ||\boldsymbol{A}||||\boldsymbol{B}||$ for any matrix norm and matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, it is easy to see that Eq. (2.12) is satisfied if

$$||\boldsymbol{A}^n||||\boldsymbol{A}^{n-1}||\ldots||\boldsymbol{A}^0|| \leq 1 \ . \tag{2.13}$$

Since in our case $\boldsymbol{A}^n$ is independent of n (i.e. $\boldsymbol{A}^n = \boldsymbol{A}^{n-1} = \cdots = \boldsymbol{A}^0$), it follows that the system is stable when the matrix in Eq. (2.10) satisfies $||\boldsymbol{A}^n|| \leq 1$. To track the largest perturbation, we use the ∞-norm which is defined, for any matrix $\boldsymbol{B}$ with elements b_{ij}, as $||\boldsymbol{B}||_\infty = \max_i \sum_j |b_{ij}|$. In our present case the condition

$$||\boldsymbol{A}^n||_\infty \leq 1 \tag{2.14}$$

for stability is readily shown to reduce to

$$|r_\alpha| + |1 - 2r_\alpha| + |r_\alpha| \leq 1. \tag{2.15}$$

Trivially, this is satisfied for $0 < r_\alpha \leq 1/2$, a well-known result Hirsch (1994) for Eq. (2.8).

II.B.2. Asynchronous schemes

Consider again Eq. (2.8) but now with $\tilde{n}$ being a random variable. For simplicity in the exposition, we will consider the case where $\tilde{n}$ can only take the values n or $n-1$. If we define the vector $\boldsymbol{W}^n$ as the concatenation of the values of u_i^n for all i's and the values of u_i^{n-1} also for all i's we can write

$$\boldsymbol{W}^{n+1} = \tilde{\boldsymbol{C}}^n \boldsymbol{W}^n \tag{2.16}$$

where $\boldsymbol{W}^n$ is of size $2N$ and given by

$$\boldsymbol{W}^{n+1} = \begin{bmatrix} \boldsymbol{V}^{n+1} \\ \boldsymbol{V}^n \end{bmatrix} \tag{2.17}$$

or explicitly,

$$\boldsymbol{W}^{n+1} = [u_1^{n+1} \quad u_2^{n+1} \quad \ldots u_i^{n+1} \ldots u_N^{n+1} |\ u_1^n \quad u_2^n \quad \ldots u_i^n \ldots u_N^n]^T \tag{2.18}$$

and the $2N \times 2N$ random matrix $\tilde{\boldsymbol{C}}^n$ can be divided into four $N \times N$ blocks as

$$\tilde{\boldsymbol{C}}^n = \left[\begin{array}{c|c} \tilde{\boldsymbol{A}}_0^n & \tilde{\boldsymbol{A}}_1^n \\ \hline \boldsymbol{I} & \boldsymbol{0} \end{array}\right] \tag{2.19}$$

with $\boldsymbol{I}$ and $\boldsymbol{0}$ being the identity and zero matrices respectively, both of size $N \times N$. The matrices $\tilde{\boldsymbol{A}}_0^n$ and $\tilde{\boldsymbol{A}}_1^n$ contain a random component due to $\tilde{n}$ which can be introduced as follows. In previous sections, we have defined $\tilde{k}_i$ to be the (random) delay seen by a PE which requires the value of the function at that point $i \in I_B$. This is related to $\tilde{n}$ as $\tilde{n} = n - \tilde{k}_i$. In our simplified case, $\tilde{k}_i$ can take the value of 0 or 1 and the probabilities associated with each of these outcomes is $p_{0[i]}$ and $p_{1[i]}$, respectively. Then we can write:

$$\tilde{\boldsymbol{A}}_0^n = \begin{bmatrix} (1-2r_\alpha) & (1-\tilde{k}_2)r_\alpha & 0 & . & . & r_\alpha \\ r_\alpha & (1-2r_\alpha) & (1-\tilde{k}_3)r_\alpha & 0 & . & . \\ 0 & r_\alpha & (1-2r_\alpha) & (1-\tilde{k}_4)r_\alpha & 0 & . \\ . & . & \ddots & \ddots & \ddots & . \\ 0 & . & . & r_\alpha & (1-2r_\alpha) & (1-\tilde{k}_N)r_\alpha \\ (1-\tilde{k}_1)r_\alpha & 0 & . & . & r_\alpha & (1-2r_\alpha) \end{bmatrix} \tag{2.20}$$

which is identical to Eq. (2.10) except for the factors $(1-\tilde{k}_i)$ in the elements above the diagonal. The matrix $\tilde{\boldsymbol{A}}_1^n$ is:

$$\tilde{\boldsymbol{A}}_1^n = \begin{bmatrix} 0 & \tilde{k}_2 r_\alpha & 0 & . & . & 0 \\ 0 & 0 & \tilde{k}_3 r_\alpha & 0 & . & . \\ 0 & 0 & 0 & \tilde{k}_4 r_\alpha & 0 & . \\ . & . & \ddots & \ddots & \ddots & \tilde{k}_N r_\alpha \\ \tilde{k}_1 r_\alpha & 0 & . & . & 0 & 0 \end{bmatrix}. \tag{2.21}$$

Note that the random elements appearing only above the diagonal is due to the use of Eq. (2.8) which has delays only on the right boundary for simplicity.

From the definitions above it is easy to see the effect of $\tilde{k}_i$ on the scheme. For a given i, if $\tilde{k}_i = 0$ then the element $(i-1, i)$ in $\tilde{\boldsymbol{A}}_1^n$ will be zero and the scheme will be governed by $\tilde{\boldsymbol{A}}_0^n$—that is a classical synchronous scheme with $\tilde{n} = n$. If on the other hand $\tilde{k}_i = 1$, then the element $(i-1, i)$ in $\tilde{\boldsymbol{A}}_0^n$ will be zero, and the corresponding element $(i-1, i)$ of $\tilde{\boldsymbol{A}}_1^n$ will be r_α—that is an asynchronous scheme with $\tilde{n} = n-1$. Thus, the random variable $\tilde{k}_i$ is responsible for switching between the two different schemes that result from $\tilde{n} = n$ and $\tilde{n} = n-1$.

We can now analyze the stability of the scheme described by Eq. (2.8). Just like Eq. (2.11) for synchronous schemes, we can write the evolution of the solution as

$$\boldsymbol{W}^{n+1} = \tilde{\boldsymbol{C}}^n \boldsymbol{W}^n = \tilde{\boldsymbol{C}}^n \tilde{\boldsymbol{C}}^{n-1} \boldsymbol{W}^{n-1} = \cdots = \tilde{\boldsymbol{C}}^n \tilde{\boldsymbol{C}}^{n-1} \ldots \tilde{\boldsymbol{C}}^0 \boldsymbol{W}^0. \tag{2.22}$$

The stability criterion

$$||\boldsymbol{W}^{n+1}||/||\boldsymbol{W}^0|| \leq 1 \tag{2.23}$$

is now satisfied if

$$|||\tilde{\boldsymbol{C}}^n \tilde{\boldsymbol{C}}^{n-1} \ldots \tilde{\boldsymbol{C}}^0|| \leq ||\tilde{\boldsymbol{C}}^n|| ||\tilde{\boldsymbol{C}}^{n-1}|| \ldots ||\tilde{\boldsymbol{C}}^0|| \leq 1 \ . \tag{2.24}$$

Obviously, this inequality holds if the norm of each individual matrix on the left-hand-size is less than unity.

Anticipating the use of the ∞-norm, we note that the $(i-1)$-th row of $\tilde{\boldsymbol{C}}^n$ is given by

$$\left[0 \quad \ldots \quad r_\alpha \quad (1-2r_\alpha) \quad (1-\tilde{k}_i)r_\alpha \quad 0 \quad \ldots \middle| \, 0 \quad \ldots \quad \tilde{k}_i r_\alpha \quad 0 \quad \ldots\right] \tag{2.25}$$

Thus, the absolute row sum to be considered to compute the ∞-norm from this row is

$$|r_\alpha| + |(1-2r_\alpha)| + |(1-\tilde{k}_i)r_\alpha| + |\tilde{k}_i r_\alpha| \tag{2.26}$$

Now it is readily seen that since $\tilde{k}_i$ is 0 or 1, then the contribution for the sum of the last two terms is always $|r_\alpha|$ independent of $\tilde{k}_i$. Furthermore, since all the rows in the upper half of $\tilde{\boldsymbol{C}}^n$ have identical structure (with the same absolute row sum) and the contribution from rows in the lower half is, trivially, $|1|$, we find that the scheme will be stable when

$$|r_\alpha| + |(1-2r_\alpha)| + |r_\alpha| \leq 1, \tag{2.27}$$

which is identical to Eq. (2.15). Stability for the asynchronous scheme is then also assured when $0 < r_\alpha \leq 1/2$.

We thus arrive at the conclusion that if the base synchronous scheme is stable (in the norm sense described in section II.B.1), the asynchronous equivalent will also be stable (in the same sense). The property that enables this is the invariance of the norm of $\tilde{\boldsymbol{C}}^n$ to random switching between numerical schemes.

While the results above were derived for the heat equation with a particular space and time discretization, the main conclusion can be generalized to a much broader class of problems and conditions. This is presented in Donzis & Aditya (2014).

II.C. Consistency and accuracy

II.C.1. General considerations

We now turn to the issue of the consistency and accuracy. A finite difference equation (FDE) at a point in the discretzed domain, is an approximation of a partial differential equation (PDE) with associated truncation error (E), i.e., PDE = FDE + E. Typically the truncation error is conveniently studied using a Taylor expansion of each term in the FDE about a grid point i and time level n.

Let us consider again Eq. (2.8) with $\tilde{n} = n$ (synchronous) and rewrite it by substituting the expansions $u_{i+1}^n = u_i^n + u'\Delta x + u''\Delta x^2/2! + u'''\Delta x^3/3! + \dots$, $u_{i-1}^n = u_i^n - u'\Delta x + u''\Delta x^2/2! - u'''\Delta x^3/3! + \dots$, and $u_i^{n+1} = u_i^n + \dot{u}\Delta t + \ddot{u}\Delta t^2/2! + \dddot{u}\Delta t^3/3! + \dots$, where space derivatives (denoted by primes) and time derivatives (denoted by a dot over the variable) are always evaluated at (i,n) and therefore the subscripts and superscripts are omitted for convenience in the notation. After rearrangement, it is easy to show that, to leading order, the truncation error is

$$E_i^n = -\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 + \mathcal{O}(\Delta t^2, \Delta x^4). \tag{2.28}$$

When $\Delta x \to 0$ and $\Delta t \to 0$, we can see that the truncation error has a limiting behavior $E_i^n \to 0$, showing that the FDE is consistent with the corresponding PDE at point i. The formal accuracy of a given scheme is defined as the power law exponent of the leading order terms in the truncation error. From Eq. (2.28), it is clear that the scheme Eq. (2.4) is first order in time and second order in space.

When asynchronous computations are allowed, $\tilde{n}$ is a random variable which can take values n, $n-1$, $n-2$, ... The truncation error at a particular point and time step, then, becomes also a random variable. If, for example, $\tilde{n} = n-1$ in Eq. (2.8), (i.e. $\tilde{k}_{i+1} = 1$), the truncation error is given by

$$\begin{aligned} \tilde{E}_i^n|_{\tilde{k}_{i+1}=1} &= -\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 - \alpha\dot{u}\frac{\Delta t}{\Delta x^2} + \alpha\dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha\dot{u}''}{2}\Delta t \\ &\quad + \mathcal{O}(\Delta x^3, \Delta t^2, \Delta x^p \Delta t^q) \end{aligned} \tag{2.29}$$

where we have extended the notation on the truncation error to indicate the location and value of the delay ($\tilde{k}_{i+1} = 1$). As indicated in Eq. (2.29), higher order terms involving the product of Δx and Δt are present due to data now being used at multiple time levels. These are here indicated by the variables p and q which are, in this case, bounded from below by -2 and 1, respectively.

The first two terms in Eq. (2.29) are identical to the leading order terms for the synchronous scheme (Eq. (2.28)). The additional terms are the result of the appearance of delayed values at $i+1$. It is interesting to observe that these contributions could actually *grow* as fast as $\Delta x^{-2} \sim N^2$ for fixed Δt as one refines the grid. Thus, consistency with the original governing equation requires careful considerations of how the time step size and grid spacing are reduced for more accurate solutions. We will expand on this issue momentarily.

For an arbitrary delay k at grid point $i+1$ (i.e. $\tilde{k}_{i+1} = k$) we can also find

$$\begin{aligned}\tilde{E}_i^n|_{\tilde{k}_{i+1}=k} &= -\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 - \alpha k \dot{u}\frac{\Delta t}{\Delta x^2} + \alpha k \dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha k \dot{u}''}{2}\Delta t \\ &\quad + \mathcal{O}(\Delta x^3, \Delta t^2, \Delta x^p \Delta t^q), \qquad (2.30)\end{aligned}$$

which obviously reduces to Eq. (2.28) for $k = 0$ (no delay). Since, as discussed in section II.B, stability is governed by the parameter $r_\alpha = \alpha \Delta t / \Delta x^2$, it is convenient to rewrite the above equation as

$$\begin{aligned}\tilde{E}_i^n|_{\tilde{k}_{i+1}=k} &= -\frac{r_\alpha \ddot{u}}{2\alpha}\Delta x^2 + \frac{\alpha u''''}{12}\Delta x^2 - r_\alpha k \dot{u} + r_\alpha k \dot{u}'\Delta x - \frac{r_\alpha k \dot{u}''}{2}\Delta x^2 \\ &\quad + \mathcal{O}(\Delta x^3, \Delta t^2, \Delta x^p \Delta t^q), \qquad (2.31)\end{aligned}$$

which shows the drastic effect of utilizing standard schemes in an asynchronous manner. Specifically, if r_α is kept constant in refining the discretization in time and space, then the third term on the right-hand-side represents a zeroth order contribution. That is, errors will not decrease under grid refinement. The scheme is thus not consistent with the original PDE.

This decrease in accuracy is not specific to the second-order space discretization used as an example. As discussed in more detail below, for any order of a (finite-difference) discretization of Eq. (2.1), asynchrony will result in truncation errors containing terms proportional to $\tilde{k}_i \Delta x^{-d} \Delta t$ where d goes from 1 to D in the most general case. If, because of a stability constraint, the time step size is determined according to a condition $\Delta t \sim \Delta x^p$ where $p < D$, then terms that grow with increasing resolution (i.e. terms proportional to $1/\Delta x^{D-p}$) will appear. If $p = D$, then zeroth-order terms will be present.

We note however, that the remarks above correspond to the truncation error at a point with a given value of $\tilde{n}$. However, delays are only expected at PE boundaries ($i \in I_B$) and not at interior points. Furthermore, even for grid points in I_B, delays are random and will depend on the particular realization of the simulation. A more general description of accuracy is thus needed to study these asynchronous schemes. In particular, accuracy will depend, in principle, on the original scheme, the number of PEs (number of boundaries where delays are expected), and the statistics of $\tilde{k}_i$ at each PE boundary which in turn will typically depend on architectural details of computing nodes, processors, network, etc. as well as the specific resource usage (e.g. by other users) at the time of the simulations. We thus now proceed to introduce some definitions in which all these elements can be taken into account.

II.C.2. Statistical description of truncation error

Consider a one-dimensional periodic domain with N grid points which is decomposed into P PEs leading also to P boundaries between processors. Our objective here is to quantify the error in order to obtain the order of accuracy. In doing so, we recognize that there are two elements that need to be considered in defining these concepts. First, the truncation error structure is not homogeneous in space. As discussed above for the heat equation, zeroth-order terms may appear at PE boundary points but the scheme is still second-order accurate in interior points. Second, even at boundary points the appearance of these terms is random. In particular, these additional terms due to asynchrony will appear with a probability $(1-p_{0[i+1]})$, where $p_{0[i+1]}$ is the probability of having no delay at point $i+1$, that is $\tilde{k}_{i+1}=0$. Thus, a probabilistic definition of the error seems appropriate in order to define convergence properties.

We first define two types of averages for a variable f: a space average and an ensemble average. In general, the space average could be taken over the entire domain or a subset of grid points (e.g. PE boundary points). If the average is over the entire domain the space average is $\langle f \rangle = \sum_{i=1,N} f_i/N$. If, instead, the average is taken over PE boundary points or interior points, the average is $\langle f \rangle_B = \sum_{i \in I_B} f_i/N_B$ or $\langle f \rangle_I = \sum_{i \in I_I} f_i/N_I$, respectively (the subscript in the angular brackets denotes the subset of grid points over which the average is taken). Ensemble averages, which take into account the stochastic nature of the delays, will be denoted by an overline $\overline{f}$.

Consider the (space and ensemble) average error over the entire domain at time step n:

$$\langle \overline{E} \rangle = \frac{1}{N} \sum_{i=1,N} \overline{E_i^n}. \tag{2.32}$$

Since the truncation error for $i \in I_B$ is random and for $i \in I_I$ is not, it is convenient to split the sum into interior and boundary points

$$\langle \overline{E} \rangle = \frac{1}{N} \left[\sum_{i \in I_I} E_i^n + \sum_{i \in I_B} \overline{\tilde{E}_i^n|_{\tilde{k}_{i+1}}} \right]. \tag{2.33}$$

where ensemble averages are only needed for the asynchronous regions at PE boundaries (second term on the right-hand-side). We now proceed to estimate the two different terms inside the brackets.

For interior points, there is no random component and the error at each i is simply given by Eq. (2.28). Thus, to leading order we can write

$$\sum_{i\in I_I} E_i^n \approx \sum_{i\in I_I} \left(-\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 \right), \tag{2.34}$$

or in terms of r_α

$$\begin{aligned}\sum_{i\in I_I} E_i^n &\approx \sum_{i\in I_I} \left(-\frac{\ddot{u} r_\alpha}{2\alpha} + \frac{\alpha u''''}{12} \right)\Delta x^2 \\ &\approx \Delta x^2 \sum_{i\in I_I} K_s \\ &\approx N_I \langle K_s \rangle_I \Delta x^2.\end{aligned} \tag{2.35}$$

For the last step we have used the fact that for the space average of $K_s \equiv -(\ddot{u} r_\alpha/2\alpha) + (\alpha u''''/12)$, which is defined here for convenience (subscript s stands for synchronous), the sum contains N_I terms corresponding to the number of interior grid points. For a central-difference scheme with a stencil size $2S+1$, the number of grid points in the domain where delays can be observed (if delays are expected only on one side of the stencil) is SP where P is, as before, the number of PEs which, for periodic boundary conditions, is equal to the number of PE boundaries. Thus the number of grid points in I_I is $N_I = (N - SP)$ which appears multiplying $\langle K \rangle_I$ in Eq. (2.35). The assumption of delays only on one side does not affect the generality of the results and is made only for clarity in the exposition. If delays are present on both sides, some expressions will have different prefactors but all conclusions regarding the accuracy of the scheme will still be valid.

For the second sum over I_B in Eq. (2.33), the truncation error is given by Eq. (2.30). Clearly when $\tilde{k}_{i+1} = 0$, which happens with probability $p_{0[i+1]}$, we recover a synchronous scheme and the second sum is similar to Eq. (2.35) with the difference that the prefactor is now the number of grid points at boundaries, which is $N_B = SP$. If, on the other hand $\tilde{k}_{i+1} > 0$, then the other terms in the truncation error appear. The ensemble average, to leading order, can be then estimated as

$$\begin{aligned}
\overline{\tilde{E}_i^n|_{\tilde{k}_{i+1}}} &\approx \sum_{k=0}^{L-1} p_{k[i+1]} \tilde{E}_i^n|_{\tilde{k}_{i+1}=k} \\
&\approx \sum_{k=0}^{L-1} p_{k[i+1]} \left(-\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 - \alpha k \dot{u}\frac{\Delta t}{\Delta x^2} + \alpha k \dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha k \dot{u}''}{2}\Delta t \right) \\
&\approx \left(-\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 \right) + \sum_{k=0}^{L-1} p_{k[i+1]} \left(-\alpha k \dot{u}\frac{\Delta t}{\Delta x^2} + \alpha k \dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha k \dot{u}''}{2}\Delta t \right) \\
&\approx \left(-\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 \right) + \left(-\alpha \dot{u}\frac{\Delta t}{\Delta x^2} + \alpha \dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha \dot{u}''}{2}\Delta t \right) \sum_{k=0}^{L-1} p_{k[i+1]} k
\end{aligned} \tag{2.36}$$

where we have used the fact that $\sum_{k=0,L} p_{k[i+1]} = 1$. Finally, we can write

$$\overline{\tilde{E}_i^n|_{\tilde{k}_{i+1}}} \approx \left(-\frac{\ddot{u}}{2}\Delta t + \frac{\alpha u''''}{12}\Delta x^2 \right) + \left(-\alpha \dot{u}\frac{\Delta t}{\Delta x^2} + \alpha \dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha k \dot{u}''}{2}\Delta t \right) \overline{\tilde{k}_{i+1}} \tag{2.37}$$

where $\overline{\tilde{k}_{i+1}}$ is the ensemble average of the delay given by $\overline{\tilde{k}_{i+1}} = \sum_{k=0,L} p_{k[i+1]} k$. The first and second parenthesis contain the synchronous and asynchronous terms, respectively. Clearly, for synchronous conditions we have $\tilde{k}_{i+1} = 0$ always, and thus $\overline{\tilde{k}_{i+1}} = 0$ making the last term in Eq. (2.37) equal to zero. It is also interesting to note that the average error increases linearly with the mean delay.

In terms of r_α we can rewrite Eq. (2.37) as

$$\overline{\tilde{E}_i^n|_{\tilde{k}_{i+1}}} \approx \left(-\frac{r_\alpha \ddot{u}}{2\alpha}\Delta x^2 + \frac{\alpha u''''}{12}\Delta x^2 \right) + \left(-r_\alpha \dot{u} + r_\alpha \dot{u}'\Delta x - \frac{r_\alpha \dot{u}''}{2}\Delta x^2 \right) \overline{\tilde{k}_{i+1}} \tag{2.38}$$

If we now take the sum over I_B following Eq. (2.35), we obtain

$$\begin{aligned}
\sum_{i \in I_B} \overline{\tilde{E}_i^n|_{\tilde{k}_{i+1}}} \approx \; & N_B \langle K_s \rangle_B \Delta x^2 + \Big(- r_\alpha N_B \langle \dot{u} \rangle_B + r_\alpha N_B \langle \dot{u}' \rangle_B \Delta x \\
& - N_B \frac{\langle \dot{u}'' \rangle_B}{2} \Delta x^2 \Big) \overline{\tilde{k}}
\end{aligned} \tag{2.39}$$

For simplicity, we have assumed that the statistics of the delays are homogeneous in space such that $\overline{\tilde{k}_{i+1}}$ is independent of i (i.e. the $\tilde{k}_{i+1}$ are i.i.d.) and thus the subscript $i+1$ has been dropped. If the statistical behavior of $\tilde{k}_{i+1}$ does depend on i, one can still take this into account in the above formulation by using the space average $\langle\overline{\tilde{k}}\rangle_B$ instead of simply $\overline{\tilde{k}}$.

Finally we substitute Eq. (2.39) and Eq. (2.35) into Eq. (2.33), and using the fact that $N_I\langle f\rangle_I + N_B\langle f\rangle_B = N\langle f\rangle$ for any quantity f, we obtain

$$\langle\overline{E}\rangle \approx \langle K_s\rangle\Delta x^2 + \frac{N_B}{N}\overline{\tilde{k}}\left(-r_\alpha\langle\dot{u}\rangle_B + r_\alpha\langle\dot{u}'\rangle_B\Delta x - r_\alpha\frac{\langle\dot{u}''\rangle_B}{2}\Delta x^2\right). \tag{2.40}$$

A number of interesting observations can be made from this result. First, for zero delays we recover second order convergence from the first term of Eq. (2.40). Second, we note that even a very small amount of asynchrony (which results in $\overline{\tilde{k}} > 0$) decreases the order of convergence significantly. In particular as Δx decreases to zero, the leading order term is the first one in parenthesis. Asymptotically, we then have

$$\begin{aligned}\langle\overline{E}\rangle &\approx -\frac{N_B}{N}\overline{\tilde{k}}r_\alpha\langle\dot{u}\rangle_B \\ &\approx -S\frac{P}{N}\overline{\tilde{k}}r_\alpha\langle\dot{u}\rangle_B,\end{aligned} \tag{2.41}$$

where we have used $N_B = SP$. For all other parameters constant, we can now write (using $\Delta x = l/N$)

$$\langle\overline{E}\rangle \sim \frac{P}{N} \sim P\Delta x, \tag{2.42}$$

showing that the original second-order scheme drops to first order (for the average error $\langle\overline{E}\rangle$) when asynchrony is present for a constant number of PEs, P. However, it is interesting to note the situation in which the number of PEs grow with the size of the problem N, which would correspond to so-called weak scaling when the parallel performance of a code is investigated. This is indeed, a common situation in applications where an increase in computational power available is typically used to increase computational work through finer grids to achieve more realistic conditions or include more physical content in the simulations. In such a case, P/N is constant and Eq. (2.42) shows that the error does not depend on Δx (that is a zeroth-order term) rendering the scheme inconsistent with the original equations. This highlights an interesting aspect of asynchronous schemes (and perhaps a number of numerical algorithms designed to run at so-called exascale (Dongarra *et al.*, 2011)), namely, the tight link between the numerical scheme and the computational system on which they are run. This is explored in more detail later on.

The decrease in formal accuracy we just explored, while asympotically true, might not be an issue in some practical simulations. From Eq. (2.40), it can be seen that if simulations are performed with a large number of grid points per PE (small N_B/N) and on a fast network (such that $\overline{\tilde{k}}$ is very small), then for realistic grids, the asynchronous contribution to the error (the entire second term on the right-hand-side) may be small and errors follow the synchronous second-order accuracy. The exact transition from the formal accuracy of the base scheme to zeroth or first order in asynchronous schemes cannot, in principle, be determined a priori. In Ch. V we show actual numerical experiments where these issues are explored.

II.C.3. Generalizations

The results in the previous section were derived for a particular discretization scheme of the heat equation. In this section we show how the conclusions found in that case indeed extend to a much wider set of situations.

First, we note that while we have used the average error $\langle\overline{E}\rangle$, the asymptotic result Eq. (2.41) also holds if one uses an L_1 norm to characterize the error (i.e. the space average of the absolute value of the error). This is clear if one considers that one of the terms in Eq. (2.38) (the zeroth-order term) is much larger than all the others for large N. After space averaging, we obtain

$$\langle\overline{|E|}\rangle \approx (N_B/N)\overline{|\tilde{k}|}r_\alpha\langle|\dot{u}|\rangle_B \sim P/N \sim P\Delta x. \tag{2.43}$$

This measure is typically more sensible in practice than the simple mean value since this eliminates the possibility of error cancellations during the averaging procedure.

The decrease of accuracy when common finite difference schemes are used asynchronously, is in fact very general. For example, if the second derivative on the right-hand-side of the heat equation is discretized using a longer stencil (higher order of accuracy), the truncation error pertaining to asynchronous terms retain the same form of the second parenthesis in Eq. (2.37), the only difference being the numerical coefficients for the different terms. For example, for a fourth-order central discretization of the second derivative $\partial^2 u/\partial x^2 = (-u_{i+2}^{\tilde{n}} + 16u_{i+1}^{\tilde{n}} - 30u_i^n + 16u_{i-1}^n - u_{i-2}^n)/12\Delta x^2 + \mathcal{O}(\Delta x^4)$, where both values at $i+1$ and $i+2$ are delayed, we obtain an analogous to Eq. (2.37)

$$\overline{\tilde{E}_i^n|_{\tilde{k}_{i+1}=\tilde{k}_{i+2}=k}} \approx \left(-\frac{\ddot{u}}{2}\Delta t - \frac{\alpha u''''}{90}\Delta x^4\right) + \left(-\frac{5}{4}\alpha k\dot{u}\frac{\Delta t}{\Delta x^2} - \frac{7}{6}\alpha k\dot{u}'\frac{\Delta t}{\Delta x} - \frac{\alpha k\dot{u}''}{2}\Delta t\right)\overline{\tilde{k}_{i+1}} \qquad (2.44)$$

with the first parenthesis being identical to the truncation error for the synchronous scheme, that is $\mathcal{O}(\Delta t, \Delta x^4)$. Again, asymptotically the first term in the second parenthesis will dominate (if one uses a constant r_α for stability) leading to Eq. (2.41) with a prefactor 5/4.

The generality of this results can be easily seen if one of the terms in the spatial derivative of a consistent scheme is delayed. In that case, the Taylor expansion for that particular term will contain the standard spatial error related to the derivative and terms in Δt due to the necessary expansion in time. More formally, consider an approximation to the d-th derivative at location i as

$$\frac{\partial^d u}{\partial x^d} \approx \sum_{j=-S}^{S} \frac{b_j u_{i+j}^n}{\Delta x^d}. \qquad (2.45)$$

Assume that the value of the function experiences a delay of k at $i+1$, that is $\tilde{k}_{i+1} = k$. Then,

$$\frac{\partial^d u}{\partial x^d} \approx \frac{\cdots + b_1\ u_{i+1}^{n-k} + \ldots}{\Delta x^d}, \qquad (2.46)$$

where b_1 is the appropriate coefficient at $i+1$ for the given discretization. The Taylor expansion of the only term explicitly written in the numerator is $u_{i+1}^{n-k} = u_i^n + [u'\Delta x + u''\Delta x^2/2! + u'''\Delta x^3/3! + \ldots] + [-k\dot{u}\Delta t + \ddot{u}k^2\Delta t^2/2! - k^3\dddot{u}/3! + \ldots] + [-k\dot{u}'\Delta x\Delta t - k\dot{u}''\Delta x^2\Delta t/2 + \ldots]$. The first bracket is identical to the expansion for synchronous schemes. The last bracket contains some cross terms involving both Δt

and Δx. It is the second bracket, however, that contains the more problematic contributions. It comprises terms in powers of Δt starting with $k\dot{u}\Delta t$. When introduced into Eq. (2.46), the result is the term

$$C_d k\dot{u}\frac{\Delta t}{\Delta x^d}, \tag{2.47}$$

where C_d is a combination of numerical prefactors determined by the original PDE, and the discretization scheme used. In general, the leading order terms in the local truncation error (i.e. at a particular point) of the discretized general equation Eq. (2.1) due to asynchrony will be

$$\sum_{d=1}^{D} C_d \beta_d k\dot{u}\frac{\Delta t}{\Delta x^d}. \tag{2.48}$$

Note that this local truncation error will appear only for the subset of grid points $i \in I_B$.

When Δt is chosen as a power law in Δx (due to, e.g., stability constraints) clearly the highest derivative in the original PDE will be the leading order term in the truncation error. In the case of the advection-diffusion equation, for example, the original PDE contains first and second order derivatives, and then the two terms $\Delta t/\Delta x$ and $\Delta t/\Delta x^2$ will be present (actually the former will be present even if the original equation contained only a second derivative because, as shown under Eq. (2.46), higher-order terms also include $\Delta x \Delta t$ which when divided by Δx^2 yields a term $\Delta t/\Delta x$).

A few important conclusions can be made about the truncation error induced by asynchrony. First, if the space resolution is fixed (i.e. fixed Δx), the *local* truncation error will be first order in time regardless of the order of accuracy of the time integration. Second, if a stability criterion requires a condition on $\beta_D \Delta t/\Delta x^D$, then the local truncation error will always have zeroth-order terms which will not vanish as resolution increases. As clear from our analysis leading to Eq. (2.43), however, if P is constant, the space average over I_B leads to an additional factor Δx. Thus, the *average* truncation error will be first order.

The stability criterion based on $\beta_D \Delta t/\Delta x^D$ can be analytically shown for simple model equations: for the wave equation one needs $r_c = c\Delta t/\Delta x \leq 1$ and for the heat equation $r_\alpha = \alpha \Delta t/\Delta x^2 \leq 1/2$. For more complex equations such as the advection-difussion equation which contain both convective as well as diffusive terms, if the time step is selected according to the diffusive condition $\Delta t = r_\alpha \Delta x^2/\alpha$ then the local truncation error will contain terms of order $r_\alpha \Delta x$ and r_α (i.e. first and zeroth order, respectively). If on the other hand the time step is chosen according to a convective condition $\Delta t = r_c \Delta x/c$ then the truncation error will contain terms of the form r_c and $r_c/\Delta x$. The magnitude of the last term actually increases under grid refinement; when averaged in space, though, the truncation error due to asynchrony becomes zeroth order. However, the stability for this equation requires Δt to be controlled by r_α asymptotically and thus the average truncation error is asymptotically first order.

Due to the number of possible combinations of paramaters, we include a summary of the results in this section in Table II.2 for the general equation Eq. (2.1) with an arbitrary finite difference discretization in space and a two-level discretization in time. The second and third columns indicate the leading order term in the local (i.e. at a grid point) and the averge truncation error, respectively for the conditions listed in the first column. The first two rows correspond to the case of a fixed number of

PEs; the third and fourth correspond to the case of P/N constant (i.e. weak scaling); the last two rows correspond to a more general setting with $\Delta t \sim \Delta x^q$ (to be discussed below).

Parameters held constant	local E_i^n	$\langle \overline{E} \rangle$ or $\langle \overline{\lvert E \rvert} \rangle$
P, r_D	1	Δx
P, Δt	$1/\Delta x^D$	$1/\Delta x^{D-1}$
P/N, r_D	1	1
P/N, Δt	$1/\Delta x^D$	$1/\Delta x^D$
P, $\Delta t/\Delta x^q$	Δx^{q-D}	Δx^{q-D+1}
P/N, $\Delta t/\Delta x^q$	Δx^{q-D}	Δx^{q-D}

Table II.2. Summary of leading terms in the local (for $i \in I_B$), and average truncation error due to asynchronicity for different scenarios. First column indicates the conditions under which grid refinement (i.e. $\Delta x \to 0$) is conducted. The variable $r_D = \beta_D \Delta t/\Delta x^D$ corresponds to the resulting parameter from the highest derivative in the problem (for the heat, wave and advection-diffusion equations it will correspond to r_c, r_α and r_α respectively.)

We conclude this section by noting that an important finding of our analysis, is that the numerical characteristics of schemes used in an asynchronous fashion to exploit extreme levels of parallelism, are intrinsically linked to architectural details of the computational system as well as the manner in which the problem is scaled up. For example, for fixed number of PEs, we have shown the scheme is first-order accurate on the average error. However, if both problem size as well as PE count grow in proportion to each other, the scheme is inconsistent. This, can clearly be alleviated if Δt is chosen in a different way than what a stability condition would suggest. For example, for the heat equation with a first-order forward difference in time and a second-order central difference in space, choosing $\Delta t \propto \Delta x^3$, would ensure a second-order formal order of convergence for the average error. If a general relation $\Delta t \sim \Delta x^q$ is used, then the resulting schemes will have the convergence properties described in the last two rows of Table II.2. Obviously, for realistic large-scale problems, a value of q beyond that required for stability reasons, may likely lead to prohibitively small time steps which may render the entire simulation unfeasible.

II.D. Summary

In this chapter, we have presented the concept of asynchronous computing, based on finite differences, to solve PDEs on parallel machines. We have investigated numerical properties of standard schemes using a novel statistical framework. We found that, though, schemes continue to remain stable in the presence of asynchrony, their accuracy is significantly affected. We have characterized the average error in terms of physical, numerical and computational parameters. Theoretical predictions show that the average error is dominated by term due to asynchrony, which is first and zeroth order accurate under strong and weak scaling of simulations, respectively.

CHAPTER III

ASYNCHRONY-TOLERANT SCHEMES

III.A. 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. III.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 $u(x_i,t_n) = u_i^n$. In this chapter, 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

$$\left.\frac{\partial^d u}{\partial x^d}\right|_i^n = \sum_{j=-J_1}^{J_2} c_j u_{i+j}^n + \mathcal{O}(\Delta x^a), \tag{3.1}$$

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. 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.

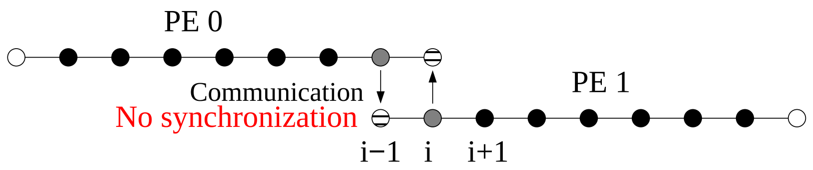


Figure III.1. Discretized one-dimensional domain in decomposed into two PEs ($P = 2$).

Usually, the numerical solution of a transient 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. (3.1). 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 in multiple PEs, computations at points near PE boundaries may need values of the function at stencil points that are computed in the neighboring PEs. Usually, such values are communicated into buffer or ghost points, as shown in Fig. III.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. We divide the set I further into two more. The set of grid points near PE boundaries, denoted by I_B, whose computations need data from the neighboring PEs. And, the complementary set of interior points I_I, whose computations are independent of communication between PEs. In commonly used parallel algorithms, computations at a point $i \in I_B$ cannot be advanced until the communication between PEs is complete. This is typically ensured by enforcing communication synchronization after messages are issued from one PE to another. As mentioned earlier, with a large number of PEs such synchronizations become expensive and result in poor scalability of codes at extreme scales. We refer to this as *synchronous* computing.

In the case of *asynchronous* computing, communication between PEs is initiated at each time step, however, the data synchronization is not enforced. This means, we cannot ensure that the time level of the function at buffer points is n. It can be n, $n-1$, $n-2$, ... depending on the status of messages from successive time advancements. Due to the random nature of the arrival of messages at different PEs Hoefler *et al.* (2008), the availability of a particular time level at a buffer point is also random. Let $\tilde{n} = n - \tilde{k}_j$ be the latest available time level at a buffer point j, where $\tilde{k}_j$ is the corresponding random delay at that point[*]. Note that $\tilde{n}$ can be different at different locations and time steps. If we restrict the maximum allowable delay levels to L, then $\tilde{n} \in \{n, n-1, ..., n-L+1\}$ and $\tilde{k}_j \in \{0, 1, ..., L-1\}$. The scheme in Eq. (3.1), when asynchrony is allowed, can be rewritten as

$$\left.\frac{\partial^d u}{\partial x^d}\right|_i^n \approx \sum_{j=-J_1}^{J_2} c_j u_{i+j}^{n-l}, \tag{3.2}$$

where $l = 0$ for $i+j \in I$ and $l = \tilde{k}_{i+j}$ for $i+j \in B$. Unlike the scheme in Eq. (2.45) which contains a single time level, this scheme uses multiple time levels when some of the points in the stencil belong to the set B. It has been shown in Donzis & Aditya (2014) that the accuracy of commonly used finite-differences in such an asynchronous fashion significantly affects the accuracy. In particular, accuracy drops to first order regardless of the original finite difference used. Thus, we proceed to derive asynchrony-tolerant schemes that maintain accuracy even when there is a communication delay.

[*]In order to distinguish random from deterministic variables, we will use a tilde ($\tilde{}$) over the variable for the former.

III.B. Asynchrony-tolerant schemes

III.B.1. General methodology

Taylor series and the method of undetermined coefficients provide a systematic procedure to derive finite-difference schemes. As we show momentarily, this approach can also be used to construct asynchrony-tolerant schemes to approximate spatial derivatives. Let u_{i+j}^{n-l} represent the function at a generic point $i+j$ in the stencil with an arbitrary delay of l levels to compute a spatial derivative at a point i and time level n. Using the L possible time delays, we can express an asynchrony-tolerant scheme as

$$\left.\frac{\partial^d u}{\partial x^d}\right|_i^n \approx \sum_{j=-J_1}^{J_2} \sum_{l=0}^{L-1} \tilde{c}_j^l u_{i+j}^{n-l}, \tag{3.3}$$

where $\tilde{c}_j^l$, for the range of j and l, are the appropriate coefficients that have to be determined. Note that this scheme represents the most general case with the function at all possible time levels at each point in the stencil. However, depending on the delay at each grid point and time step, which is given by $\tilde{k}_{i+j}$, only one or few time levels may be used in approximating the derivative.

The random nature of $\tilde{k}_{i+j}$ is, now, embedded into $\tilde{c}_j^l$. The merits of using older time levels not just at buffer points, but also at interior points will be shown later.

The coefficients in the scheme expressed in Eq. (3.3) can be obtained by imposing constraints on different terms of the Taylor series, upon expansion of the function at each combination of point and time level in the stencil. Let us consider the Taylor series of u_{i+j}^{n-l} about the point i and time level n. The series is an expansion in two variables, namely Δx and Δt, which is given by

$$u_{i+j}^{n-l} = \sum_{\eta=0}^{\infty}\sum_{\zeta=0}^{\infty} u^{(\eta,\zeta)} \frac{(j\Delta x)^{\eta}(-l\Delta t)^{\zeta}}{\eta!\zeta!}, \tag{3.4}$$

where $u^{(\eta,\zeta)}$ denotes the ηth and ζth partial derivative in space and time of u evaluated at i and n. When $l = 0$, the function corresponds to a synchronous value of u. This makes the terms in the series a function of Δx only:

$$u_{i+j}^{n} = \sum_{\eta=0}^{\infty} u^{(\eta,0)} \frac{(j\Delta x)^{\eta}}{\eta!} \tag{3.5}$$

To obtain the constraints that will assure a given order of accuracy, we substitute the Taylor series of u in the right hand side of Eq. (3.3).

$$\begin{aligned}
\sum_{j=-J_1}^{J_2}\sum_{l=0}^{L-1} \tilde{c}_j^l u_{i+j}^{n-l} &= \sum_{j=-J_1}^{J_2}\sum_{l=0}^{L-1} \tilde{c}_j^l \sum_{\eta=0}^{\infty}\sum_{\zeta=0}^{\infty} u^{(\eta,\zeta)} \frac{(j\Delta x)^{\eta}(-l\Delta t)^{\zeta}}{\eta!\zeta!} \\
&= u^{(0,0)} \sum_{j=-J_1}^{J_2}\sum_{l=0}^{L-1} \tilde{c}_j^l + u^{(1,0)}\Delta x \sum_{j=-J_1}^{J_2}\sum_{l=0}^{L-1} j\tilde{c}_j^l - \\
&\quad u^{(0,1)}\Delta t \sum_{j=-J_1}^{J_2}\sum_{l=0}^{L-1} l\tilde{c}_j^l - u^{(1,1)}\Delta x\Delta t \sum_{j=-J_1}^{J_2}\sum_{l=0}^{L-1} \tilde{c}_j^l + \\
&\quad \frac{u^{(2,0)}}{2}\Delta x^2 \sum_{j=-J_1}^{J_2}\sum_{l=0}^{L-1} j^2\tilde{c}_j^l + \frac{u^{(0,2)}}{2}\Delta t^2 \sum_{j=-J_1}^{J_2}\sum_{l=0}^{L-1} l^2\tilde{c}_j^l + \ldots
\end{aligned} \tag{3.6}$$

The linear combination of the function values, in the above equation, represents a scheme when (i) the coefficient of the d-th derivative of u in space is unity, and (ii) high order terms are eliminated according to the desired accuracy of the scheme.

Let a be the desired order of accuracy in space. This means that the leading order term in the truncation error should vary with the grid spacing as Δx^a. In the Taylor series of the function for synchronous schemes, as in Eq. (3.5), the higher order terms can be readily identified as the ones with the power of Δx less than $d+a$.

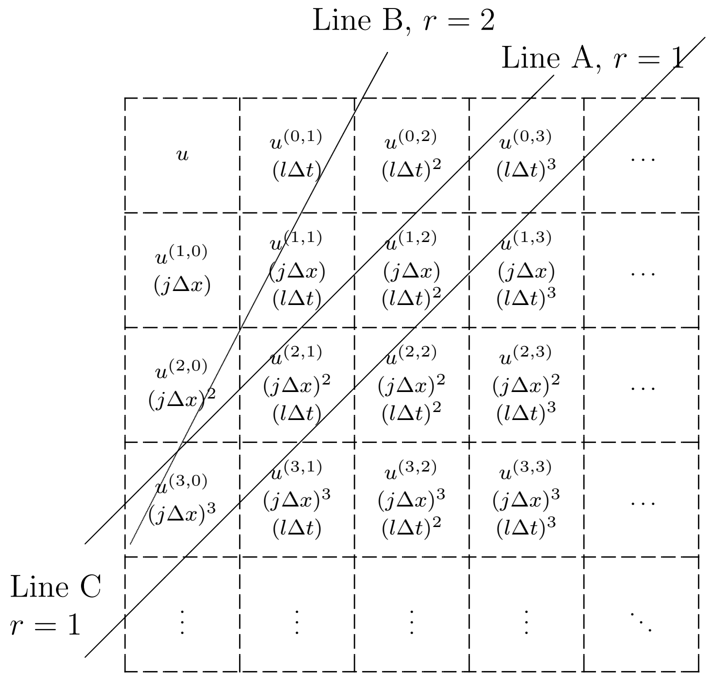
However, when asynchrony is present this is not obvious. The terms in the series can now be a function of either or both Δx and Δt, which are usually not independent. In order to identify higher order terms, let us assume the relation $\Delta t \sim \Delta x^r$. Such a relation is often obtained from analysis of the scheme's numerical stability or other constraints posed by the physics of the problem. Using this relation, we can arrive at the condition to identify higher order terms that need to be eliminated to obtain a scheme of order a. This expression is: $\eta + r\zeta < d + a$. Using Eq. (3.6), we can then summarize the constraints as

$$\sum_{j=-J_1}^{J_2} \sum_{l=0}^{L-1} \tilde{c}_j^l \frac{(j\Delta x)^\eta (-l\Delta t)^\zeta}{\eta!\zeta!} = \begin{cases} 1 & \text{for } (\eta,\zeta) = (d,0) \\ 0 & \text{for } \eta + r\zeta < d + a; (\eta,\zeta) \neq (d,0). \end{cases} \tag{3.7}$$

Clearly, the first condition in the above equation makes the coefficient of the d-th derivative term on the right hand side of Eq. (3.6) unity. The second condition will set to zero all the necessary higher order terms to obtain an overall accuracy a. For a given stencil, these conditions give rise to a system of linear equations. The number of equations in the system is one more than the number of higher order terms that have to be eliminated from Eq. (3.6). Let $\boldsymbol{A}\tilde{\boldsymbol{c}} = \boldsymbol{b}$ represent this system, where $\boldsymbol{A}$ is the coefficient matrix whose elements are a function of j and l, $\tilde{\boldsymbol{c}}$ is the vector of variables that contains coefficients in the scheme and $\boldsymbol{b}$ is the vector with zero elements except for the row corresponding to the order of the derivative to be approximated. The solution to this system determines the coefficients of the scheme.

Before getting into the discussion on the choice of stencil, we make a few observations regarding the linear system when asynchrony is present. To aid the discussion we express the terms in the Taylor series of the function at the generic stencil point,

u_{i+j}^{n-l}, in a matrix format as shown in Fig. III.2. This provides a simple format to visualize different terms in the series and help us easily identify the terms on which the conditions in Eq. (3.7) have to be imposed. In this graphical representation, we



Line B, $r = 2$
Line A, $r = 1$
Line C $r = 1$
u
$u^{(0,1)}$ $(l\Delta t)$
$u^{(0,2)}$ $(l\Delta t)^2$
$u^{(0,3)}$ $(l\Delta t)^3$
$u^{(1,0)}$ $(j\Delta x)$
$u^{(1,1)}$ $(j\Delta x)$ $(l\Delta t)$
$u^{(1,2)}$ $(j\Delta x)$ $(l\Delta t)^2$
$u^{(1,3)}$ $(j\Delta x)$ $(l\Delta t)^3$
$u^{(2,0)}$ $(j\Delta x)^2$
$u^{(2,1)}$ $(j\Delta x)^2$ $(l\Delta t)$
$u^{(2,2)}$ $(j\Delta x)^2$ $(l\Delta t)^2$
$u^{(2,3)}$ $(j\Delta x)^2$ $(l\Delta t)^3$
$u^{(3,0)}$ $(j\Delta x)^3$
$u^{(3,1)}$ $(j\Delta x)^3$ $(l\Delta t)$
$u^{(3,2)}$ $(j\Delta x)^3$ $(l\Delta t)^2$
$u^{(3,3)}$ $(j\Delta x)^3$ $(l\Delta t)^3$

Figure III.2. Terms in the Taylor series of u_{i+j}^{n-l} illustrated in a matrix format. Constant in each term are omitted for clarity. Lines A, B and C represent $\eta + r\zeta = d + a$ for different sets of parameters.

omit constants in each term for the sake of clarity.

In words, Eq. (3.7) implies constraints on the term containing the derivative of order $(d, 0)$ and on all the terms that satisfy the inequality $\eta + r\zeta < d + a$, that is, all terms above the $\eta + r\zeta = d + a$ line in Fig. III.2. With this representation, we can easily separate terms that need to be eliminated from those that do not. To illustrate this, let us choose $d = 1$, $a = 2$ and $r = 1$, which corresponds to a second-order approximation of the first derivative, using a convective-type CFL condition such that $\Delta t \sim \Delta x$. For these parameters, conditions are imposed on the terms with $(\eta, \zeta) = \{(0,0), (1,0), (2,0), (0,1), (1,1), (0,2))\}$, which are the terms above the line A in the figure. If asynchrony is absent, that is, $l = 0$, the only terms that are non-zero in the table belong to the first column. This show that, for a given accuracy, the number of terms on which conditions are imposed is larger when asynchrony is present, which thus results in a larger linear system.

The increase in the number of equations also depends on r, which relates Δt and Δx. For example, the situation for $r = 2$ is also shown in Fig. III.2 with line B. The number of terms above the line B is less than A, which implies that a higher r will reduce the number of higher order terms due to asynchrony for a given accuracy.

The other aspect is the increase in stencil size with increase in accuracy. In commonly used synchronous schemes, a successive increase in the order of accuracy will impose condition on one more term in the Taylor series, corresponding to line C in Fig. III.2 with $l = 0$. This adds an additional equation to the linear system that can be solved by adding one more grid point to the stencil. However, in deriving asynchrony-tolerant schemes more than one additional equation may be added to the system, as exemplified above. Thus, we expect the stencil of asynchrony-tolerant schemes to grow larger than commonly used synchronous schemes when the accuracy is increased.

III.B.2. Choice of stencil

In principle one can choose a stencil that consists of different points and time levels to approximate spatial derivatives. However, the stencil of commonly used synchronous schemes are constructed exclusively with spatial grid points. This is because of two reasons.First, the function at the synchronous time level is available for spatial derivative evaluation at all points in the domain. Second, as argued in Donzis & Aditya (2014), and elaborated in section III.B.1 above, this choice avoids the additional terms that will appear in Taylor series when the stencil consists of delayed time levels. When asynchrony is present, on the other hand, as is clear from Eq. (3.3), the function can belong to multiple time levels. Thus, we can take advantage of using the function at delayed time levels in deriving asynchrony-tolerant schemes. To understand this let us recall the tabular representation of the Taylor series of u_{i+j}^{n-l}, as shown in Fig. III.3. We can classify terms into four groups, as

u	$u^{(0,1)}$ $(l\Delta t)$	$u^{(0,2)}$ $(l\Delta t)^2$	$u^{(0,3)}$ $(l\Delta t)^3$	$\cdots$
$u^{(1,0)}$ $(j\Delta x)$	$u^{(1,1)}$ $(j\Delta x)$ $(l\Delta t)$	$u^{(1,2)}$ $(j\Delta x)$ $(l\Delta t)^2$	$u^{(1,3)}$ $(j\Delta x)$ $(l\Delta t)^3$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
$u^{(d,0)}$ $(j\Delta x)^d$	$u^{(d,1)}$ $(j\Delta x)^d$ $(l\Delta t)$	$u^{(d,2)}$ $(j\Delta x)^d$ $(l\Delta t)^2$	$u^{(d,3)}$ $(j\Delta x)^d$ $(l\Delta t)^3$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Figure III.3. Terms in the Taylor series of u_{i+j}^{n-l} illustrated in a matrix format. Constants in each term are omitted for clarity. Different colors represent terms from different groups, as explained in section III.B.2.

represented by the four different colors in the figure. Terms in blue are a function of Δx alone. These terms will appear in the Taylor series of the function when $j \neq 0$. Similarly, terms that are a function of Δt only are shown in green and they appear when $l \neq 0$. Terms in red are a function of both Δx and Δt, and these appear when $j \neq 0$ and $l \neq 0$. The term u in black is a function of neither Δx nor Δt and is present in the Taylor series of the function at any point and time level. In order to eliminate specific terms in the truncation error, it is apparent that we cannot arbitrarily choose the points and time levels in a stencil. They have to be selected according to the number of terms in each of these groups. For example, if a linear system consists of three equations that correspond to condition on terms that belong to red group, then the scheme would need the function evaluated at a minimum of three combinations of j and l such that $j \neq 0$ and $l \neq 0$. If not, the linear system may not have a solution or may have a solution which correspond to stencils completely biased towards the synchronous side of the stencil, like forward and backward differences.

The choice of stencil has consequences also in terms of the performance of simulation codes on parallel machines. Expanding the stencil in space will lead to larger message sizes to be sent over the network, which may be too expensive at extreme scales. Using multiple levels in time will keep the messages relatively smaller, but will increase the memory requirements at each PE. This choice, thus, would require information on the specific computing system to be used for the simulation.

The rectangular box in Fig. III.4 illustrates the layout of the stencil used in expressing the general scheme in Eq. (3.3). However, as mentioned earlier, not all the time levels at all points are requir ed to approximate the derivative. Instead, one can limit the number of time levels at each grid point in such a w ay to introduce the exact number of coefficients that would make the linear system solvable. Eq. (3.3),

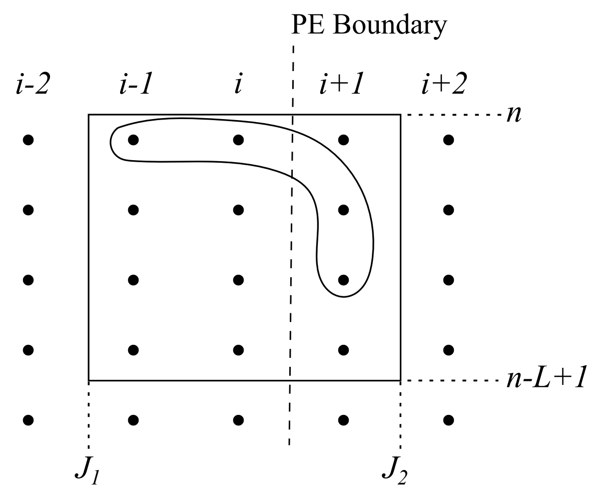


Figure III.4. Discretized one-dimensional domain in decomposed into two PEs.

then, becomes:

$$\left.\frac{\partial^d u}{\partial x^d}\right|_i^n \approx \sum_{j=-J_1}^{J_2} \sum_{l=\tilde{l}_1(j)}^{\tilde{l}_2(j)} \tilde{c}_j^l u_{i+j}^{n-l}, \tag{3.8}$$

where $\tilde{l}_1(j)$ and $\tilde{l}_2(j)$ are the lower and upper limits on the time levels used at the point $i+j$. These limits are computed according the latest time level available and the number of time levels chosen at that point in the stencil. As an example, in Fig. III.4 we identify a stencil to solve a system with four equations. At the two interior points the latest available time level is n, which has a zero delay. Thus, the limits are $\tilde{l}_1(j) = \tilde{l}_2(j) = 0$, for $j \in \{-1, 0\}$. As specified before, the latest available time level at the buffer point is given by $\tilde{n} = n - \tilde{k}_{i+1}$, and we use two successive time levels at this point. The limits on the time level at this point are then $\tilde{l}_1(1) = \tilde{k}_{i+1}$ and $\tilde{l}_2(1) = \tilde{k}_{i+1} + 1$.

A choice of stencil will lead to a scheme only when there exists a solution to the resulting linear system $\boldsymbol{A\tilde{c}} = \boldsymbol{b}$. Since $\boldsymbol{b} \neq \boldsymbol{0}$ due to the first condition in Eq. (3.7), the system is non-homogeneous and has a unique solution only when the matrix $\boldsymbol{A}$

is non-singular or has a full rank. If N_A is the size of the linear system, then the matrix has full rank when $rank(\boldsymbol{A}) = N_A$. We obtain the scheme by solving the system and substituting the coefficients into Eq. (3.8). On the other hand, when $rank(\boldsymbol{A}) < N_A$, the matrix is singular and the linear system possesses either no solution or infinite solutions. We can distinguish these two cases by computing the rank of the augmented matrix $\boldsymbol{A}|\boldsymbol{b}$. If $rank(\boldsymbol{A}) \neq rank(\boldsymbol{A}|\boldsymbol{b})$, then the system is inconsistent and the choice of stencil does not result in a scheme. In the case where $rank(\boldsymbol{A}) = rank(\boldsymbol{A}|\boldsymbol{b})$, the linear system is consistent, but has infinite solutions. The linear system, then, contains two or more equations that are linearly dependent. This means that for the choice of stencil, conditions on at least two of the terms are mathematically equivalent or condition on at least one of the terms can be obtained from linear transformation of others. In such a situation, we can get a scheme with greater accuracy with the same stencil. In some cases, it is possible to construct a smaller linear system with linearly independent equations, which can be solved by reducing the stencil size accordingly. The greater the number of linearly dependent equations, the smaller will be the linear system with linearly independent equations. This suggests that a judicious selection of grid points and time levels can be used to increase the number of linearly dependent equations in the resulting system, and thus reduce the stencil size which in turn reduces computations as well as the size of communication messages. This will be of interest in deriving asynchrony-tolerant schemes, which demand larger stencil due the presence of terms due to asynchrony.

Let us recall the second condition from Eq. (3.7), imposed to eliminate the terms due to asynchrony in deriving a scheme. After cancelling out the term $\Delta x^{\eta} \Delta t^{\zeta}/\eta!\zeta!$, which is constant across the equation corresponding for each (η, ζ), we get

$$\sum_{j=-J_1}^{J_2} \sum_{l=\tilde{l}_1(j)}^{\tilde{l}_2(j)} \tilde{c}_j^l j^\eta l^\zeta = 0. \tag{3.9}$$

It is evident from the above equation that the existence of linearly dependent equations rests on the values of j and l which are defined by the stencil, as well as η and ζ which represent the order of the derive corresponding to the equation. As mentioned earlier, the function in the stencil can belong to multiple time levels. The time level of the function at the interior points has a zero delay, that is $l = 0$, and hence, will not appear in equations corresponding to asynchrony terms. If we choose a single uniform time level with a delay $\tilde{k}_{i+j} = \tilde{k}$ for all $i + j \in B$ in the stencil, then $\tilde{l}_1(j) = \tilde{l}_2(j) = \tilde{k}$ which leads to a uniform value of l^ζ in Eq. (3.9). The equation then reduces to

$$\sum_{i+j \in B} \tilde{c}_j^{\tilde{k}} j^\eta = 0, \tag{3.10}$$

which is independent of ζ, and shows that for a given η, equations corresponding to $\zeta > 0$ are linearly dependent. With reference to Fig. III.3, when we eliminate a term in the red or green groups, all the other terms in the corresponding row that are in the same group are also eliminated. This illustration shows that it is possible to choose a stencil which results in linearly dependent equations in a system. Of course, there are other ways to obtain such systems, but we will not discuss them here.

We conclude this section by summarizing the steps to derive asynchrony-tolerant schemes.

1. Identify the terms on which conditions have to be imposed for a given d and a

2. Identify an appropriate stencil $(J_1, J_2, \tilde{l}_1(j), \tilde{l}_2(j))$ according to the terms in the

list

3. Compute the rank of the matrix $\boldsymbol{A}$
   - $rank(\boldsymbol{A}) = N_A$: unique solution
   - $rank(\boldsymbol{A}) < N_A$ and $rank(\boldsymbol{A}) \neq rank(\boldsymbol{A}|\boldsymbol{b})$: no solution, identify a new stencil
   - $rank(\boldsymbol{A}) < N_A$ and $rank(\boldsymbol{A}) = rank(\boldsymbol{A}|\boldsymbol{b})$: infinite solutions, add more conditions to get greater accuracy or reduce the system size with only linearly independent equations and adjust the stencil size
4. Solve for $\tilde{\boldsymbol{c}}$ and substitute the coefficients into general scheme

III.B.3. Alternative approach

It is often necessary to use schemes with a specific structure in terms of stencil and the corresponding coefficients to either improve computational performance or satisfy numerical properties. In the context of asynchrony-tolerant schemes, it is desirable to use schemes at PE boundary points that are similar in nature to those at interior points. Such an implementation may improve the overall stability of a numerical method and relieve the natural tendency of concentrated errors in the spatial distribution near PE boundaries (e.g. see Fig. (3) in Donzis & Aditya (2014)). Though the method described earlier gives the flexibility to choose a particular structure for the stencil, there is not much control over the nature of the resulting coefficients in the scheme. This is because the necessary conditions imposed on the terms on the right hand side of Eq. (3.6) are all solved in a single linear system. And, explicit conditions on the coefficients have to be added to the linear system to address this issue.

In an alternative approach to derive schemes, we propose to impose necessary conditions (similar to Eq. (3.7)) on the set of terms arising from Taylor series in a step-by-step process. In each step, a subset of higher order terms are eliminated, while retaining the derivative order term using a particular stencil. This process is repeated until the desired accuracy is achieved. Linear systems of smaller size can be constructed in each step to enforces the conditions and obtain the coefficients. The procedure described in bullet 3 in the summary of section III.B.1 should be used in computing the solution of these systems.

We now proceed to outline the procedure to derive schemes similar to central differences using this approach, and will later provide a detailed illustration in example 3.

Central difference schemes are widely used in solving parabolic and elliptic PDEs, and are shown to have low numerical dissipation, necessary to resolve all scales in multi-scale phenomena Hirsch (1994). If we consider the structure of central difference schemes, it can be characterized by a symmetric stencil about the point of computation, and symmetry coefficients (in absolute value). A general synchronous central difference scheme can be expressed as

$$\left.\frac{\partial^d u}{\partial x^d}\right|_i^n \approx \sum_{j=0}^{J} \phi_j \left(u_{i+j}^n + (-1)^d u_{i-j}^n\right), \tag{3.11}$$

where J determines the size of stencil and ϕ_j are the appropriate coefficients. Let us consider this stencil in the presence of asynchrony. In practical simulations each PE, typically, is assigned a large number of grid points. When asynchrony is allowed in such cases, delays are experienced only on one side of the stencil, that is, either on the left or right about the point of computation i for $i \in I_B$. If we assume the delay on the left side, which implies $i-j$ is a buffer point, then the terms in the sum in Eq. (3.11) take the form $\left(u_{i+j}^n + (-1)^d u_{i-j}^{n-l}\right)$. To maintain the above mentioned symmetries in asynchrony-tolerant schemes, we use this sum to eliminate some of the higher order terms and retain the derivative order term in the Taylor series in the first step. We consider coefficients to be symmetric, if they are so when $l = 0$, that is, when there is a zero delay. As delay is present only at $i-j$, none of the terms due to asynchrony in the expansion of u_{i-j}^{n-l} are cancelled out in the sum of the function at the two points. However, some of the terms, which are not a function of Δt, cancel out depending on the order of derivative d. If d is odd, then terms that correspond to even power of Δx cancel out, as shown next.

Consider the difference $u_{i+j}^n - u_{i-j}^n$ where we choose $l = 0$ to simplify the analysis. The conclusions, though, are valid for arbitrary delays $l > 0$. A Taylor series expansion can then be written as

$$u_{i+j}^n - u_{i-j}^n = 2\left[u^{(1,0)}\frac{(j\Delta x)}{1!} + u^{(3,0)}\frac{(j\Delta x)^3}{3!} + u^{(5,0)}\frac{(j\Delta x)^5}{5!} + \dots\right] \tag{3.12}$$

Similarly, if we consider the sum, $u_{i+j}^n + u_{i-j}^{n-l}$, terms with odd powers of Δx will vanish. This reduces some of the terms on which conditions need to be imposed, as we move on to the next step. A further decrease in the number of conditions can be achieved by artificially imposing the same delay of l levels on the other side of the stencil, that is, $\left(u_{i+j}^{n-l} + (-1)^d u_{i-j}^{n-l}\right)$. The Taylor series expansion of this difference, for odd d, is

$$\begin{aligned} u^{n-l}_{i+j} - u^{n-l}_{i-j} &= 2\left[u^{(1,0)}\frac{(j\Delta x)}{1!} + u^{(1,1)}\frac{(j\Delta x)(-l\Delta t)}{1!1!}\right. \\ &\left. +u^{(3,0)}\frac{(j\Delta x)^3}{3!} + u^{(1,2)}\frac{(j\Delta x)(-l\Delta t)^2}{1!2!} + \dots\right], \end{aligned} \tag{3.13}$$

which shows that all the terms with even powers of Δx, regardless of the power of Δt, are absent. Indeed, imposing this artificial delay on the function at the interior point, though demands additional storage of more time levels at each grid point, will lead to a smaller number of constraints. Thus, schemes with delay on both sides will need a smaller stencil to compute derivatives.

In Mudigere *et al.* (2014), this approach was used to recover the drop in accuracy due to delay in communication in central difference schemes. The authors further suggested that imposing delay on both sides of the stencil in central differences would suffice to maintain the accuracy under asynchronous conditions. However, it can be shown from Taylor series expansion that the schemes cannot be accurate beyond second order under the conditions they presented. It is essential to increase the stencil size to achieve higher order accuracy when asynchrony is present, as shown in this work.

The remaining higher order terms can be eliminated by expanding the stencil with additional terms of the form $\left(u^n_{i+j} + (-1)^d u^{n-l}_{i-j}\right)$ for different values of j or l. This can be done either in a single or multiple steps, and both of them will ensure symmetry in the coefficients. Assuming delay on the left of the stencil, the resultant asynchrony-tolerant scheme takes the form:

$$\left.\frac{\partial^d u}{\partial x^d}\right|^n_i \approx \sum_{j=0}^{J} \sum_{l=\tilde{l}_1(-j)}^{\tilde{l}_2(-j)} \tilde{\phi}^l_j \left(u^n_{i+j} + (-1)^d u^{n-l}_{i-j}\right) \tag{3.14}$$

It is often useful to derive asynchrony-tolerant schemes that reduce to central difference schemes when all delays are zero, that is $\tilde{k}_j = 0$ for $j \in B$. Such schemes can be derived by expanding the stencil in space, using the sum at different j, to eliminate terms that are not a function of Δt. And, use the sum at different levels in time to cancel out the terms due to asynchrony. This approach, which contains the essence of the alternative procedure, presented in this section will be illustrated in detail as Example 3 below.

III.B.4. Classification of schemes

In arriving at asynchrony-tolerant schemes there are several choices available in terms of choosing points and time levels in a stencil and on the nature of coefficients. We first provide a simple classification of asynchrony-tolerant schemes based on these choices, then we present some examples.

Let us consider the stencil of the general asynchrony-tolerant scheme in Eq. (3.8), which is given by the limits J_1 and J_2 in space and $\tilde{l}_1(j)$ and $\tilde{l}_2(j)$ in time. If $J_1 = J_2$, the number of points are equal on either sides of the point of computation i. We refer to this as a *symmetric* stencil in space. Else, $J_1 \neq J_2$ and the stencil is *asymmetric*. Regarding the nature of the delays, schemes can potentially have different delay values at different points in a stencil. However, enforcing a uniform delay across all the buffer points in a scheme, that is $\tilde{k}_{i+j} = \tilde{k}$ for all $i + j \in B$, may lead to linearly dependent conditions and a simpler implementation of schemes. We can, thus, classify schemes according to the presence or absence of uniform delay in schemes. In addition to the uniformity of delays, schemes can also be classified with respect to the time levels chosen at interior points. The function at these points can be either at the synchronous time level or at artificially imposed levels which, as we have shown, provide some numerical advantages.

Schemes can also be classified on the basis of the nature of coefficients. When asynchrony is present, a stencil with symmetric points may not necessarily give rise to symmetry in coefficients. This due to non-uniform time levels in the stencil at these points. To obtain symmetry in coefficients, like in standard central difference schemes, we have earlier proposed to use a sum or difference of the function at symmetric grid points. In this regard, we classify schemes with symmetric coefficients as the ones which have $|\tilde{c}^0_{-j}| = |\tilde{c}^0_j|$ (i.e. when $\tilde{k}_{i+j} = 0$). A summary of these classifications is given in Table III.1.

Feature	Classification	
Layout of grid points	symmetric $J_1 = J_2$	asymmetric $J_1 \neq J_2$
Nature of delay at buffer points	unconstrained	uniform delay $\tilde{k}_{i+j} = \tilde{K} \;\; \forall \;\; i+j \in B$
Artificial delay at interior point	zero delay $\tilde{k}_{i+j} = 0 \;\; \forall \;\; i+j \in I$	non-zero delay $\tilde{k}_{i+j} \leq 0 \;\; \forall \;\; i+j \in I$
Coefficients	symmetric $|\tilde{c}^0_{-j}| = |\tilde{c}^0_j|$	asymmetric $|\tilde{c}^0_{-j}| \neq |\tilde{c}^0_j|$

Table III.1. Summary of classification of asynchrony-tolerant schemes.

From the discussions in previous sections, it is clear that each of these classifications will have consequences in terms of numerical properties and computational performance of schemes. We will now proceed to derive three asynchrony-tolerant schemes and demonstrate how the conditions corresponding to the classification can be implemented in arriving at them.

Example 1: first derivative - second order accurate ($d = 1$, $a = 2$)
Using $r = 2$, conditions in Eq. (3.7) are imposed on terms that satisfy the inequality $\eta + 2\zeta < 3$. This gives rise to a linear system with four equations corresponding to the terms with $(\eta, \zeta) = \{(0,0), (1,0), (2,0), (0,1)\}$ in the Taylor series. The next step is to select a stencil with the function defined at four different combinations of points and time levels. Let, as before, $n - \tilde{k}_{i+j}$ be the latest available time level at a point $i+j \in B$ with $j > 0$. Further, let us choose the function set $\{u_{i-1}^n, u_i^n, u_{i+1}^{n-\tilde{k}_{i+1}}, u_{i+2}^{n-\tilde{k}_{i+2}}\}$ to construct the linear system $\boldsymbol{A}\tilde{\boldsymbol{c}} = \boldsymbol{b}$. The results is

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -\Delta x & 0 & \Delta x & 2\Delta x \\ \frac{\Delta x^2}{2} & 0 & \frac{\Delta x^2}{2} & 2\Delta x^2 \\ 0 & 0 & -\tilde{k}_{i+1}\Delta t & -\tilde{k}_{i+2}\Delta t \end{bmatrix} \begin{bmatrix} \tilde{c}_{i-1}^0 \\ \tilde{c}_i^0 \\ \tilde{c}_{i+1}^{\tilde{k}_{i+1}} \\ \tilde{c}_{i+2}^{\tilde{k}_{i+2}} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{3.15}$$

The rank of the coefficient matrix in the above equation is 4, which is equal to the size of the system. Thus, the choice of stencil results in a scheme without any further adjustments. After solving for the coefficients, we obtain the scheme as

$$\left.\frac{\partial u}{\partial x}\right|_i^n i = \frac{(-4\tilde{k}_{i+1}+\tilde{k}_{i+2})u_{i-1}^n + 3\tilde{k}_{i+1}u_i^n - \tilde{k}_{i+2}u_{i+1}^{n-\tilde{k}_{i+1}} + \tilde{k}_{i+1}u_{i+2}^{n-\tilde{k}_{i+2}}}{2(3\tilde{k}_{i+1}-\tilde{k}_{i+2})\Delta x} + \mathcal{O}\left(\frac{6\tilde{k}_{i+1}-\tilde{k}_{i+2}}{18\tilde{k}_{i+1}-6\tilde{k}_{i+2}}\Delta x^2, \frac{\tilde{k}_{i+1}\tilde{k}_{i+2}}{3\tilde{k}_{i+1}-\tilde{k}_{i+2}}\Delta t\right). \tag{3.16}$$

(3.17)

Note that the coefficients are a function of the random delay at buffer points. It is easy to see by inspection that this scheme has to be complemented in two specific circumstances. First, when $3\tilde{k}_{i+1}-\tilde{k}_{i+2}=0$ the approximation has an infinite value. This can be avoided by artificially altering the delays such that $3\tilde{k}_{i+1}-\tilde{k}_{i+2}\neq 0$. Second, when the function at both the buffer points is at a synchronous time level, i.e., no delay. In this case, an implementation of the above scheme will result in an indeterminate form. In that case one can use

$$\left.\frac{\partial u}{\partial x}\right|_i^n = \frac{-3u_{i-1}^n + 3u_i^n - u_{i+1}^{n-\tilde{k}} + u_{i+2}^{n-\tilde{k}}}{4\Delta x} + \mathcal{O}\left(\Delta x^2, \tilde{k}^2\Delta t\right), \tag{3.18}$$

which is obtained by substituting $\tilde{k}_{i+1}=\tilde{k}_{i+2}=\tilde{k}$ and simplifying the expression in Eq. (3.17). It is interesting to see that the coefficients in the above scheme, with a uniform delay across the buffer points, are independent of the delay value, which eliminates the limitations of the scheme in Eq. (3.17). Similar schemes can be derived by considering delays on the left of the stencil.

Example 2: second derivative - second order accurate ($d=2$, $a=2$)

The relationship between the time step and grid spacing is assumed as $\Delta t \sim \Delta x$. For these set of parameters, Eq. (3.7) enforces conditions on 21 terms in the Taylor series, which are highlighted in red in Fig. III.5. The resulting linear system has 21

u	$u^{(0,1)}$ $(l\Delta t)$	$u^{(0,2)}$ $(l\Delta t)^2$	$u^{(0,3)}$ $(l\Delta t)^3$	$u^{(0,4)}$ $(l\Delta t)^4$	$u^{(0,5)}$ $(l\Delta t)^5$	$\cdots$
$u^{(1,0)}$ $(j\Delta x)$	$u^{(1,1)}$ $(j\Delta x)$ $(l\Delta t)$	$u^{(1,2)}$ $(j\Delta x)$ $(l\Delta t)^2$	$u^{(1,3)}$ $(j\Delta x)$ $(l\Delta t)^3$	$u^{(1,4)}$ $(j\Delta x)$ $(l\Delta t)^4$	$u^{(1,5)}$ $(j\Delta x)$ $(l\Delta t)^5$	$\cdots$
$u^{(2,0)}$ $(j\Delta x)^2$	$u^{(2,1)}$ $(j\Delta x)^2$ $(l\Delta t)$	$u^{(2,2)}$ $(j\Delta x)^2$ $(l\Delta t)^2$	$u^{(2,3)}$ $(j\Delta x)^2$ $(l\Delta t)^3$	$u^{(2,4)}$ $(j\Delta x)^2$ $(l\Delta t)^4$	$u^{(2,5)}$ $(j\Delta x)^2$ $(l\Delta t)^5$	$\cdots$
$u^{(3,0)}$ $(j\Delta x)^3$	$u^{(3,1)}$ $(j\Delta x)^3$ $(l\Delta t)$	$u^{(3,2)}$ $(j\Delta x)^3$ $(l\Delta t)^2$	$u^{(3,3)}$ $(j\Delta x)^3$ $(l\Delta t)^3$	$u^{(3,4)}$ $(j\Delta x)^3$ $(l\Delta t)^4$	$u^{(3,5)}$ $(j\Delta x)^3$ $(l\Delta t)^5$	$\cdots$
$u^{(4,0)}$ $(j\Delta x)^4$	$u^{(4,1)}$ $(j\Delta x)^4$ $(l\Delta t)$	$u^{(4,2)}$ $(j\Delta x)^4$ $(l\Delta t)^2$	$u^{(4,3)}$ $(j\Delta x)^4$ $(l\Delta t)^3$	$u^{(4,4)}$ $(j\Delta x)^4$ $(l\Delta t)^4$	$u^{(4,5)}$ $(j\Delta x)^4$ $(l\Delta t)^5$	$\cdots$
$u^{(5,0)}$ $(j\Delta x)^5$	$u^{(5,1)}$ $(j\Delta x)^5$ $(l\Delta t)$	$u^{(5,2)}$ $(j\Delta x)^5$ $(l\Delta t)^2$	$u^{(5,3)}$ $(j\Delta x)^5$ $(l\Delta t)^3$	$u^{(5,4)}$ $(j\Delta x)^5$ $(l\Delta t)^4$	$u^{(5,5)}$ $(j\Delta x)^5$ $(l\Delta t)^5$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Figure III.5. Terms in the Taylor series of u_{i+j}^{n-l} illustrated in a matrix format. Constants in each term are omitted for clarity. Conditions in Eq. (3.7) are imposed on the red color terms.

equations which, in principle, will need the function at 21 combinations of points and time levels. However, Eq. (3.9) has shown that a stencil with only two time levels, n for interior points and $n-\tilde{k}$ for buffer points, will lead to linearly dependent equations in the system. We use this choice of stencil to reduce the size of the linear system. Choosing the limits $\{J_1, J_2\} = \{5, 6\}$ in space and assuming the buffer points are on the right side of the stencil, leads to a smaller linear system with 11 equations that has a unique solution. Upon solving the system, the resulting asynchrony-tolerant scheme is

$$\begin{aligned}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n &= \frac{1}{12\Delta x^2}\Big[35u_{i-5}^n - 164u_{i-4}^n + 294u_{i-3}^n - 236u_{i-2}^n + 71u_{i-1}^n - 45u_{i+1}^{n-\tilde{k}} \\ &\quad +225u_{i+2}^{n-\tilde{k}} - 450u_{i+3}^{n-\tilde{k}} + 450u_{i+4}^{n-\tilde{k}} - 225u_{i+5}^{n-\tilde{k}} + 45u_{i+6}^{n-\tilde{k}}\Big] \\ &\quad +\mathcal{O}\left(\Delta x^4, \tilde{k}\Delta x^3 \Delta t\right). \end{aligned} \tag{3.19}$$

Note that a single linear system has been used to obtain the above scheme.

Example 3: second derivative - fourth order accurate ($d = 2$, $a = 4$)

In this example, we will use the alternative step-by-step approach described in section III.B.3 to derive an asynchrony-tolerant scheme that reduces to a standard central difference scheme in the absence of delays. If we consider the Taylor series of u at a generic point and time level in a stencil, and assume $\Delta t \sim \Delta x^2$, conditions have to be imposed on terms with

$$\begin{aligned}(\eta, \zeta) &= \{(0,0),(1,0),(2,0),(3,0),(4,0), \\ &\qquad (0,1),(1,1),(2,1),(3,1),(0,2),(1,2)\}. \end{aligned} \tag{3.20}$$

In order to maintain a symmetry in the stencil points and coefficients, we use the sum $\left(u_{i+j}^n + u_{i-j}^{n-l}\right)$ in the first step, which eliminates the terms with $(\eta,\zeta) = \{(1,0),(3,0)\}$ upon Taylor series expansion. In the second step, conditions are enforced on the terms that are only a function of Δx or the terms with $\zeta = 0$ by expanding the stencil in space. These are the three terms corresponding to $(\eta,\zeta) = \{(0,0),(2,0),(4,0)\}$, which result in three equations using the function $\left(u_{i+j}^n + u_{i-j}^{n-l}\right)$ for $j = 0, 1, 2$.

$$\begin{bmatrix} 2 & 2 & 2 \\ 0 & \Delta x^2 & 4\Delta x^2 \\ 0 & \frac{\Delta x^4}{12} & \frac{4\Delta x^4}{3} \end{bmatrix} \begin{bmatrix} \tilde{\phi}_0^l \\ \tilde{\phi}_1^l \\ \tilde{\phi}_2^l \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{3.21}$$

After solving the equations we obtain the linear combination of the function that is free from necessary higher order synchronous terms. We can express the linear combination as

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n = \frac{-u_{i+2}^n + 16u_{i+1}^n - 30u_i^n + 16u_{i-1}^{n-l} - u_{i-2}^{n-l}}{12\Delta x^2} + \mathcal{O}(\Delta x^4, l\Delta t, l\Delta t/\Delta x^2). \quad (3.22)$$

When $l = 0$, the terms due to asynchrony disappear from the above expression, and clearly represents the standard fourth order central difference scheme. In the next step, we eliminate the remaining higher order terms which appear due to asynchrony. If we expand the stencil further in space, which corresponds to $j > 2$, then the scheme would possess the required symmetries, but will not reduce to the fourth order central difference in the absence of delay. On the other hand, when the stencil size is increased in time, i.e., $l \in \{\tilde{k}, \tilde{k}+1, \tilde{k}+2, \dots\}$, we get a scheme that resembles a standard central difference. The conditions on the six asynchrony terms from the set in Eq. (3.20), need Eq. (3.22) at six time levels. However, with the use of multiple time levels, the resulting linear system has three linearly dependent conditions. We find that conditions on terms with the same ζ are all mathematically equivalent. This reduces the size of the linear system that uses the linear combination in Eq. (3.22) at $l \in \{\tilde{k}, \tilde{k}+1, \tilde{k}+2\}$ to three equations:

$$\begin{bmatrix} -\tilde{k}\frac{5\Delta t}{4\Delta x^2} & -(\tilde{k}+1)\frac{5\Delta t}{4\Delta x^2} & -(\tilde{k}+2)\frac{5\Delta t}{4\Delta x^2} \\ \tilde{k}^2\frac{5\Delta t^2}{8\Delta x^2} & (\tilde{k}+1)^2\frac{5\Delta t^2}{8\Delta x^2} & (\tilde{k}+2)^2\frac{5\Delta t^2}{8\Delta x^2} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \tilde{\phi}_0^{-\tilde{k}} \\ \tilde{\phi}_1^{-\tilde{k}-1} \\ \tilde{\phi}_2^{-\tilde{k}-2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.23)$$

The solution to this linear system results in the scheme

$$\begin{aligned}
\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n &= \frac{1}{2}(\tilde{k}^2+3\tilde{k}+2)\frac{-u_{i+2}^n+16u_{i+1}^n-30u_i^n+16u_{i-1}^{n-\tilde{k}}-u_{i-2}^{n-\tilde{k}}}{12\Delta x^2} \\
&-(\tilde{k}^2+2\tilde{k})\frac{-u_{i+2}^n+16u_{i+1}^n-30u_i^n+16u_{i-1}^{n-\tilde{k}-1}-u_{i-2}^{n-\tilde{k}-1}}{12\Delta x^2} \\
&+\frac{1}{2}(\tilde{k}^2+\tilde{k})\frac{-u_{i+2}^n+16u_{i+1}^n-30u_i^n+16u_{i-1}^{n-\tilde{k}-2}-u_{i-2}^{n-\tilde{k}-2}}{12\Delta x^2} \\
&+\mathcal{O}\left(\Delta x^4, \tilde{k}(\tilde{k}+1)(\tilde{k}+2)\Delta t^3, \tilde{k}(\tilde{k}+1)(\tilde{k}+2)\Delta t^3/\Delta x^2\right). \quad (3.24)
\end{aligned}$$

Note that, like the scheme in example 1, the coefficients in the above scheme are a function of the random delay $\tilde{k}$. However, unlike Eq. (3.17) in example 1, this scheme can take any delay value in the range $[0, L-1]$.

III.C. Error analysis

In previous sections, we presented a method to derive asynchrony-tolerant schemes of arbitrary accuracy. As explained earlier, these schemes are, typically, used at PE boundaries ($i \in I_B$) where asynchrony is experienced. The number of computations that are carried out asynchronously in a domain depends on the number of PEs used to solve the problem, the stencil size of schemes used at interior points and statistics of the random delays, which in turn depend on the characteristics of communications in a computing system. These dependencies bring new challenges while trying to understand the overall accuracy of these asynchrony-tolerant schemes. First, due to the random nature of the delay, the associated truncation error is also random in nature. Second, schemes to compute spatial derivatives at interior points are not the same as asynchrony-tolerant schemes at PE boundary points and have, thus, different truncation errors. These issues result in a non-homogeneity of error in the domain, both in space as well as time.

In our previous work Donzis & Aditya (2014), we have proposed a statistical description to analyze the overall error and determine the accuracy of the numerical solution. We follow a similar procedure in this work. Before we develop the error analysis, we present some necessary definitions that will be used. First, let us define the probability of having a time level $\tilde{n} = n - \tilde{k}_i$ at a grid point i as $p_{k[i]}$. The sum of probabilities of all levels at point i is obviously

$$\sum_{k[i]=0}^{L-1} p_{k[i]} = 1. \tag{3.25}$$

To obtain the statistics of the error, we define two types of averages for a variable f: a space average and an ensemble average. The space average can be performed over all points in the set I or the subsets I_I and I_B. If the average is over the entire domain, that is $i \in I$, it is denoted by angular brackets and given by $\langle f \rangle = \sum_{i=1,N} f_i/N$. On the other hand, the average over the points in the subsets I_I and I_B are given by $\langle f \rangle_B = \sum_{i \in I_B} f_i/N_B$ and $\langle f \rangle_I = \sum_{i \in I_I} f_i/N_I$, respectively. The random nature of delays is taken into account by ensemble averages, which is denoted by an overline $\overline{f}$.

A common measure of the error incurred by using a finite difference representation of the original PDE is given by the so called truncation error. Formally, it is given by the difference between the PDE and the approximate finite difference equation, that is $E = PDE - FDE$. As introduced in Donzis & Aditya (2014) and Ch. II, the assessment of the error of asynchrony schemes which are random in nature and heterogeneous in space, can be done by applying the two averages described above.

That is,

$$\langle \overline{E} \rangle = \frac{1}{N} \sum_{i=1,N} \overline{E_i^n}, \tag{3.26}$$

where E_i^n is the truncation error at the point i and time level n. Due to the non-uniform expression for the truncation error at interior and PE boundary points, it is convenient to split the error according to the two sets of points:

$$\langle \overline{E} \rangle = \frac{1}{N} \left[\sum_{i \in I_I} E_i^n + \sum_{i \in I_B} \overline{\tilde{E}_i^n} \right] \tag{3.27}$$

Note that the error due to interior points does not possess randomness due to delays and are, hence, unaffected by the ensemble average. On the other hand, errors at PE boundary points have both random asynchronous and deterministic synchronous components. This allows us to further split the error in the set I_B as

$$\langle \overline{E} \rangle = \frac{1}{N} \left[\sum_{i \in I_I} E_i^n + \sum_{i \in I_B} E_i^n|_s + \sum_{i \in I_B} \overline{\tilde{E}_i^n}|_a \right], \tag{3.28}$$

where the subscripts s and a denote the synchronous and asynchronous components, respectively. It is clear that in the absence of delays $\overline{\tilde{E}_i^n}|_a = 0$.

The order of accuracy of a scheme will depend on the leading order term in each of the error terms in the above equation. These terms comprise the sum of the truncation error due to all terms in the original PDE, including the time derivative. Thus, it is important to choose the accuracy of time integration to match the order of accuracy of space derivatives. We will discuss this topic next and then present an example to illustrate the effect of asynchrony on the error. We will end this section with a generalization of the results on accuracy of asynchronous schemes.

III.C.1. Time integration

To understand the effect of time discretization on the overall order of accuracy, let us consider the equation $\partial u/\partial t = f$, where f depends on spatial derivatives of u, integrated using Euler scheme. The scheme is first order in time with the leading order term being $-u^{(0,2)}\Delta t/2$. As mentioned in section III.B.1, if we assume a relation of the form $\Delta t \sim \Delta x^r$, then the leading order term is equivalent to $\mathcal{O}(\Delta x^r)$ in space. When the accuracy of the space derivatives is greater than r, the total error will, very likely, be dominated by the temporal term and will dictate the order of accuracy of the solution.

Thus, if a certain order is desired for space derivatives, it is important to select a time discretization with the same (or greater) order to keep the overall order unchanged. We will follow this practice as we demonstrate the accuracy of the proposed asynchrony-tolerant schemes next. For this, we choose linear multi-step method to compute the time derivative. A general expression with T time steps is given by

$$u_i^{n+1} = u_i^n + \Delta t \sum_{m=0}^{T-1} \beta_m f_i^m, \tag{3.29}$$

where the coefficients β_m determine the particular temporal scheme Stoer & Bulirsch (2013).

The advantage of using a temporal scheme of the form Eq. (3.29) is that the terms f_i^m can be computed using asynchrony-tolerant schemes and are thus,free of asynchrony errors to the desired order of accuracy. Thus, so will the linear combination of f_i at different time steps. For example, if one uses an asynchrony-tolerant scheme which is fourth order accurate, with $r = 2$ (i.e. $\Delta t \sim \Delta x^2$), then one needs a temporal scheme with second order accuracy to maintain fourth order accuracy globally. This can be accomplished by a two-step Adams-Bashforth method

$$u_i^{n+1} = u_i^n + \Delta t \left(\frac{3}{2} f_i^n - \frac{1}{2} f_i^{n-1} \right), \tag{3.30}$$

which is readily shown to be second order in time Stoer & Bulirsch (2013). The generalization to higher orders is straightforward.

III.C.2. Example: heat equation with fourth-order accurate asynchrony-tolerant schemes

Let us consider the 1D heat equation,

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \tag{3.31}$$

where $u(x, t)$ is the temperature and α is the thermal diffusivity of the medium. The above equation is solved on a uniform grid shown in Fig. III.1 with periodic boundary conditions.The equation is approximated with the second order Adams-Bashforth scheme shown in Eq. (3.30) and standard fourth order central difference for the space derivative at interior points.At the PE boundary points, the space derivative is computed with the asynchrony -tolerant scheme Eq. (3.24) derived in Example 3.

Using Taylor series, the truncation error at interior points is

$$E_i^n = \left(-\frac{1}{6} u^{(0,3)} - \frac{1}{4} \alpha u^{(2,2)} \right) \Delta t^2 - \frac{1}{90} \alpha u^{(6,0)} \Delta x^4 + \mathcal{O}\left(\Delta x^6, \Delta t^3, \Delta x^4 \Delta t \right). \tag{3.32}$$

As mentioned above, at PE boundary points, the truncation error can be split into the synchronous and asynchronous components,

$$\tilde{E}_i^n|_{\tilde{k}=k} = E_i^n|_s + \tilde{E}_i^n|_{a,\tilde{k}=k}. \tag{3.33}$$

Because by construction, the asynchrony-tolerant scheme in Eq. (3.24) reduces to the standard central difference in the absence of delay, the synchronous component of the error, $E_i^n|_s$, is the same as Eq. (3.32). Note that this scheme assumes a uniform delay at buffer points. Hence, we drop the subscript for $\tilde{k}$ in the above expression. The asynchronous component of the error, considering delays only on the left side of the stencil, can be readily shown to be

$$\tilde{E}_i^n|_{a,\tilde{k}=k} = -\frac{5}{24}\left(k^3+3k^2+2k\right)\alpha u^{(0,3)}\frac{\Delta t^3}{\Delta x^2} + \mathcal{O}\left(k^3\Delta t^3/\Delta x\right). \tag{3.34}$$

The leading order term in the error remains the same when the delays are experienced on the right of the stencil. Clearly, when $\tilde{k}=0$, we have $\tilde{E}_i^n|_{a,\tilde{k}=k}=0$ and thus also its ensemble average. On the other hand, if $\tilde{k}>0$, then the ensemble average is

$$\begin{aligned}
\overline{\tilde{E}_i^n|_a} &\approx \sum_{k=0}^{L-1} p_k \tilde{E}_i^n|_{a,\tilde{k}=k} \\
&\approx \sum_{k=0}^{L-1} p_k \left(-\frac{5}{24}\left(k^3+3k^2+2k\right)\alpha u^{(0,3)}\frac{\Delta t^3}{\Delta x^2}\right) \\
&\approx \left(-\frac{5}{24}\alpha u^{(0,3)}\frac{\Delta t^3}{\Delta x^2}\right)\sum_{k=0}^{L-1} p_k\left(k^3+3k^2+2k\right) \\
&\approx \left(-\frac{5}{24}\alpha u^{(0,3)}\frac{\Delta t^3}{\Delta x^2}\right)\left(\overline{\tilde{k}^3}+3\overline{\tilde{k}^2}+2\overline{\tilde{k}}\right),
\end{aligned} \tag{3.35}$$

where moments are given by $\overline{\tilde{k}^n} = \sum_{k=0,L-1} p_k k^n$. It is interesting that the average error under the presence of asynchrony, depends not just on the mean of the delay as in Donzis & Aditya (2014), but also on its higher order moments. The implication of this result, is that in assessing the performance of asynchronous numerical schemes a certain degree of details about the architecture of the computing system would be needed, such as the probability density function of the delays $\tilde{k}$. Conversely, one can quantitatively compare the performance of different computing systems by comparing moments of $\tilde{k}$.

We now substitute the leading order terms in Eqs. (3.32) and (3.35) into Eq. (3.33). Assuming the statistics of the delays are homogeneous in space, the average error is

$$\begin{aligned}\langle \overline{E} \rangle \;\approx\; & \frac{1}{N}\Bigg[\sum_{i\in I_I}\left(\left(-\frac{1}{6}u^{(0,3)}-\frac{1}{4}\alpha u^{(2,2)}\right)\Delta t^2-\frac{1}{90}\alpha u^{(6,0)}\Delta x^4\right)\\ & +\sum_{i\in I_B}\left(\left(-\frac{1}{6}u^{(0,3)}-\frac{1}{4}\alpha u^{(2,2)}\right)\Delta t^2-\frac{1}{90}\alpha u^{(6,0)}\Delta x^4\right)\\ & +\sum_{i\in I_B}\left(\left(-\frac{5}{24}\alpha u^{(0,3)}\frac{\Delta t^3}{\Delta x^2}\right)\left(\overline{\tilde{k}^3}+3\overline{\tilde{k}^2}+2\overline{\tilde{k}}\right)\right)\Bigg]. \end{aligned} \tag{3.36}$$

In the above equations, the first two sums on the right hand side are due to synchronous computations and can be conveniently combined by noting that $I = I_I \cup I_B$.To determine the spatial accuracy of the solution, we use the stability parameter $r_\alpha = \alpha\Delta t/\Delta x^2$ to substitute time step Δt in terms of Δx. This corresponds to $r = 2$ in the formulation presented in section III.B.1. The above equation then reduces to

$$\begin{aligned}\langle \overline{E} \rangle \;\approx\; & \frac{1}{N}\Bigg[\sum_{i\in I}\left(-\frac{1}{6}\alpha^2 r_\alpha^2 u^{(0,3)}-\frac{1}{4}\alpha^3 r_\alpha^2 u^{(2,2)}-\frac{1}{90}\alpha u^{(6,0)}\right)\Delta x^4\\ & +\sum_{i\in I_B}\left(-\frac{5}{24}\alpha^4 r_\alpha^3 u^{(0,3)}\right)\left(\overline{\tilde{k}^3}+3\overline{\tilde{k}^2}+2\overline{\tilde{k}}\right)\Delta x^4\Bigg], \end{aligned} \tag{3.37}$$

which can be rewritten as

$$\begin{aligned}\langle \overline{E} \rangle \;\approx\; & \left[-\frac{1}{6}\alpha^2 r_\alpha^2\langle u^{(0,3)}\rangle-\frac{1}{4}\alpha^3 r_\alpha^2\langle u^{(2,2)}\rangle-\frac{1}{90}\alpha\langle u^{(6,0)}\rangle\right]\Delta x^4\\ & +\left[\frac{N_B}{N}\left(\overline{\tilde{k}^3}+3\overline{\tilde{k}^2}+2\overline{\tilde{k}}\right)\left(-\frac{5}{24}\alpha^4 r_\alpha^3\langle u^{(0,3)}\rangle_B\right)\right]x^4. \end{aligned} \tag{3.38}$$

The average error is clearly seen to possess components due to synchronous and asynchronous computations.Either of the terms can dominate the average error depending on physical parameters (e.g. α, initial conditions, etc.), numerical parameters (e.g. Δx, r_α, etc.), and simulation parameters (e.g. P, network performance, etc.). If the synchronous part dominates the overall error, then the resulting scheme is fourth order accurate, that is $\langle \overline{E} \rangle \sim \mathcal{O}(\Delta x^4)$. If, on the other hand, the asynchronous component dominates, the error is given by

$$\langle \overline{E} \rangle \approx \frac{P(J_1 + J_2)}{N} \left(\overline{\tilde{k}^3} + 3\overline{\tilde{k}^2} + 2\overline{\tilde{k}} \right) \left(-\frac{5}{24} \alpha^4 r_\alpha^3 \langle u^{(0,3)} \rangle_B \right) \Delta x^4, \tag{3.39}$$

where we have used $N_B = (J_1 + J_2)P$, with J_1 and J_2 being the stencil size in space at interior points. Using $N = \mathcal{L}/\Delta x$, where $\mathcal{L}$ is the length of the domain, and for all other parameters kept constant, the average error is found to scale as

$$\begin{aligned} \langle \overline{E} \rangle &\sim \frac{P}{N} \left(\overline{\tilde{k}^3} + 3\overline{\tilde{k}^2} + 2\overline{\tilde{k}} \right) \Delta x^4 \\ &\sim P \left(\overline{\tilde{k}^3} + 3\overline{\tilde{k}^2} + 2\overline{\tilde{k}} \right) \Delta x^5 \end{aligned} \tag{3.40}$$

Interestingly, the order of accuracy of the numerical method now depends on how the problem is scaled on a parallel machine. In the case of weak scaling, where the computational effort per PE is kept constant, that is $P/N = const$, the error varies as Δx^4 and the method is fourth order accurate in space. On the other hand, when the total computational effort is kept constant ($N = const$) and the simulations are carried out on increasingly large number of PEs, the average error is $\langle \overline{E} \rangle \sim \mathcal{O}(\Delta x^5)$ and the method is fifth order accurate. We also observe that the error scales linearly with P.

III.C.3. Generalization

We now proceed to generalize the expressions for average error ($\langle\overline{E}\rangle$) presented in Eq. (3.40). For this, we restate the conditions and assumptions that lead to Eq. (3.40). First, we assumed that the asynchronous component dominates the overall error. Second, the asynchrony-tolerant scheme used in the analysis is fourth order accurate, which lead to an $\mathcal{O}(\Delta x^4)$ leading term due to asynchrony. We have also assumed a uniform random delay in the stencil, that is $\tilde{k}_{i+j} = \tilde{k}$ for all $i+j \in B$. Also, the scheme uses three successive asynchronous time levels (with delays $\tilde{k}$, $\tilde{k}+1$, $\tilde{k}+2$), which results in a cubic polynomial in k in the leading order error term.

With the above observations, we can arrive at a general case which uses asynchrony-tolerant schemes with $\mathcal{T}$ number of successive asynchrony time levels and is accurate to an order a. If asynchrony component dominates the average error, then it is easy to generalize Eq. (3.40) as:

$$\begin{aligned}\langle\overline{E}\rangle &\sim \frac{P}{N}\Delta x^a \sum_{m=1}^{\mathcal{T}} \gamma_m \overline{\tilde{k}^m} \\ &\sim P\Delta x^{a+1} \sum_{m=1}^{\mathcal{T}} \gamma_m \overline{\tilde{k}^m} \qquad (3.41)\end{aligned}$$

Note that the average error still scales linearly with the number of PEs. However, higher order moments of the delay are necessary to characterize the error when the stencil size of asynchrony-tolerant schemes is expanded in time. A minimum accuracy of order a is then assured, regardless of how simulations are scaled up.

III.D. Summary

In this chapter, we have presented a methodology to derive asynchrony-tolerant schemes of arbitrary accuracy. We have also provided an alternative approach to obtained schemes that resemble widely used central difference schemes in the absence of asynchrony. A simple classification of the schemes based on the structure of stencil and nature of coefficients has been presented. Sample derivations of asynchrony-tolerant schemes that belong to different classes are shown to illustarte the proposed methodolgy. We, later, provided a procedure to analyze the overall error in a numerical method which uses standard synchronous scheme at the interior points and asynchrony-tolerant scheme at the PE boundary points in the domain. Theoretical predictions show that the average error not just depends on the mean of the random delay, but also on its higher order moements.

CHAPTER IV

IMPLEMENTATIONS

In this chapter, we turn our attention to actual application of the asynchronous computing method to solve PDEs. We provide two different implementations to evaluate the numerical and computational performance of the proposed method. In the first implementation, delays experienced due to asynchrony are artificially imposed using a random number generator. Such an implementation will permit a complete control over the statistics of message delays and allows us to assess the theoretical predictions, made in the previous chapters, in different parameter regimes. The second implementation is an approach to use the asynchronous computing method in simulation of real life applications on massively parallel supercomputers. The implementation uses communications based on Message Passing Interface (MPI) to exchange information between different PEs in an asynchronous fashion. Before presenting the two implementations, we briefly define the problem used in numerical simulations.

IV.A. Problem definition

For the simulations, our focus will be on the advection-diffusion equation as it contains both first and second derivatives. The interest of this equation is well known in fluid mechanics as it governs the time evolution of fluid flow phenomena involving convection and diffusion processes. The equation is

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = \alpha\frac{\partial^2 u}{\partial x^2} \tag{4.1}$$

where $u = u(x,t)$ is the velocity, c is the convection speed and α the viscosity coefficient. If c is a constant, the equation is linear and possess simple analytical solution with which we can compare our theoretical developments. This provides a systematic way to compare the results from numerical simulation and verify the accuracy of the schemes. When $c = u$, Eq. (4.1) is nonlinear in nature and is the celebrated viscous Burgers' equation. This equation helps us understand the effect of asynchrony on multi-scale phenomena.

In this problem, we assume an initial condition given by a sum of sinusoidal waves in a periodic domain of length 2π:

$$u(x,0) = \sum_{\kappa} A(\kappa)\sin(\kappa x + \phi_\kappa) \tag{4.2}$$

where κ is the wavenumber, and $A(\kappa)$ and ϕ_κ are the amplitude and phase angle corresponding to a particular wavenumber κ. The analytical solution of Eq. (4.1) and Eq. (4.2), indicated by a subscript a, when c is a constant is readily found to be

$$u_a(x,t) = \sum_{\kappa} e^{-\alpha\kappa^2 t} A(\kappa)\sin(\kappa x + \phi_\kappa - ct). \tag{4.3}$$

The reason for including the phase ϕ_κ is to avoid situations in which the PE boundaries coincide with specific features of the solution which may lead to very particular conditions in the truncation error. As an example, suppose we solve the heat equation with initial conditions given by a single sine wave and that the number of processors P is such that the PE boundaries are co-located with the zero crossings of this initial condition. From Eq. (4.3) with $c = 0$, it is clear that the solution at the zero crossings (and PE boundaries) will not change with time, that is $\dot{u} = 0$. Thus, the average over all boundary points will also vanish: $\langle \dot{u} \rangle_B = 0$. By looking at Eq. (2.40), we see that the leading order term vanishes and thus the resulting scheme is of higher order that it would in a more general condition. To avoid these very particular cases, the phases are chosen randomly for different wavenumbers and the results are averaged over this space too.

The numerical solutions of Eq. (4.1) and Eq. (4.2) are obtained with different standard synchronous schemes as well as asynchrony-tolerant schemes. The details of the schemes used in particular simulation are provided along with results in the next chapter.

IV.B. Implementation I

To illustrate this implementation, we discretize Eq. (4.1) using second order central differences in space and a forward first order difference in time. For grid points close to a PE boundary on the right, we have

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^{\tilde{n}} - u_{i-1}^n}{2\Delta x} = \alpha\frac{u_{i+1}^{\tilde{n}} - 2u_i^n + u_{i-1}^n}{\Delta x^2}. \tag{4.4}$$

A similar expression can be written for grid points close the PE boundary on the left, which will have delays on $i - 1$, instead of $i + 1$.

As mention in the beginning of this chapter, in order to have complete control over the statistics of message delays and compare against the theoretical predictions, we use random number generators to simulate these delays. In particular, the delays on both sides are drawn from uniform distributions in the interval $[0, 1]$

with some initial random seed. For a given L, we arbitrarily set the probabilities $\{p_{0[i+1]}, p_{1[i+1]}, \ldots, p_{L-1[i+1]}\}$ corresponding to delays $\tilde{k}_{i+1} = 0, 1, 2, \ldots, L-1$ respectively by comparing each random number to the corresponding partition of $[0, 1]$ into L bins. Since in our numerical experiments we use i.i.d. random sequences at the different PE boundaries, there is no dependence on location and we will thus drop the subscript $i+1$ for simplicity and write $\{p_0, p_1, \ldots, p_{L-1}\}$. For example, for $L = 3$, the set $\{p_0, p_1, p_2\} = \{0.6, 0.3, 0.1\}$ represents the situation where the probability of having $\tilde{k}_{i+1} = 0$ (i.e. no delay), $\tilde{k}_{i+1} = 1$ and $\tilde{k}_{i+1} = 2$ is 0.6, 0.3 and 0.1, respectively. In this case the mean $\overline{\tilde{k}_{i+1}}$ is easily found to be $\sum_{k=0,2} p_k k = 0.5$. Ensemble averages in practice are taken by running multiple simulations with different initial seeds for the random number generator.

To compare against our theoretical predictions in previous sections, we define the error at grid point i and time level n as $E_i^n = u_i^n - u_a(x_i, t_n)$ with $u_a(x, t)$ from Eq. (4.3) and take the different averages presented in Ch. II.

IV.C. Implementation II

In this section, we present a highly scalable asynchronous implementation to solve PDEs on massively parallel machines, which exploits the relaxation of synchronization between PEs at a mathematical level. We first understand the features that are necessary to facilitate asynchronous computations. We then present a novel asynchronous communication algorithm based on two different communication models available with the current MPI standard. Later we provide details of the asynchronous implementation, and also, discuss commonly used synchronous implementation to compare our results on scalability. Let us review the mathematical requirements that are necessary to carry out asynchronous computations using asynchrony-tolerant schemes. Fig. IV.1 compares the stencil size of schemes under asynchrony relative to a synchronous scheme. If the size of the asynchrony-tolerant scheme is same as synchronous scheme, then the computational effort remains the same. However, such schemes will have relatively lower accuracy. As discussed in Ch. III, it is necessary to increase the size of stencil to maintain the accuracy similar to synchronous schemes. When the stencil size is increased in space, the asynchrony-tolerant scheme will result in larger message size. On the other hand, an increase in stencil size in time will lead to additional storage or memory requirements. These consequences, in regard to the computational performance, should be taken into account while selecting schemes for simulations.

Consider a typical example of asynchrony-tolerant scheme to understand the implementation requirements of asynchronous computing method.

$$\begin{aligned}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n &= \frac{1}{2}(\tilde{k}^2+3\tilde{k}+2)\frac{-u_{i+2}^n+16u_{i+1}^n-30u_i^n+16u_{i-1}^{n-\tilde{k}}-u_{i-2}^{n-\tilde{k}}}{12\Delta x^2}\\ &\quad -(\tilde{k}^2+2\tilde{k})\frac{-u_{i+2}^n+16u_{i+1}^n-30u_i^n+16u_{i-1}^{n-\tilde{k}-1}-u_{i-2}^{n-\tilde{k}-1}}{12\Delta x^2}\\ &\quad +\frac{1}{2}(\tilde{k}^2+\tilde{k})\frac{-u_{i+2}^n+16u_{i+1}^n-30u_i^n+16u_{i-1}^{n-\tilde{k}-2}-u_{i-2}^{n-\tilde{k}-2}}{12\Delta x^2} \qquad (4.5)\end{aligned}$$

For the above scheme, the necessary features that have to be present in the implementation are:

- asynchronous or non-blocking communication calls to overlap computations and communications
- storage of data from multiple time levels

Synchronous
Asynchronous
Poor accuracy
AT: space
Larger message size
AT: time
Increase memory requirement

Figure IV.1. A schematic comparing stencil of different schemes to compute spatial derivative. The point $i \in I_B$ and schemes need function values from neighboring PE. The consequences due to the schemes relative to synchronous case are listed on the right side.

- communication of time stamp of the data to obtain the value of $\tilde{k}_i$
- synchronize communications only when $\tilde{k}_i > L - 1$
- atomicity while accessing data

We now proceed show a communication algorithm that will provide these features.

IV.C.1. Asynchronous communication algorithm

Commonly used synchronous implementations usually carry out communication of data between PEs in a two step process. In the first step, messages are issued to initiate the transfer of data, and in the next step a synchronization process is issued to ensure the completion of communication. As discussed in Ch. I, this synchronization process significantly affects the scalability of applications. The exchange of information is directly done between the compute arrays, which store the function values at all the grid points in a PE.

A simple extension of the above described method to asynchronous communications is shown in Fig. IV.2. Multiple levels are used at the buffer points to accommodate the requirement of different time levels. However, determining the target location of successive messages at the buffer points and keeping track of their time stamp becomes a confusingly difficult task. Also, this method poses atomicity issues while accessing the values of the function. To overcome these issues, we propose a novel algorithm where messages are sent to auxiliary arrays in the target PE, instead of directly sending them to compute arrays. Unlike synchronous communications for which MPI standard provides several options for communication and synchronization calls, there exists only two options for asynchronous communications:

1. Two-sided non-blocking (MPI_Isend/MPI_Irecv)

2. One-sided Remote Memory Access (RMA) (MPI_accumulate)

We discuss the algorithm based on these two methods below.

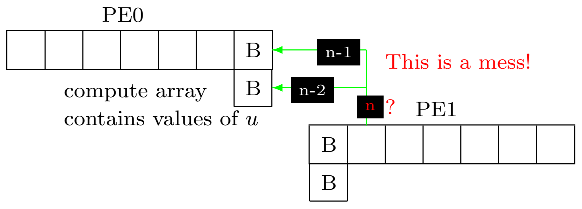


Figure IV.2. A schematic of communications used in common synchronous implementations.

IV.C.1.a. Two-sided non-blocking communications

In the asynchronous communications using two-sided non-blocking calls, we first create two auxiliary buffer arrays, B_{send} and B_{recv}, of size J_1 or J_2 (depending on left or right PE boundary) in one dimension and L time levels in the other dimension, as shown in the schematic in Fig. IV.3. For simplicity we present only the communication of data into left PE boundary in the figure. The purpose of the auxiliary arrays is to send, receive and store data from multiple time levels, while maintaining atomicity. At each time advancement, function at the latest computed time level (n) in the compute array are first copied into one of the levels in B_{send} array at the source PE. Once the copy is complete, the source PE can proceed with the further computations in the compute array. The latest function values are then sent from B_{send} to an equivalent time level in B_{recv} at the target PE using MPI_Isend/MPI_Irecv. No explicit synchronization like MPI_wait_all is imposed in this algorithm, unless $\tilde{k}_i > L - l$. According to the status of communication of each time level, which is determined using MPI_test, necessary function values are copied from B_{recv} into the compute array at target PE to evaluate spatial derivatives at PE boundary points. In the schematic, communication of nth time level is still under progress. Hence, time levels $n-1$ and $n-2$ are copied into compute array.

The communication algorithm using two-sided calls is easy to apply in an asynchronous implementation. However, this algorithm involves several unnecessary overheads, as both source and target PEs are involved in the communication (by posting Isend/Irecv) even when data synchronization is relaxed. An ideal algorithm will be the one where only the source PE carries out asynchronous communications with-

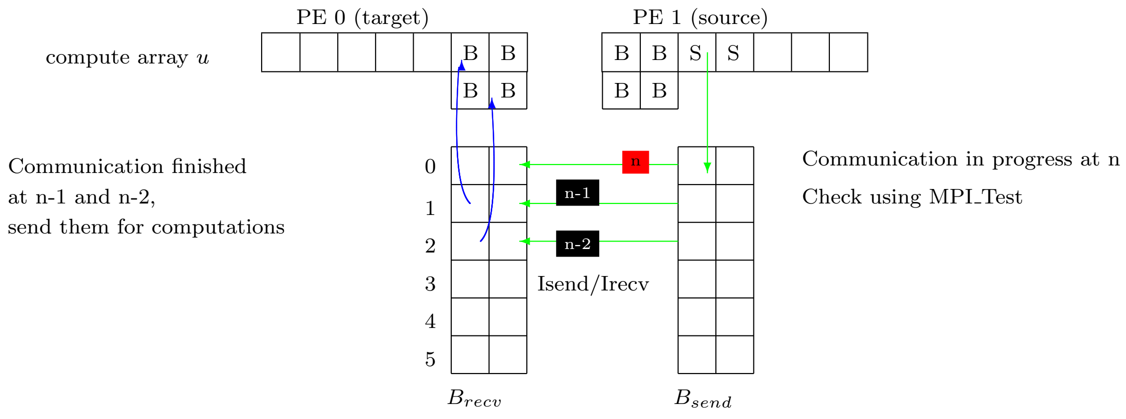


Figure IV.3. An illustration of asynchronous communication algorithm using non-blocking Isend/Irecv. Data are sent from the cells S in PE1 to cells B in PE0.

out *hand shaking* with the target PE. The latest one-sided RMA communications provide such possibility, which is discussed next. Note that these communications are difficult to implement in a solver, and thus, the above described method is also useful in some circumstances.

IV.C.1.b. One-sided RMA communications

Fig. IV.4 illustrates the communication algorithm using one-sided RMA communications. In this method, messages can be sent directly from compute array at source PE, and hence, an auxiliary B_{send} is not necessary. At the target PE, messages are, however, received in B_{recv}. According to RMA communication model, memory location of B_{recv} has to be exposed as a *window* (using MPI_Win_allocate), which will be visible to other PEs over the network to send messages. At each time advancement, function at the latest computed time level (n) is sent to one of time levels in B_{recv} using MPI_Accumulate. Unlike two-sided communications, there is no explicit function that can determine progress of communications in this model. Hence, we create an additional array B_{chg} to track the progress. In the initialization step of the asynchronous implementation (which will be discuss momentarily), arrays B_{recv} and B_{chg} are initialized with the same values in the corresponding time levels. Once the time loop in the implementation begins, messages are sent to successive time levels in B_{recv} in a cyclic ring fashion. In each time advancement, the memory locations of different time levels in both the arrays are compared. If values in the array at a particular time level are unequal, then it indicates completion of a communication. Else, the communication corresponding to that time level is still under progress. In the case of completion of communication, the latest values are copied from B_{recv} to B_{chg}. The time levels necessary for computations are copied from B_{chg}, according to the status of messages in each time level.

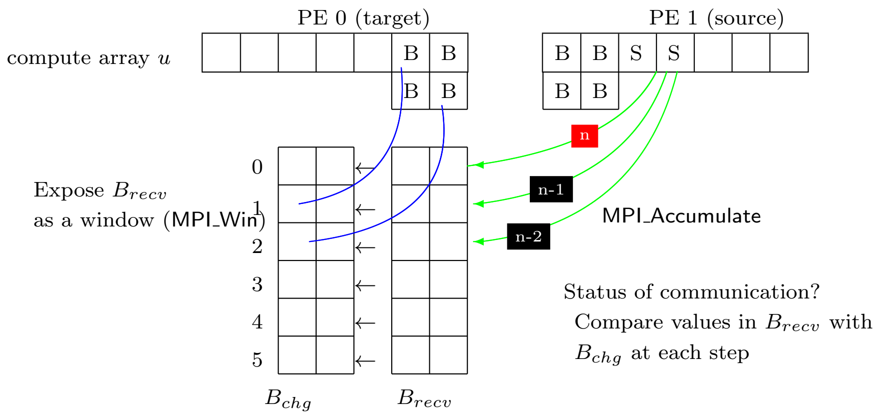

PE 0 (target)
PE 1 (source)
compute array u
B
S
n
n-1
n-2
MPI_Accumulate
Expose B_{recv} as a window (MPI_Win)
0
1
2
3
4
5
Status of communication? Compare values in B_{recv} with B_{chg} at each step
B_{chg}
B_{recv}

Figure IV.4. An illustration of asynchronous communication algorithm using one-sided RMA model. Data are sent from the cells S in PE1 to cells B in PE0.

IV.C.2. Synchronous Implementation (SI)

A simple synchronous implementation starts with simulation initialization process which involves calculating size/limits of arrays for the computational variables, imposing the initial conditions, determining neighboring PEs and setting up communication. For RMA communications, we also setup the buffer grid points as windows using MPI_Win_allocate in this step. Communications are carried out in a synchronous fashion, as discussed in section IV.C.1. Once the initialization process is complete, simulation is advanced in time from the initial conditions according to discretized governing equations. This is done in a time loop that involves calculation of time step, computation of the solution at next time level and communication of the updated values at the boundaries. Following is an illustration of the algorithm.

```
1. Initialize
2. Begin time loop
        2a. Communicate values at time level $n$
        2b. Calculate time step ($\Delta t$)
        2c. Compute $\boldsymbol{U_i}^{n+1} = f\left(\boldsymbol{U_i}^{n}\right)$, $\boldsymbol{i} \in I$
   End time loop
```

IV.C.3. Asynchronous Implementation (AI)

The asynchronous implementation begins with initialization process, similar to the synchronous implementation. Once this is complete, time evolution of the solution is started. For the first L time steps, computations are advanced according to the time loop in synchronous implementation. This is necessary to populate the auxiliary buffers with function values at different time levels in order to maintain accuracy under asynchrony. After the synchronous time loop, the different time levels in the auxiliary arrays are index to obtain the time stamps. When communications are done according to one-sided algorithm, B_{chg} is created and initialized with same values as in B_{recv}, in this step. Up on completion of the indexing process, the solution is further evolved according to asynchronous time loop, as illustrated below. Note that the computations at the interior points in the domain are overlapped with the asynchronous communications. This will lead to significant improvements in scalability and fully exploits the asynchronous nature of the computations.

```
1. Initialize

2. Begin synchronous time loop to populate $B_{recv}$
        2a. Communicate values at time level $n$
        2b. Calculate time step $(\Delta t)$
        2c. Compute $\boldsymbol{U_i}^{n+1} = f\left(\boldsymbol{U_i}^{n}\right)$, $\boldsymbol{i} \in I$
    End synchronous time loop

3. Index values in $B_{recv}$ and create $B_{chg}$

4. Begin asynchronous time loop
        4a. Initiate communication of values at time level $n$
        4b. Calculate time step $(\Delta t)$
        4c. Compute $\boldsymbol{U_i}^{n+1} = f\left(\boldsymbol{U}^{n}\right) \forall \boldsymbol{i} \in I_I$
        4d. Check communication status
        4e. Synchronize if delays $k > L - 1$
        4f. Pick $n - k$ and $n - k - 1$ for computations at $I_B$
        4g. Compute $\boldsymbol{U_i}^{n+1} = f\left(\boldsymbol{U}^{n}, \boldsymbol{U}^{n-k}, \boldsymbol{U}^{n-k-1}\right) \forall \boldsymbol{i} \in I_B$
    End asynchronous time loop
```

IV.D. Summary

In this chapter, we have presented two different implementations to apply asynchronous computing method in solving practical problems. The first implementation simulates delays due to asynchrony using a random number generator. This implementation allows us to evaluate numerical properties of asynchronous method under a wide range of conditions. The second implementation is a highly scalable algorithm to simulation problems on massively parallel computers. We have presented a novel asynchronous communication approach that will relax synchronizations as well as hides the communication with computations at interior points in the domain. This implementation will be used to evaluate the computational performance of the proposed asynchronous method. Results using both the implementations are presented in the next chapter.

CHAPTER V

RESULTS*

In this chapter, we present results from numerical simulations to understand the numerical and computational performance of the asynchronous computing method. The numerical performance of both standard synchronous and asynchrony-tolerant schemes is studied using Implementation I presented in Ch. IV. Later, we demonstrate computational performance of the asynchronous method using Implementation II.

V.A. Numerical performance

V.A.1. Standard schemes

In this section, we now solve the linear advection-diffusion equation using Euler scheme and second order central differences for time and space derivatives, respectively. In Fig. V.1 we show a typical realization of a simulation with $N = 128$, $P = 4$, $L = 2$ and $\{p_0, p_1\} = \{0.3, 0.7\}$ and an initial condition given by a single wave with $\kappa = 2$. In part (a) we show the time evolution of velocity obtained with synchronous (dashed red line) and asynchronous (solid black line) schemes. As expected, the initial wave is convected to the right at a speed of c with a decreasing amplitude due to the effect of viscosity. From this figure the difference between the two solutions is

*Parts of this chapter have been used from "Asynchronous finite-difference schemes for partial differential equations", Diego A. Donzis and Konduri Aditya, Volume 274, 1 October 2014, Pages 370-392, with permission from Elsevier under license number 3842150790850.

hardly seen. However, this is apparent in part (b) where we show the error for both schemes. As expected the asynchronous solution presents higher errors. We also see that errors seem to be larger at PE boundaries (vertical dashed lines) and spread to inner regions of the domain as time proceeds. This is not surprising as the evolution of error is, for linear systems, governed by the same PDE as u itself.

We note that since these errors appear at PE boundaries and are typically localized in space, the effect on statistical features of the solution may be small. For some applications, these localized (and bounded) errors may not affect specific quantities of interest. For example, in high-Reynolds number turbulence, a great deal of interest is in statistical features of the flows due to the tremendous complexity of instantaneous fields. A few examples, include the calculation of the mean turbulent kinetic energy and dissipation, scaling of spectra and structure functions all of which are typically insensitive to (numerical) perturbations at scales smaller than the so-called Kolmogorov scale (e.g. Watanabe & Gotoh, 2007; Ishihara *et al.*, 2009). High-order moments of velocity gradients, on the other hand, could become a challenge (Donzis *et al.*, 2008). As we will show momentarily, however, these errors can be made smaller with finer grids with specifically designed schemes. Numerical experiments related to these issues will be presented in section V.A.2.

We now turn to a statistical description of the error with more realistic initial conditions. The initial condition comprises a range of wavenumbers (typically spanning a decade or so) with random phases and for simplicity only two possible delays are allowed ($L = 2$): $\tilde{k}_{i+1} = 0$ and 1. Note that in this case the probabilities are given by $\{p_0, p_1\}$ where $p_1 = 1 - p_0$. Thus, p_0 is enough to characterize the simulations.

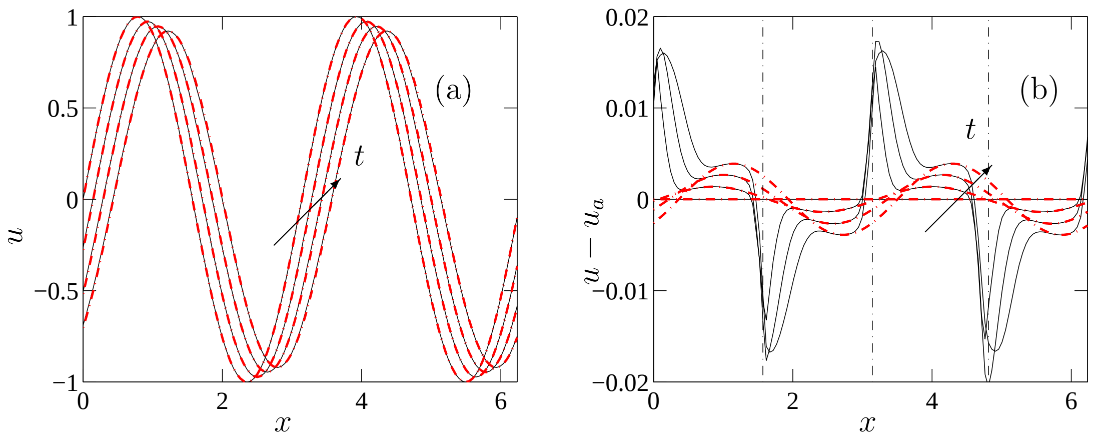


Figure V.1. Typical time evolution of the numerical solution of the advection-diffusion equation using synchronous (dashed red lines) and asynchronous (solid black lines) schemes. (a) The velocity field. (b) Error $E_i^n = u_i^n - u_a(x_i, t_n)$. Vertical dash-dotted lines correspond to PE boundaries. Simulation parameters: $N = 128$, $P = 4$, $L = 2$, with $\{p_0, p_1\} = \{0.3, 0.7\}$ for the asynchronous scheme.

The ensemble and space average error $\langle\overline{|E|}\rangle$ is shown in Fig. V.2 as a function of resolution N for fixed r_α, and different set of the probabilities. The error was calculated at a normalized time of $tc/l = 0.08$ (l is the length of the domain). We have verified that the conclusions below are the same for longer times as well.

The synchronous (deterministic) case corresponds to $p_0 = 1$ (solid red line) which is seen to be second order. When asynchrony is introduced we have $p_0 < 1$ and we clearly see that the formal order of accuracy drops to one as predicted by the theory for constant values of P and r_α (Eq. (2.42)). For a given resolution, we can also see that the numerical value of the error increases as p_0 decreases, that is as the probability of having delayed values increases. The inset shows the same error as a function of p_0 (squares) at the largest resolution available. The good agreement between the data and the best linear fit (solid line through data points) support a linear relation between $\langle\overline{|E|}\rangle$ and p_0. This is consistent with Eq. (2.41) since the mean delay in the present case ($L = 2$) is simply $\overline{\tilde{k}} = 1 - p_0$.

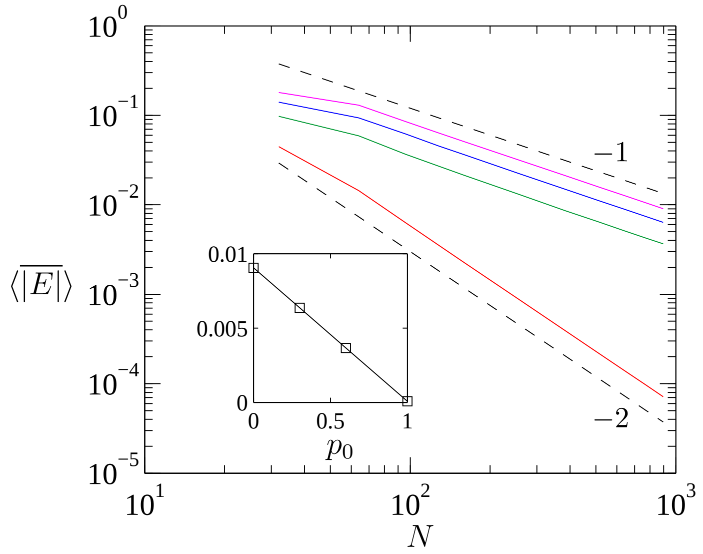


Figure V.2. Average error for the advection-diffusion equation with constant $r_\alpha = 0.1$. Different lines correspond to $p_0 = 1$ (red), 0.6 (green), 0.3 (blue), and 0.0 (magenta). Dashed lines with slopes -2 and -1 are included for reference. Inset: $\langle\overline{|E|}\rangle$ versus p_0 for the largest resolution in the main figure. Symbols are from simulations and the solid line is a linear best fit.

To test the more general claim in Eq. (2.41) that the error is linearly dependent on the average value of the delay $\tilde{k}$, we have performed simulations with different values of L and probability sets with the same mean value of $\overline{\tilde{k}}$. A set of results is shown in Fig. V.3 with different colors for different values of $\overline{\tilde{k}}$. The dependence on $\overline{\tilde{k}}$ instead of L or $\{p_0, p_1, \dots\}$ is seen by observing the collapse of, for example, the red squares ($L = 2$, $\{0.3, 0.7\}$) and the red solid line ($L = 4$, $\{0.6, 0.2, 0.1, 0.1\}$) both with $\overline{\tilde{k}} = 0.7$. The same observation can be made for the other set of simulations at $\overline{\tilde{k}} = 1$ (magenta) and 3 (blue). The inset shows a larger set of simulations with a wide range of values of $\overline{\tilde{k}}$ (with different combinations of L and probabilities). The solid line is a best linear fit through the data points essentially confirming the linear dependence in Eq. (2.41).

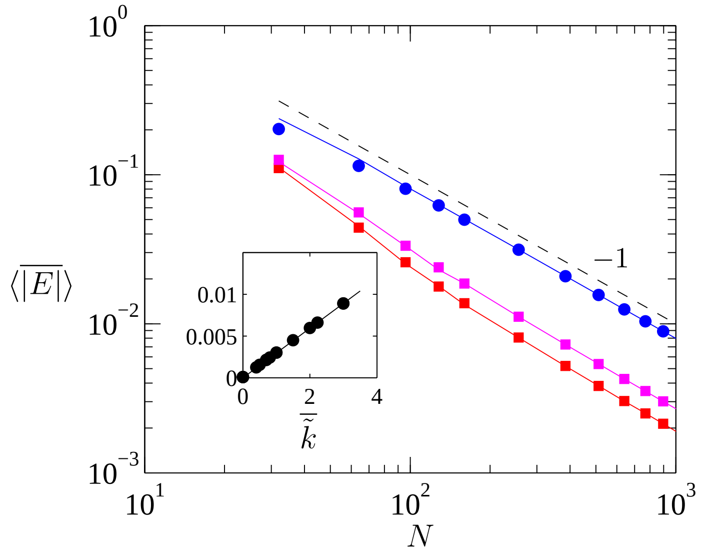


Figure V.3. Effect of mean value of delay on the average error for the advection-diffusion equation with constant $r_\alpha = 0.1$ and $P = 32$. Different colors correspond to different values of the average delay: $\overline{\tilde{k}} = 0.7$ (red), 1.0 (magenta) and 3 (blue). Squares, solid lines and circles correspond to $L = 2$, 4 and 8 respectively. Dashed line with slope -1 is included for reference. Inset: $\langle\overline{|E|}\rangle$ as a function of $\overline{\tilde{k}}$ for the largest resolution in the main figure with $P = 32$.

Since in future computing systems, the number of PEs may be extremely large, it is also important to understand the behavior with P. Eq. (2.42) predicts that the average error is proportional to P/N. Thus, in an exercise of so-called strong scaling (that is, keeping the problem size, N, constant and progressively increasing the number of PEs, P) the average error $\langle\overline{|E|}\rangle$ is first order in Δx and grows linearly with P. These two predictions are indeed supported by our simulations. In Fig. V.4(a) we clearly see asymptotic first order accuracy. For small PE counts (red line), we can see the transition from second order to first order which is also predicted by Eq. (2.40) . As P increases, the numerical value of the error increases for fixed N. In the inset, we show again $\langle|\overline{E}|\rangle$ as a function of P at the largest resolution in the main figure ($N = 1024$). The linear increase with P is clearly seen as symbols appear to be very well aligned with the best linear fit shown as the solid line.

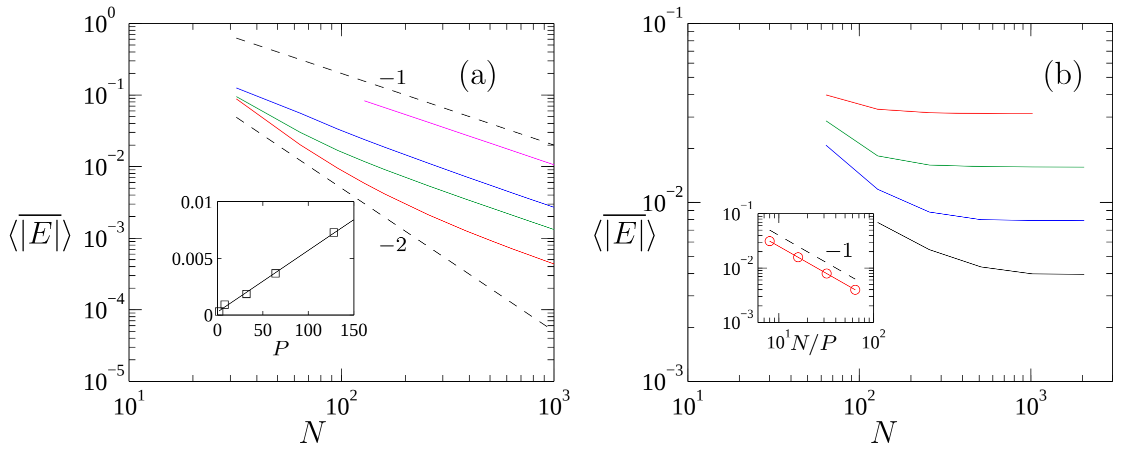


Figure V.4. Effect of number of PEs on the average error for the advection-diffusion equation with constant $r_\alpha = 0.1$. (a) Cases with constant P. Different lines correspond to $P = 2$ (red), 8 (green), 32 (blue) and 128 (magenta). Inset: $\langle|\overline{E}|\rangle$ versus P for the largest resolution ($N = 1024$) in the main figure. (b) Cases with constant P/N. Different lines correspond to $P/N = 1/64$ (black), $1/32$ (blue), $1/16$ (green) and $1/8$ (red). Inset: $\langle|\overline{E}|\rangle$ versus N/P for the largest resolution ($N = 1024$) in the main figure. Dashed lines with slope -2 and -1 are included for reference.

As discussed in section II.C.2, in a number of applications the number of PEs is increased proportionally to the work load (weak scaling). In this case Eq. (2.42) shows that the error is constant. This result is also supported by our simulations in Fig. V.4(b) where grid refinement does not lead to any decrease in the average error.

The inset shows that the asymptotic value of $\langle\overline{|E|}\rangle$ increase linearly with P/N as predicted by Eq. (2.42).

There is another important consequence of using common finite differences in an asynchronous fashion. One of the objectives of using asynchronous schemes is to avoid global synchronizations. However, in many realistic simulations where diffusion effects are small, the time step is determined by a convective stability condition which is computing from the maximum velocity in the entire domain. In such a case, obtaining a consistent time step across all PEs, requires a global synchronization. If a good estimate of the maximum velocity in the problem is known a priori, however, then a constant Δt can be set based on this knowledge to satisfy the convective condition everywhere. While this avoids a global synchronization, the accuracy of the scheme is strongly modified. This can be seen from Eq. (2.48) and Table II.2 which, with $D = 2$ at constant P, results in an average error scaling as $1/\Delta x^{D-1} = 1/\Delta x$. These predictions are in fact consistent with the numerical data shown in Fig. V.5(a) where convergence transitions from Δx^2 to $1/\Delta x$ for asynchronous schemes (blue lines). For synchronous schemes, that is $p_0 = 1.0$, the transition is from Δx^2 to a constant (red lines) since the truncation error stemming from the time discretization is proportional to Δt and thus remains constant as the space discretization error decreases with N. The effect of decreasing Δt in both cases is, as expected, a reduced numerical value of the error.

Even if a global maximum velocity can be determined efficiently, the use of a convective condition (that is $r_c = c\Delta t/\Delta x$ kept constant in our linear case) results in higher derivatives providing divergent terms in the truncation error. This is also

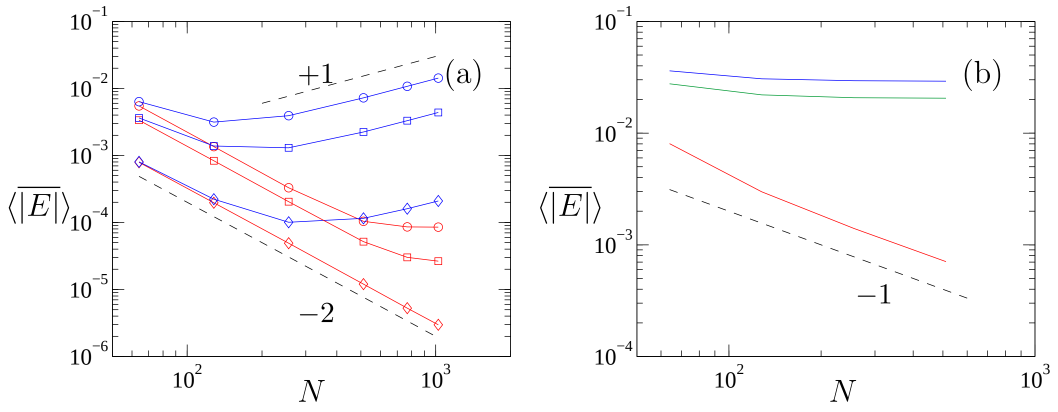


Figure V.5. Convergence properties with different relations between time step size and grid spacing for the advection-diffusion equation. (a) Cases with constant Δt. Different lines correspond to $\Delta t = 1 \times 10^{-5}$ (circles), 5×10^{-6} (squares), and 1×10^{-6} (diamonds). (Note that time steps are small because they need to be chosen such that the stability condition is satisfied at the largest resolution.) Red and blue lines correspond to $p_0 = 1.0$ (synchronous) and 0.3, respectively. (b) Cases with constant $r_c = c\Delta t/\Delta x$. Red, green and blue lines correspond to $p_0 = 1.0$ (synchronous), 0.3, and 0.0, respectively. Dashed lines in both panels are reference power laws with slope noted next to the corresponding line.

seen from the last two rows in Table II.2. For the case of the advection-diffusion equation ($D = 2$) with a convective condition ($q = 1$ in the table) and constant P, the average error is also constant as the grid is refined. Fig. V.5(b) shows again this theoretical result to be consistent with the numerical data. The figure also shows the standard synchronous scheme which is seen to be first order. This is expected since the time discretization which is $O(\Delta t)$, becomes $O(\Delta x)$ when r_c is kept constant.

V.A.2. Asynchrony-tolerant schemes

As shown in previous sections, while common finite differences used in an asynchronous fashion retain stability characteristics, its accuracy can be degraded significantly making these schemes potentially unusable for real applications. Thus, alternative strategies are needed to make asynchronous approaches feasible for accurate solutions of general PDEs.

Since leading order terms in the truncation error due to asynchrony are proportional to Δt (see e.g. Eq. (2.48)), the simplest strategy is to select the time step according to $\Delta t \sim \Delta x^q$ with appropriate values of q. For $D = 2$ (i.e. heat or advection-diffusion equations) a value of $q = 3$ will recover the original average truncation error of Δx^2 when the number of PEs is kept constant according to Table II.2. This is indeed observed from the numerical data presented in Fig. V.6(a). However, as noted in section II.C.3, this may render the simulations prohibitively expensive as time step size decreases rapidly with increasing space resolution, especially when high-order schemes are utilized. Thus, we use asynchrony-tolerant schemes which retain a desired order of accuracy without the need to decrease the time step size with a steep power law on the grid spacing.

In Fig. V.7, we show typical results from the simulations using an asymmetric stencil asynchrony-tolerant scheme (see Eq. (5.1)) at PE boundary points which is, with $r = 2$, first order accurate in space.

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+2}^{\tilde{n}} - u_{i+1}^{\tilde{n}} - u_i^n + u_{i-1}^n}{2\Delta x^2} + \mathcal{O}(k\Delta t, \Delta x, k\Delta t/\Delta x) \tag{5.1}$$

Comparison of Fig. V.7(b) and Fig. V.1(b) reveals smaller errors for the new scheme. As predicted by our analysis, the scheme recovers second-order convergence asymptotically even under asynchronous conditions as seen in Fig. V.6(b). At lower resolutions, a dependence on the probability p_0 is seen but weakens as N increases. These cases were run at $r_\alpha = 0.1$ and r_c in the range $0.035 - 0.245$ which were found to provide a stable solution.

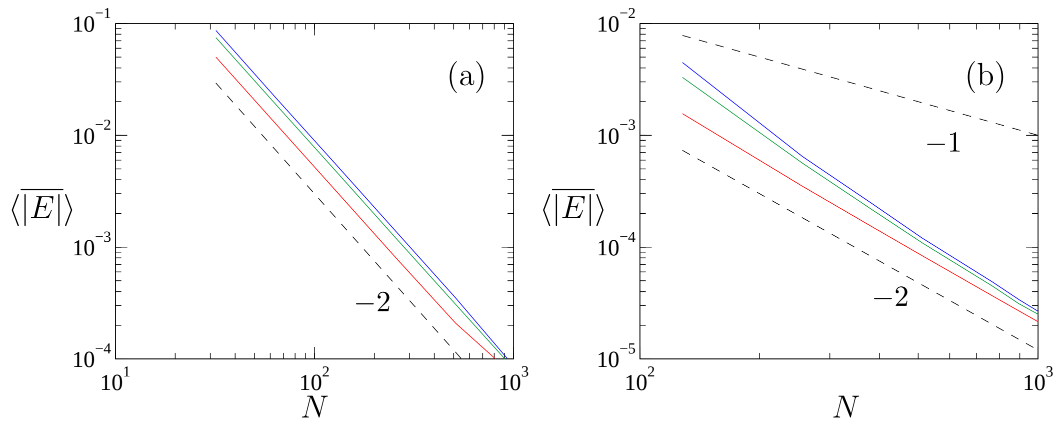


Figure V.6. Recovery of convergence properties for the advection-diffusion equation. (a) Standard scheme Eq. (4.4) with time step determined as $\Delta t \sim \Delta x^3$. (b) Scheme using Eq. (5.1) with a fixed r_α of 0.1. Dashed lines in both panels are reference power laws with slope noted next to the corresponding line. Red, green and blue lines correspond to $p_0 = 1.0$ (synchronous), 0.3, and 0.0, respectively.

To verify the accuracy of higher order schemes, we use simulations of linear advection-diffusion equation approximated with Adams-Bashforth scheme for the time derivative and fourth order central difference and fourth order symmetric stencil symmetric coefficients asynchrony-tolerant schemes for space derivatives. In Fig. V.8, we present convergence graphs in the cases of both strong and weak scaling of the problem. Different lines in the graphs are for different probabilities of asynchrony. As predicted from the theory, all the cases converge according to a slope of -4, which shows that the schemes are fourth order accurate in space. It is interesting to note that there is hardly any visible distinction for different probabilities of asynchrony. This is because the errors due to asynchrony is bounded by a value of $\mathcal{O}(\Delta x^4)$.

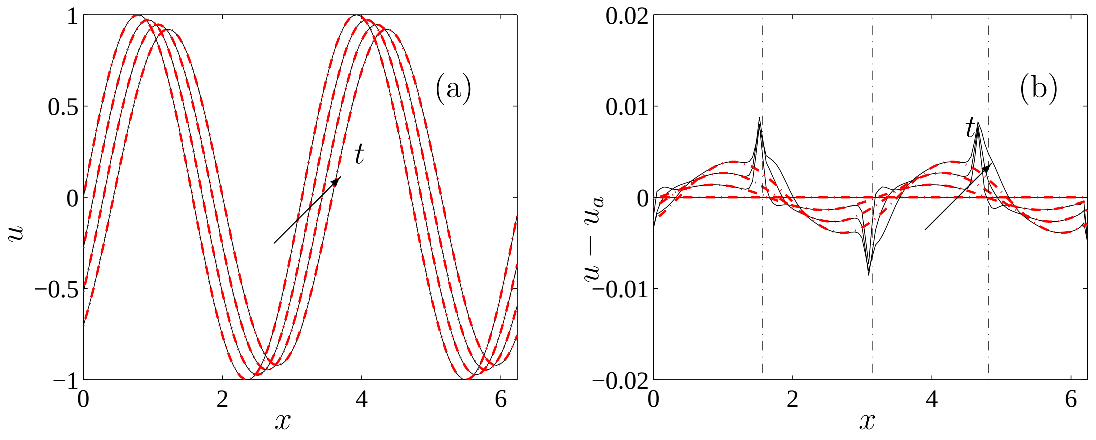


Figure V.7. Typical time evolution of the numerical solution of the advection-diffusion equation using the standard synchronous scheme Eq. (4.4) with $\tilde{n} = n$ (dashed red lines) and the new asynchronous scheme Eq. (5.1) (solid black lines). (a) The velocity field. (b) Error $E_i^n = u_i^n - u_a(x_i, t_n)$. Vertical dash-dotted lines correspond to PE boundaries. Simulation parameters: $N = 128$, $P = 4$, $L = 2$, with $\{p_0, p_1\} = \{0.3, 0.7\}$ for the asynchronous scheme.

V.B. Nonlinear equation

Although the theoretical predictions were done for the advection-diffusion equation, there is obvious interest also in its non-linear version, that is Burgers' equation. This is so because of this equation resembles a one-dimensional (and more analytically amenable) version of the full equations describing the motion of fluid flows in important applications such as high-Reynolds numbers turbulence. In Fig. V.9(a) we show an example of a scheme like Eq. (4.4). Specifically,

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + u_i^n \frac{u_{i+1}^{\tilde{n}} - u_{i-1}^n}{2\Delta x} = \alpha \frac{u_{i+1}^{\tilde{n}} - 2u_i^n + u_{i-1}^n}{\Delta x^2}. \tag{5.2}$$

In part (a) of the figure we show results with $r_\alpha = 0.1$ and initial conditions given by Eq. (4.2) with different ranges of wavenumbers and random phases. Conclusions below are insensitive to those parameters in the initial conditions as long as the

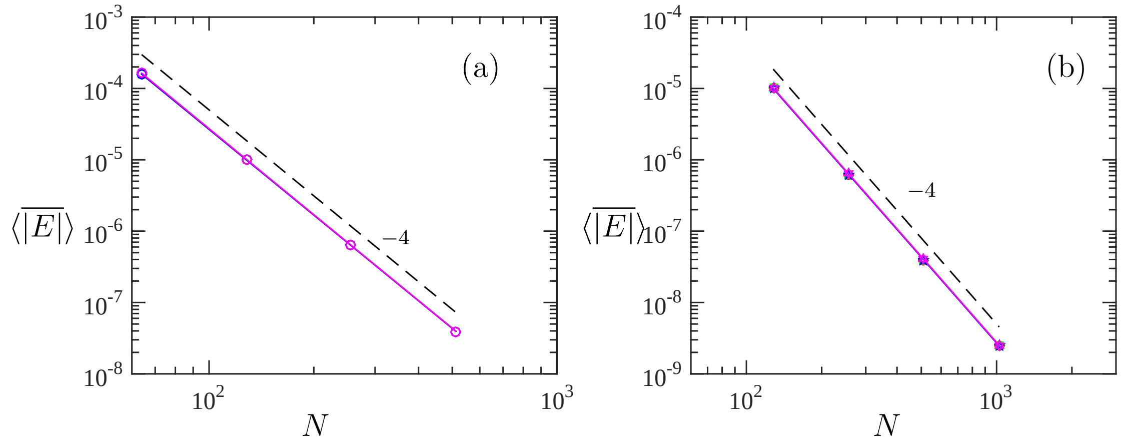


Figure V.8. Effect of number of PEs on the average error for the advection-diffusion equation with constant $r_\alpha = 0.1$. (a) Cases with $P = 16$. Different lines correspond to $p_0 = 1.0$ (red), 0.6 (green), 0.2 (blue) and 0.0 (magenta). (b) Cases with $P/N = 1/16$. Different lines correspond to $p_0 = 1.0$ (red), 0.6 (green), 0.2 (blue) and 0.0 (magenta). Dashed line with slope -4 is included for reference.

scheme remains stable. Just as in the linear case, we observe that the accuracy drops from second to first order and that the effect of decreasing p_0 is to increase the error linearly (inset). If the new scheme Eq. (5.1) is applied to the non-linear equation, the scheme is now

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + u_i^n \frac{u_{i+1}^{\tilde{n}} - u_{i-1}^n}{2\Delta x} = \alpha \frac{u_{i+2}^{\tilde{n}} - u_{i+1}^{\tilde{n}} - u_i^n + u_{i-1}^n}{2\Delta x^2}. \tag{5.3}$$

A typical result using this scheme is shown in Fig. V.9(b). All the conclusions we arrived at with the linear equation appear to apply equally well in the non-linear case. However, more thorough testing as well as a deeper fundamental understanding of the non-linear case is required to make a more generally valid claim. This is especially so for small values of α, where increasingly large gradients (and numerical instabilities) are known to appear.

As mentioned in section V.A, since errors may be localized in space, some sta-

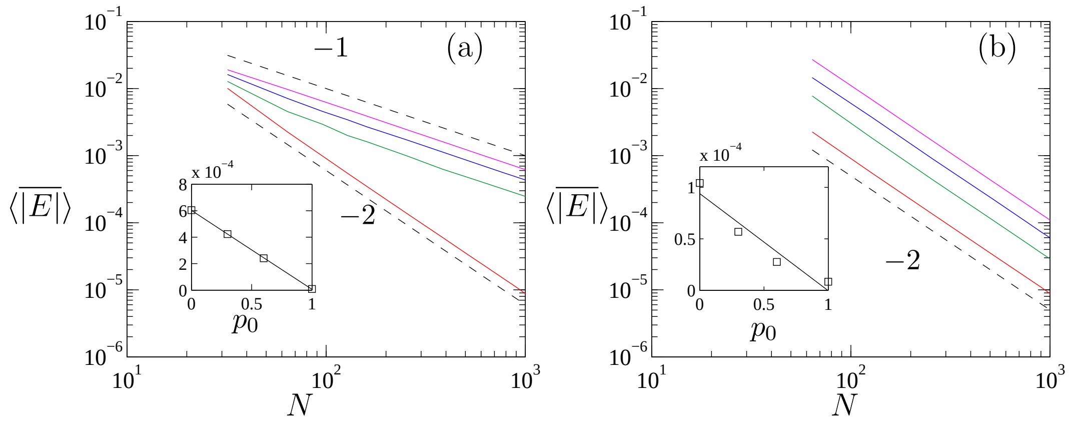


Figure V.9. Convergence for non-linear Burgers equation. (a) Results using Eq. (5.2). (b) Results using Eq. (5.3). Different lines correspond to $p_0 = 1.0$ (red), 0.6 (green), 0.3 (blue), and 0.0 (magenta). Dashed lines are reference power laws with the slope noted next to the corresponding line. Inset: $\langle\overline{|E|}\rangle$ as a function of p_0 for the highest resolution in the main figure and $P = 32$ and $L = 2$. Solid line is a best linear fit.

tistical features of the solution may be insensitive to the numerical perturbations due to asynchrony. An example of this, is shown in Fig. V.10 where we can see the decay of the space-averaged kinetic energy $K = \langle u^2\rangle/2$ and its dissipation rate $\epsilon = \alpha\langle(\partial u/\partial x)^2\rangle$ as a function of time. A grid convergence study for the synchronous schemes (also seen in Fig. V.10) show that K becomes grid-independent at $N = 96$ — that is, results are indistinguishable from results at $N = 128$ and higher. The same behavior is observed for asynchronous computations (dashed lines) which practically overlap with synchronous results when the grid is fine enough for (synchronous) results to have converged.

The decay of the dissipation rate is shown in part (b). As expected this is more sensitive to resolution than K because ϵ depends on gradients. Thus, it is not surprising that larger errors appear. Still grid-independent results are also attained at

N larger than 96. Again, asynchronous and synchronous results are indistinguishable from each other in the figure.

We then conclude that, in this case, the decay of both K and ϵ are virtually unaffected by asynchrony for grid-converged simulations. Thus, if the interest is, for example, in decay rates, then asynchronous computations would provide results which are as accurate as those using synchronous schemes. Obviously, extending this conclusion to three-dimensional, high-Reynolds number turbulence will require careful examination of numerical data from asynchronous simulations. In this endeavor it would also be important to study the behavior of other methods more suitable for the non-linear Burgers equation when asynchrony is present.

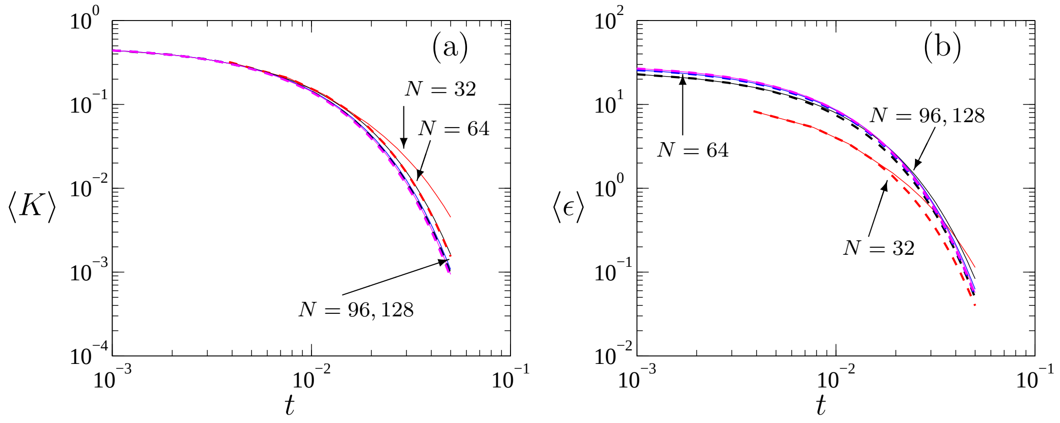


Figure V.10. Time evolution of overall (a) kinetic energy, $\langle K \rangle$ and (b) dissipation rate, $\langle \epsilon \rangle$ of the flow at different grid resolutions. Red, black, blue, and magenta correspond to $N = 32$, 64, 96, and 128 respectively. Synchronous (solid lines) and asynchronous results (dashed lines) were obtained with Eq. (5.2) and Eq. (5.3). Initial conditions: Eq. (4.2) with wavenumbers $\kappa = 6, 7, 8$. Other simulation parameters: $P = 16$, $L = 4$, $\{p_0, p_1, p_2, p_3\} = \{0.4, 0.2, 0.3, 0.1\}$.

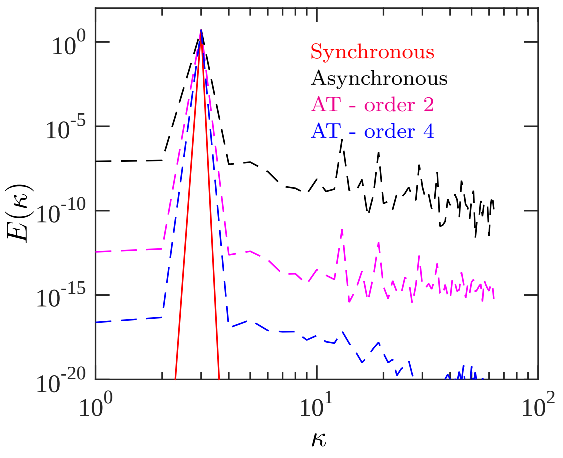


Figure V.11. Energy spectrum of the solution evolved according to the diffusion equation. Simulation parameters are $p = 16$, $L = 2$, $\kappa = 3$. Different lines in the graph are with different schemes used at PE boundary points.

V.C. Spectral space

In this section, we study the effects of asynchronous computing method in spectral or wavenumber space. Fig. V.11 illustrates the energy spectrum of the solution evolved according to the diffusion equation. The initial condition we have used in this simulations is single sine wave with $\kappa = 3$. As the equation is linear, one would no additional numbers in the spectrum. This is seen with synchronous solution (red line in the figure) which has energy only in $\kappa = 3$. On the other hand, when the solution is computed asynchronously, we observe that additional wave numbers with low energy appear in the spectrum. This shows that the random switching of time levels due to asynchrony results is a nonlinear process. It is interesting to see that when asynchrony-tolerant schemes are used, the energy in the additional wavenumbers due to asynchrony is significantly reduced.

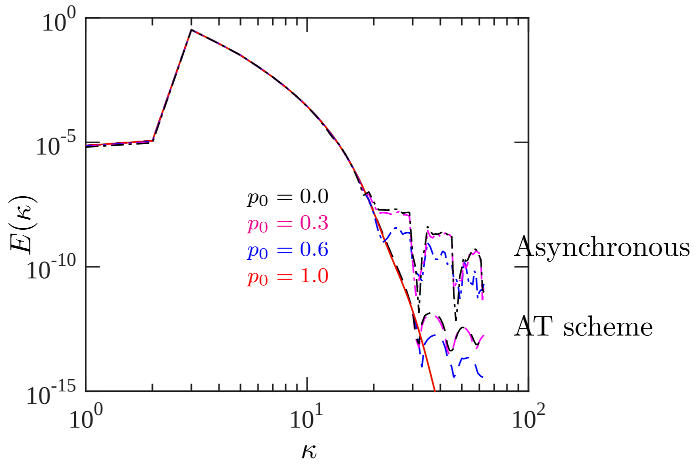


Figure V.12. Energy spectrum of the solution evolved according to the nonlinear Burgers' equation. Simulation parameters are $p = 16$, $L = 2$, κ =multi-scale. Continuous, dash-dotted and dashed lines are for synchronous, asynchronous and asynchrony-tolerant schemes, respectively.

We now understand the effect on asynchrony in a nonlinear process. The result shown in Fig. V.12 are obtained from the simulations of the Burgers' equation using second order synchronous and asynchrony-tolerant schemes. A multiscale spectrum has been used as an initial condition. Because of the nonlinear nature of the equation, additional wavenumber are created as the solution evolves in time. In the figure we see that the energy is distributed across a wide range of wavenumbers. At high wavenumbers the energy is expected to decay exponentially, which is observed in the synchronous case. In the presence of asynchrony, the spectrum deviates from the exponential decay at high wavenumbers. The behavior in these wavenumbers is similar to the observations in linear case. The use of asynchrony-tolerant schemes, again, minimize the affect of asynchrony.

V.D. Computational performance

In this section we present results from the simulations of a 1D advection-diffusion equation, developed using Implementation II described in Ch. IV. The computational effort per PE in these simulations is very similar to what one would expect in massively parallel simulations. An important parameter in asynchronous simulations is the distribution of $\tilde{k}$, which determines the characteristics of delay in communications. Fig. V.13 shows the distribution for two different MPI communication implementations (MPI-2 and foMPI (fast one-sided MPI, Gerstenberger *et al.* (2013))). They have been obtained using a spatial average among all the PEs and an ensemble average over 10 runs. A buffer length $L = 20$ has been used in the simulations, which means that synchronization of communication is invoked only when $\tilde{k} > 20$. In a simulation, it is very likely that computations in a PE are ahead of its neighbors. In such a case, the PE will receive delayed values with $\tilde{k} > 0$ into its buffer. At the same time, the latest values received by its neighbors can be from advanced time levels with respect to their n, i.e., $\tilde{k} < 0$. From the graph two major observations can be made. First, foMPI implementation provides $\tilde{k}$ close to zero. Second, a non-zero probability of $\tilde{k} = 20$ indicating explicit synchronization of communication during the simulations. These are due to faster communication with foMPI that will result in better performance and more accurate solution.

To compare the overall performance of the algorithms, we present the strong-scaling graph in Fig. V.14(a). Clearly, asynchronous method scales well much beyond the range in which synchronous counterpart scales. Also, the scaling is nearly linear due to the absence of any explicit synchronization during the simulation, which is

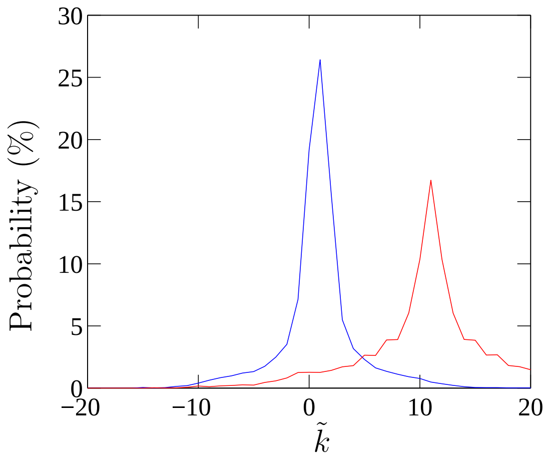


Figure V.13. Probability distribution of $\tilde{k}$ in a simulation. Two lines correspond to communication using MPI-2 (red) and foMPI (blue) implementations.

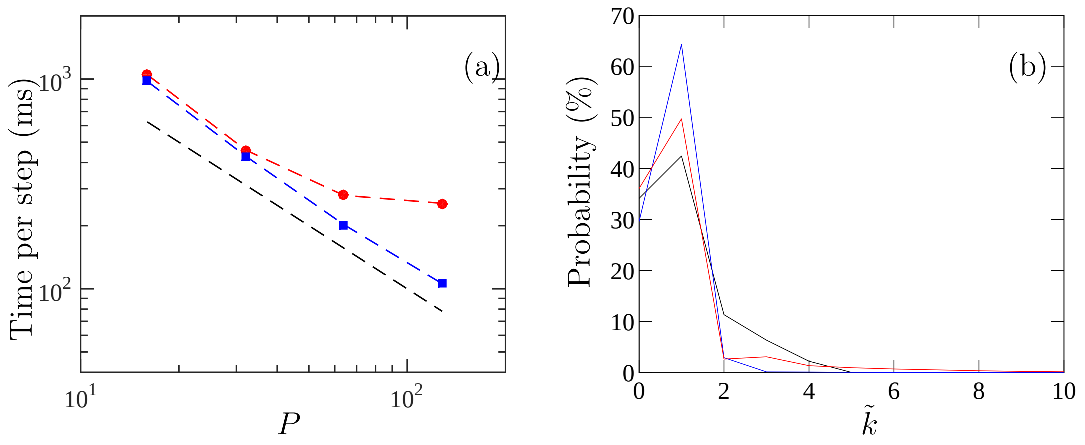


Figure V.14. (a) Graph of strong scaling on Hopper. Synchronous (red), asynchronous (blue), linear scaling (black). (b) Probability distribution of $\tilde{k}$ for asynchronous simulations in figure (a). $P = 16$ (blue), $P = 64$ (red), $P = 128$ (black).

evident from Fig. V.14(b).

V.E. Summary

In this chapter, we have first presented results on the numerical performance of both synchronous and asynchrony-tolerant schemes. The results from the simulations are very well in agreement with theoretical predictions made in Ch. II and III. The scaling of the average error with simulations parameters like number of PEs, statistics of delay have also been presented. We have also studied the effect of asynchrony in spectral space. We show that asynchrony creates additional scales with low energy content. With simulations using Implementations II, we have measured typical delays experienced on actual machine. We have shown the excellent scalability of the asynchronous computing method.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

VI.A. Conclusion

Due to the massive levels of parallelism already available on petascale systems, communication between PEs presents significant challenges in practical applications for sustained performance at scale. In future exascale systems, parallelism is expected to increase further in a significant way. Communication, global synchronizations as well as minor load imbalances could represent major bottlenecks to scalability on future systems.

In this work, we have proposed a highly scalable asynchronous computing method based on finite difference schemes to solve PDEs at extreme scale such that all synchronizations and communication overheads are avoided. Not only this allows for overlap of computation and communication but it also relaxes expensive synchronizations due to small but common delays across PEs which are already present in current systems due to e.g. system noise.

We first investigated the effect of asynchrony on numerical properties of standard synchronous schemes. We showed that these schemes remain stable when the value of the function at one or more grid points correspond to time steps previous to the most current one. This property was shown using bounds on the infinity-norm of the matrix representing the evolution of the solution at every step. The use of a matrix method to analyze stability is necessary because of the random nature of the delays as well as the localized nature of these delays (at PEs boundaries) which renders a classical von Neumann analysis inappropriate in this case. Our analysis can only give sufficient conditions, though, which in some cases may be too restrictive. We have also analyzed the issues posed by asynchrony on consistency and accuracy of standard schemes. We have shown that when used asynchronously widely used finite differences may present consistency issues, which can, in principle, be mitigated by the way in which time step size and grid spacings are reduced simultaneously. By studying the details of the truncation error, it is clear that a zeroth-order term always appears. However, these errors will only be present at PE boundaries. Due to the random nature of the delays expected at PE boundaries, a statistical approach has been used to study the convergence properties of these asynchronous schemes. In particular we used the average of the error and the absolute value of the error (corresponding to the 1-norm of the error) across the domain. We showed that this error always drops to first order when asynchrony is present regardless of the order of the original scheme. Furthermore, we found that the error is also proportional to the number of PEs (P) and the mean delay $\overline{|\tilde{k}|}$, that is $\langle\overline{|E|}\rangle \sim \overline{|\tilde{k}|}P\Delta x$. This is both an interesting and important result. It states that the error of the solution depends not only on grid spacing but also on the characteristics of the system it is run on. At extremely large levels of parallelism due to the enormous challenges in data movement it is indeed expected that scalable codes are able to trade some accuracy for computational performance. This scaling of the error has been verified by thorough numerical experiments.

Since first-order methods are rarely of practical use, we have shown that it is possible to construct new asynchrony-tolerant schemes robust to asynchrony. By using larger stencils, one can mitigate the errors incurred by delays in messages and recover higher orders of accuracy. We have discussed the consequences of increasing the stencil size in space and time on the computational performance. Using Taylor series and the method of undetermined coefficients, we have presented a general methodology to derive asynchrony-tolerant schemes of arbitrary accuracy. With careful selection of points and time level in a scheme, it is possible to reduce the expected increase of stencil size to achieve higher orders of accuracy. A simple classification has been provided for the schemes based on their stencil layout and nature of coefficients. We have derived sample schemes that are representative of different classes in the classification. We have analyzed the average error in a numerical method which uses both standard and asynchrony-tolerant schemes in different regions of computational domain. It has been found that the average error scales linearly with P when asynchrony component dominates. Unlike standard schemes, the average accuracy with asynchrony-tolerant schemes can depend on mean as well as higher order moments of delay. Numerical experiments have been carried out using the linear advection-diffusion and the non-linear Burgers equation to validate the theoretical work. Results show an excellent agreement with the theoretical predictions.

To exploit the potential of asynchronous computing method to eliminate virtually all synchronizations and communication overheads, we have developed a novel implementation algorithm for practical applications. We have utilized modern remote memory access schemes that have been shown to provide significant speedup on modern supercomputers, to efficiently implement communications suitable for asynchronous-tolerant schemes. Results from 1D simulations have shown nearly linear scalability and demonstrated that the proposed method has a potential to provide extreme scalability and may be a viable path towards the development of future exascale applications.

VI.B. Future work

In this section, we discuss some possible research directions that can be continued from the current work. We divide this section into numerical and computational

VI.B.1. Numerical

- Our work has been based on finite difference method, which in principle can be applied to other methods like finite volume and finite element. Also, we have carried out the analysis using 1D linear equations. This has to be extended to higher dimensions for practical applications.

- It is important to devise a frame work to understand the stability of asynchrony-tolerant schemes as it will expand the range of conditions in which one can compute solutions with greater confidence.

- The properties of asynchrony-tolerant schemes, in terms of spectral resolution, dissipative and dispersive error, etc. have to be studied. For complex problems it may be necessary to derive these schemes which possess features like energy conserving, flux limiting, etc.

- Using the concepts in uncertainty quantification, one can develop a model to predict the distribution of error due asynchrony with inputs on the distribution of delays.

- The numerical performance of asynchronous computing method should be tested with more complex equations like the Navier-Stokes equations and in complex conditions like the turbulence phenomena.

VI.B.2. Computational

- In this work, we have demonstrated the computational performance of asynchronous computing method using a simple 1D problem. This has to be extended to solve complex 3D problems on massively parallel machines.
- Implementation of the method has to be extended to heterogeneous architectures which use accelerators like GPU or co-processors.
- The implementation of the asynchronous method should done on more high-level implementations (e.g. STAPL), which will facilitate rapid development, experimentations and portability of asynchronous solvers.

REFERENCES

Amitai, D, Averbuch, A, Israeli, M & Itzikowitz, S 1996 On parallel asynchronous high-order solutions of parabolic PDEs. *Numer. Algorithms* **12**, 159–192.

Amitai, D., Averbuch, A., Itzikowitz, S. & Israeli, M. 1992 Parallel adaptive and time-stabilizing schemes for constant-coefficient parabolic PDEs. *Comput. Math. Appl.* **24**, 33–53.

Amitai, Dganit, Averbuch, Amir, Itzikowitz, Samuel & Turkel, Eli 1994 Asynchronous and corrected-asynchronous finite difference solutions of pdes on mimd multiprocessors. *Numer. Algorithms* **6** (2), 275–296.

Bertsekas, D. P. & Tsitsiklis, J. N. 1989 *Parallel and distributed computation: numerical methods.*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Dongarra, Jack, Beckman, Pete, Moore, Terry, Aerts, Patrick, Aloisio, Giovanni, Andre, Jean-Claude, Barkai, David, Berthou, Jean-Yves, Boku, Taisuke, Braunschweig, Bertrand, Cappello, Franck, Chapman, Barbara, Chi, Xuebin, Choudhary, Alok, Dosanjh, Sudip, Dunning, Thom, Fiore, Sandro, Geist, Al, Gropp, Bill, Harrison, Robert, Hereld, Mark, Heroux, Michael, Hoisie, Adolfy, Hotta, Koh, Jin, Zhong, Ishikawa, Yutaka, Johnson, Fred, Kale, Sanjay, Kenway, Richard, Keyes, David, Kramer, Bill, Labarta, Jesus, Lichnewsky, Alain, Lippert, Thomas, Lucas, Bob, Maccabe, Barney, Matsuoka, Satoshi, Messina, Paul, Michielse, Peter, Mohr, Bernd, Mueller, Matthias S., Nagel, Wolfgang E., Nakashima, Hiroshi, Papka, Michael E., Reed, Dan, Sato, Mitsuhisa, Seidel, Ed, Shalf, John, Skinner, David, Snir, Marc, Sterling, Thomas, Stevens, Rick, Streitz, Fred, Sugar, Bob, Sumimoto, Shinji, Tang, William, Taylor, John, Thakur, Rajeev, Trefethen, Anne, Valero, Mateo, van der Steen, Aad, Vetter, Jeffrey, Williams, Peg, Wisniewski, Robert & Yelick, Kathy 2011 The international exascale software project roadmap. *Int. J. High Perform. Comp. App.* **25** (1), 3–60.

Donzis, D. A. & Aditya, K. 2014 Asynchronous finite-difference schemes for partial differential equations. *J. Comp. Phys.* **274** (0), 370 – 392.

Donzis, D. A. & Sreenivasan, K. R. 2010 The bottleneck effect and the Kolmogorov constant in isotropic turbulence. *J. Fluid Mech.* **657**, 171–188.

Donzis, D. A., Yeung, P. K. & Sreenivasan, K. R. 2008 Dissipation and enstrophy in isotropic turbulence: scaling and resolution effects in direct numerical simulations. *Phys. Fluids* **20**, 045108.

Frommer, A & Szyld, DB 2000 On asynchronous iterations. *J. Comput. Appl. Math.* **123**, 201–216.

Gerstenberger, Robert, Besta, Maciej & Hoefler, Torsten 2013 Enabling highly-scalable remote memory access programming with mpi-3 one sided. In *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*, pp. 1–12. IEEE.

HIRSCH, C. 1994 *Numerical computation of internal and external flows*, , vol. 1. Wiley, New York.

HOEFLER, T., SCHNEIDER, T. & LUMSDAINE, A. 2008 Accurately measuring collective operations at massive scale. In *Proceedings of the 22nd IEEE International Parallel & Distributed Processing Symposium, PMEO'08 Workshop*.

HOEFLER, T., SCHNEIDER, T. & LUMSDAINE, A. 2010 Characterizing the influence of system noise on large-scale applications by simulation. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10)*.

ISHIHARA, TAKASHI, GOTOH, TOSHIYUKI & KANEDA, YUKIO 2009 Study of high-Reynolds number isotropic turbulence by direct numerical simulation. *Annual Review of Fluid Mechanics* **41**, 165–180.

JAGANNATHAN, S. & DONZIS, D. A. 2012 Massively parallel direct numerical simulations of forced compressible turbulence: a hybrid MPI/OpenMP approach. *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment* p. 23.

KARIMABADI, H, DRISCOLL, J, OMELCHENKO, YA & OMIDI, N 2005 A new asynchronous methodology for modeling of physical systems: breaking the curse of courant condition. *J. Comp. Phys.* **205** (2), 755–775.

LEE, M., MALAYA, N. & MOSER, R. D. 2013 Petascale direct numerical simulation of turbulent channel flow on up to 786k cores. In *Proceedings of the International*

Conference on High Performance Computing, Networking, Storage and Analysis, pp. 61:1–61:11. New York, NY, USA: ACM.

Maqui, Agustin & Donzis, Diego 2011 Dynamics of non-local interactions in isotropic turbulence. In *APS Meeting Abstracts*, , vol. 1, p. 12002.

Mudigere, Dheevatsa, Sherlekar, Sunil D & Ansumali, Santosh 2014 Delayed difference scheme for large scale scientific simulations. *Physical review letters* **113** (21), 218701.

Rogallo, R. S. 1981 Numerical experiments in homogeneous turbulence p. NASA Tech. Memo. 81315.

Stoer, Josef & Bulirsch, Roland 2013 *Introduction to numerical analysis*, , vol. 12. Springer Science & Business Media.

Tannehill, J. C., Anderson, D. A. & Pletcher, R. H. 1997 *Computational fluid mechanics and heat transfer*. Taylor & Francis.

Tavakoli, Rohallah & Davami, Parviz 2006 New stable group explicit finite difference method for solution of diffusion equation. *Applied Mathematics and Computation* **181** (2), 1379 – 1386.

Watanabe, T. & Gotoh, T. 2007 Inertial-range intermittency and accuracy of direct numerical simulation for turbulence and passive scalar turbulence. *J. Fluid Mech.* **590**, 117–146.

Yoo, Chun Sang, Lu, Tianfeng, Chen, Jacqueline H & Law, Chung K 2011 Direct numerical simulations of ignition of a lean n-heptane/air mixture with temperature inhomogeneities at constant volume: Parametric study. *Combustion and Flame* **158** (9), 1727–1741.