Enhancing CFL barrier for Transient Systems

Nishant Soni

Engineering Mechanics Unit
JNCASR

This dissertation is submitted for the degree of
M.S.(Engg.)

August 2020

I would like to dedicate this thesis to *Prof Manoj T Nair*.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Nishant Soni
August 2020

Certificate

I hereby certify that the matter embodied in this thesis entitled **"Enhancing CFL barrier for Transient Systems"** has been carried out by **Mr. Nishant Soni** as a student of the Engineering Mechanics Unit, Jawaharlal Nehru Centre for Advanced Scientific Research, Bangalore, India under my supervision and that it has not been submitted elsewhere for the award of any degree or diploma.

Prof. Santosh Ansumali
Dr. Diwakar S.V.
(Research Supervisors)

Acknowledgements

I would like to express my deep gratitude to, Prof Santosh Ansumali and Dr. Diwakar S.V., my advisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work. It was a pleasure to learn the intricacies of numerical solution of PDEs and high performance computing from them.

I thank Prof. Ganesh Subramaniam for teaching Fluid Mechanics. I thank Prof. K.R. Sreenivas for teaching Heat Transfer and giving me an opportunity to perform numerical studies on “Marangoni Convection”. I thank Prof N.S.Vidhyadhiraja for his lectures on Python in Basic Computational Science.

It was fun working with my lab mates Mahan, Praveen, Manjusha, Atif, Samarth, Shaurya, Akshay where I got relieved of the stress. I thank Swastik and Shaumya for the combined studies and motivation during hard times. I thank Dr. Ponnulakshmi VK, Mr. Abhijit Verma, Mr. Niranjan Kumar and Mr. Iqbal Ahmad for mentoring me. I thank my batchmates Prashanth, Albin, Pulkit for all their help and fun we had together during semesters and after exams. I thank Dr. Jaju, Dr. Soumyadeep for helping me on various computational issues from time to time.

I thank all my EMU friends Prateek, Arun, Sankalp, Sabarish, Rafi, Vaibhav for all their help. I thank Meha, Karthik, Rhoheth, Dr. Subrat, Nikhil for the enjoyable time in JNCASR. I thank my friends Anurag, Vikash, Roohi, Shweta, Smrithi, Kaushal, Vikram, Niraj, Sahil, Shreyas, Rajiv, Raushan for their constant support and encouragement.

I thank our administrative officers and staff, library, complab, hostel and medical staff for their cooperation. I am indebted to Dr. Elizabeth Daniel who extended her support through my high and lows and in finishing this journey finally.

I wish to thank my parents and my family for their unconditional support and encouragement.

Abstract

A number of scientific and engineering systems are governed by PDEs which have solutions comprising wide range of spatial and temporal scales. The accompanying details can only be captured by high-fidelity simulations on high performance computational systems. Although, advances in computing technology have made it possible to carry out intensive simulations on massively parallel computers, the presence of order of magnitude gaps in spatial and temporal scales is a major obstacle to sustained performance for a number of scientific codes. Thus, the development of PDE solvers which can bridge these order of magnitude gaps in spatial and temporal scales is a major challenge for computational physics and forms a central topic for high-performance computing. In this thesis, we have presented a general methodology to analyze and derive explicit schemes that overcome the issue of small time-steps by allowing some tunable level of asynchrony i.e. using spatial data from present and previous time levels for computing derivatives.

The concept relies on finite differences to approximate derivatives using values of the function from neighboring points and the realization that wider stencils have better stability. The performance of explicit solvers is stalled due to the time-step restrictions imposed by the stability criteria which is more severe for a problem with higher number of degrees of freedom. In few recent works, it was shown that the stability limit for the explicit solver of the diffusion equation was doubled in the delayed difference scheme where computations can proceed using values from past time levels. In this work, we highlight a checkerboard instability in the formulation of the delayed difference scheme for 1D diffusion equation. In order to overcome this issue, we consider a general scheme that is weighted between the delayed and the conventional explicit schemes. We have established via stability analysis that an optimum value of the weighting factor exists such that the critical CFL number obtained is larger than the original delayed difference scheme. These schemes are referred as weighted difference schemes.

The multidimensional extension of weighted differencing approach for diffusion equation is accomplished using isotropic differential operators to compute the spatial derivatives. By analyzing in detail the effective differential equation, we demonstrate the consistency and order of accuracy of weighted scheme. We have provided a general framework in which stability of schemes can be accessed by analyzing the eigen value spectra of the diffusion operator for different discretization procedure namely central difference and isotropic operators. Fourier analysis of the diffusion operator revealed that isotropic discretization, $\nabla \cdot \nabla$, alleviates the dependency of CFL on dimension of problem as opposed to the conventional central difference discretization. The use of isotropic differential operators along with weighted combination of delayed and non-delayed difference schemes has shown significant improvement in CFL criteria for multi-dimensional problems.

In order to access the computational efficiency of weighted schemes, we have performed a series of numerical experiments for transient and steady computations and compared the performance against standard explicit time marching schemes and iterative solvers. Theoretical predictions on the accuracy and convergence rates of weighted schemes have been compared to numerical experiments for transient and steady problems. Good agreement has been found across different initial and boundary conditions. We have also performed weak and strong scaling studies for the parallel version of the code on Intel KNL architecture. The work presented here provides a strong foundation for deriving schemes which inherit the simplicity and parallel efficiency of the Jacobi method while retaining the fast convergence property of the Gauss-Seidel method.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Understanding the time evolution of systems is a common quest in plethora of science and engineering applications. Many of these systems typically manifest complex structures that have both spatial & temporal hierarchy of scales and their common examples include turbulent fluid flow, magneto-hydrodynamic (MHD) flow in solar corona, tumor growth in tissues, etc.,[2–6]. Any process of understanding them precisely via theoretical/numerical approach would require formulating a mathematical model that can truly describe the physical behavior of system/phenomena. These mathematical models would generally consist of the underlying governing equations derived from first principles and the accompanying constraints which include all the relevant initial and boundary conditions. Unfortunately in most cases, the models involve a system of non-linear partial differential equations (PDEs) that rarely yields closed form solutions. Only in the case of simple problems like modeling of an electric circuit or a simple pendulum, it is possible to find an analytical solution which describes the time evolution of the imposed initial condition. For all the other scenarios of non-linear systems involving complex structures and hierarchy of scales, the recourse is normally sought via intensive numerical simulations carried out using state-of-the-art high performance compute clusters with tens to hundreds of thousands of processing elements (PEs).

In order to elucidate the above arguments, we now consider the example of Navier-Stokes equations which describe the underlying fluid flow phenomena in numerous scientific and engineering applications ranging from micro-circulation in biological systems to flow past an aircraft wing to weather prediction etc. Navier-Stokes equations come in different flavors and a simplified version of these equations can be obtained for the incompressible flow regime as,

$$\nabla \cdot \mathbf{u} = 0, \tag{1.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}. \tag{1.2}$$

Here, $\mathbf{u}$ and p represent the flow velocity and pressure fields, respectively. ρ is the fluid density and ν is the kinematic viscosity. In the above set of equations, the essential nonlinearity comes from the term representing the convective acceleration ($\mathbf{u}\cdot\nabla\mathbf{u}$) and there is no simple analytical technique that can help in solving the system. Only in a few special cases where the non-linearity is overcome by using simplifying assumptions, closed form solutions for Navier-Stokes equations can be obtained. For example, fully developed laminar flow in a pipe.

A common remedy to the above problem is to look for approximate solutions involving different numerical methods. One such approach is the method of weighted residuals [7] that have been extensively used for obtaining approximate numerical solutions of differential equations. Based on the choice of basis and test functions, different variants of this method have been developed in the past and the simplest of such approach is the method of finite differences [3, 8]. In this approach, the continuous partial differential equation is converted to a set of algebraic difference equations that are evaluated at discrete set of points in the domain commonly known as grid points or mesh points. A typical cartesian grid is shown in fig. 1.1 where the grid points are labelled using indices 'i' and 'j' in the x and y directions respectively. Δx, Δy represent the grid spacing in these directions. Using Taylor series expansions, the continous partial derivatives are replaced by approximate algebraic difference quotients that connect different variables $u,\ v,\ ,p,\ \rho$, etc. at these discrete grid points (x_i, y_j). So essentially, we are foregoing the possibility of obtaining continuous field information and we are settling for solution at a few selected points at the benefit of being able to solve the problem in the first place. In other words, the validity of the governing equations are enforced only at selected points in a discrete form. Subsequently, the field information is obtained by either solving a system of equations or by marching in time, based on the temporal nature of the system i.e. the system being either steady or transient.

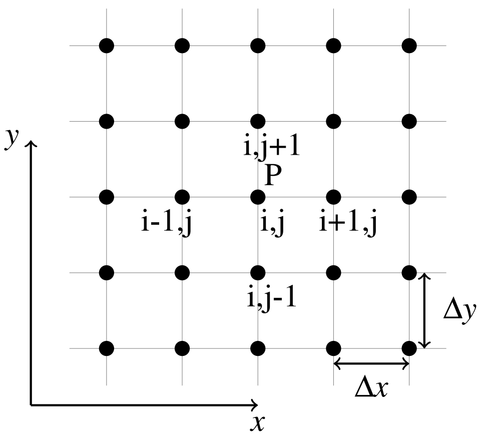


Fig. 1.1 Discrete grid points in xy-plane.

1.1 Explicit and Implicit Formulation of Time-dependent PDE's

In majority of the problems of interest, the main focus is to understand the temporal evolution of the physical system which are fraught with hierarchy of scales. For example, turbulent flows contain spatial structures (eddies) ranging from the smallest Kolmogrov scale to the size of the actual physical system. Such disparities in the spatial scales are accompanied with a hierarchy in temporal scales and hence, it is imperative that the methodology adopted for performing numerical time integration in such problems is capable of handling such diversity.

The focus of the current work is on developing efficient time integration tools that can be used for a variety of problems. Note that the efficiency of the time integration tools is often determined by the rate determining transport process, which is often diffusion. Hence, we devote our attention here on increasing efficiency of such integration methods for prototype diffusion problems. For the sake of simplicity, we now consider the unsteady one-dimensional diffusion equation with constant diffusivity as the model equation and understand certain attributes of time marching. The one-dimensional transient diffusion equation is given as,

$$\frac{\partial \phi}{\partial t} = \kappa \frac{\partial^2 \phi}{\partial x^2}. \tag{1.3}$$

Based on the classification of PDE's, Eq.(1.3) can be identified as a mixed parabolic in time and elliptic in space type partial differential equation. Being parabolic in time, Eq.(1.3) lends itself to a marching solution that is evolved in time from a given initial condition. Conventionally, the time discretization procedure involves the use of either an explicit or an implicit estimation of the temporal derivatives. The explicit procedure estimates the temporal derivatives at current (or previous) time levels to effectuate the marching procedure. An explicit time marching scheme for Eq.(1.3) where the derivatives are estimated at time level 'n' can be written as,

$$\frac{\phi_i^{n+1}-\phi_i^n}{\Delta t}=\frac{\kappa\left(\phi_{i+1}^n-2\phi_i^n+\phi_{i-1}^n\right)}{\Delta x^2}, \tag{1.4}$$

where the discrete representation ϕ_i^n represents the value of ϕ at grid point x_i corresponding to time level n i.e., $\phi_i^n=\phi\,(i\Delta x, n\Delta t)$. Eq.(1.4) can be further simplified as,

$$\phi_i^{n+1}=\phi_i^n+\frac{\kappa\Delta t}{\Delta x^2}\left(\phi_{i+1}^n-2\phi_i^n+\phi_{i-1}^n\right). \tag{1.5}$$

The above procedure is specifically known as the forward Euler time integration or FTCS scheme (forward in time central in space). As can be clearly seen, the difference equation for an explicit time integration scheme contains only one unknown which can be computed in a straightforward manner. Thus, the explicit methods are simple to formulate and like the underlying PDE, they preserve the causality (past not affected by future). However, issues with explicit methods occur on account of their stability. To understand this better, we look at the evolution pattern of numerical errors in the above procedure. It can be shown that the evolution of error (ε) in the solution also follows the same difference equation i.e.,

$$\varepsilon_i^{n+1}=\varepsilon_i^n+\frac{\kappa\Delta t}{\Delta x^2}\left(\varepsilon_{i+1}^n-2\varepsilon_i^n+\varepsilon_{i-1}^n\right). \tag{1.6}$$

Based on the von-Neumann stability analysis, the amplification factor of error in each time step at any point 'i' can be given by the formula,

$$G=1-4\frac{\kappa\Delta t}{\Delta x^2}\sin^2\theta, \tag{1.7}$$

where $\sin^2\theta\leq 1$. Thus for the scheme to remain numerically stable, we must have $\mid G\mid\leq 1$ which leads to the following condition that

$$\frac{\kappa\Delta t}{\Delta x^2}\leq\frac{1}{2}. \tag{1.8}$$

The quantity $\frac{\kappa\Delta t}{\Delta x^2}$ is commonly known as Courant-Friedrichs-Lewy number or in short CFL number for diffusion equation. Since the choice of Δx is usually decided based on the accuracy considerations, the maximum allowable Δt is decided by the CFL limit once Δx is fixed. Any choice of Δt above this prescribed limit leads to divergence of the solution.

Hence, it can be concluded that the explicit methods are *conditionally* stable wherein the CFL number always has to be less than a certain value prescribed by the *stability criterion*.

Implicit approach, on the other hand, evaluates the temporal derivatives at time level 'n+1'. As a result, the unknown ϕ_i^{n+1} is not only expressed in terms of known quantities at level 'n' but also in terms of other unknown quantities at time level 'n+1'. The discrete analog of Eq.(1.3) for an implicit time marching scheme can be written as

$$\phi_i^{n+1} = \phi_i^n + \frac{\kappa \Delta t}{\Delta x^2}\left(\phi_{i+1}^{n+1} - 2\phi_i^{n+1} + \phi_{i-1}^{n+1}\right) \tag{1.9}$$

The difference equation Eq. (1.9) is commonly known as backward Euler time integration method. Since there are more than one unknown, it becomes neccessary to formulate a system of equations by appying Eq. (1.9) at all the interior grid points. The resulting system of equations are solved simultaneously by using special solvers to compute the unknowns ϕ_i^{n+1} for all i. The need to solve large systems of simultaneous algebraic equations also requires handling large matrices. However, by performing a von-Neumann stability analysis for Eq.(1.9) we can arrive at the following relation for amplification factor.

$$G = \frac{1}{1 + 4\frac{\kappa \Delta t}{\Delta x^2}\sin^2\theta}. \tag{1.10}$$

From the above equation it is clear that $\mid G \mid$ is always ≤ 1 irrespective of the value of Δt and hence the implicit approach is unconditionally stable. Thus, typical implicit methods integrate with time-steps which exceed the fastest time-scales present in the system by an order of magnitude. This is achieved by violating causality which reduces the problem to that of solving a system of equations at each time step.

A graphical representation of the space-time stencil associated with Eq.(1.5) and Eq.(1.9) is shown below in Fig. 1.2.

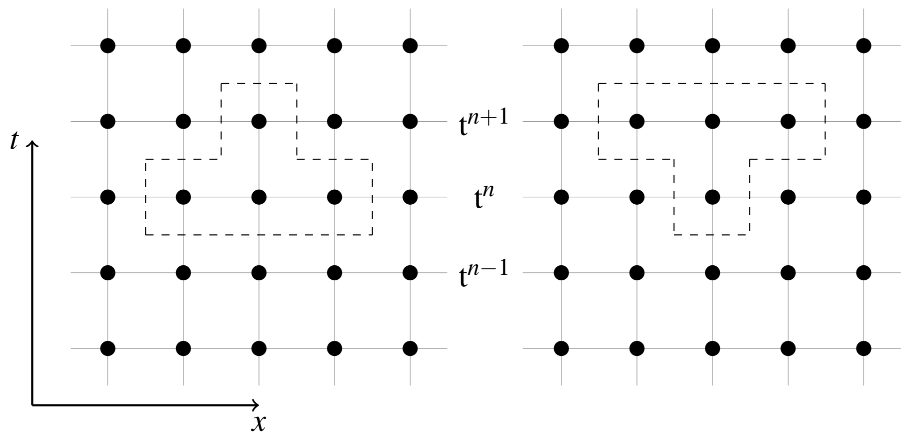


Fig. 1.2 Space-time stencils for explicit (left) and implicit (right) discretization.

In summary, pros and cons associated with the two approaches can be written as,

Explicit methods

Advantages:

- Ease of implementation.
- Well suited for vectorization and parallelization on modern computer architectures.
- Requires lower memory storage.
- CPU cost per time step is low (no matrix inversion involved).

Disadvantages:

- Conditional stability leads to restriction on maximum allowable time step.
- In some cases, Δt must be very small to maintain stability, this can lead to long computation time to perform calculations over a given time interval.

Implicit methods

Advantages:

- Unconditional stability allows for use of larger Δt in simulations.
- Calculations over a given time interval can be done in considerably less number of time steps resulting in reduction of computation time for serial applications.

Disadvantages:

- Computer program implementation is more complicated even for serial applications.
- Parallelization turns out to be non-trivial.
- CPU cost per time update is significantly higher in contrast to explicit methods (matrices have to be inverted at each time step).
- Storing matrices leads to additional memory requirements.
- Large Δt can yield inaccurate results as compared to explicit methods owing to large truncation error which is not desirable when timewise accuracy for transient systems is concerned.

From the above summary, it can be concluded that in the conventional serial computing environment, implicit approaches such as Crank-Nicolson scheme for diffusion equation are the de-facto first choice and over years, such approaches have become even more powerful [9, 10]. However, it is well recognized that in a modern heterogeneous computing infrastructure involving inter-connected multi-core servers, it is difficult to extract performance from implicit methods owing to the data dependency in the algorithm. This has lead to a renewed interest in harnessing the data independent nature of the explicit schemes [5, 11]. It may be noted that the computational efficiency per time-step and parallel performance of a typical explicit scheme is quite high. However, the small time steps used in explicit methods often overshadow the gain due to better computational efficiency. Correspondingly, many approaches have been attempted towards relaxing the time-step restriction of the explicit procedures. The current work also proposes and characterizes one approach considered in this regard.

1.1.1 The curse of dimensionality

The issue of CFL limit as described in the previous section is not just limited to one-dimensional problems. Infact, the situation worsens at higher dimensions. We have previously seen that for 1-D diffusion equation the stability criteria for FTCS discretization obtained via von-Neumann stability analysis states that

$$\frac{\kappa \Delta t}{\Delta x^2} \leq \frac{1}{2} \tag{1.11}$$

i.e. CFL number should be less than or equal to 0.5 for the numerical solution to be stable. By performing a similar analysis we can obtain the CFL stability criteria for multi-dimensional diffusion equation. Specifically, the CFL limit obtained with FTCS discretization are given below.

$$\frac{\kappa \Delta t}{\Delta x^2} \leq \frac{1}{4} \text{ in 2-D} \tag{1.12}$$

and

$$\frac{\kappa \Delta t}{\Delta x^2} \leq \frac{1}{6} \text{ in 3-D.} \tag{1.13}$$

An important conclusion that can be drawn from the above equations namely Eq. (1.11), Eq. (1.12) and Eq. (1.13) is that the maximum allowable CFL varies inversely with the dimension. The implication of such a dependence is that the explicit approach can become computationally too restrictive in higher dimensions because of the stringent time step criteria imposed by stability requirements.

1.2 Overview of the current work

As seen in the previous sections, the stringent stability criteria associated with explicit methods compounded with the curse of dimensionality makes it difficult for these methods to be used for solving large scientific problems. Therefore, numerical methods that can overcome the stability limits of conventional explicit methods such as forward-Euler are required if one wishes to harness the high computational efficiency and parallel performance of explicit schemes. An attractive option in this regard is to use 'super time-stepping' (STS) methods ([5], [12], [13]) which are quite popular due to its simplicity and wide stability window in contrast to forward Euler type methods.

An alternate route for obtaining explicit methods with large time-steps have emerged with the attempts to solve PDE's in an asynchronous fashion [14–18]. This has lead to the development of a class of asynchronous schemes called as the delayed difference schemes where the derivatives are estimated at a time-step earlier than the regular explicit procedure. An unexpected outcome of the work was the doubling of the stability limit while obtaining an explicit solution for the diffusion equation [19, 14]. However, the delayed scheme is fraught with oscillatory behavior. It will be shown that the method exhibits checkerboard instability in 1-D for certain initial conditions. Nevertheless, the improved stability has motivated the present study for developing an efficient explicit method that improves on the delayed difference scheme to provide enhanced stability and accuracy. We show that a reformulation of the method by taking a weighted combination of delayed and conventional explicit scheme can eliminate the checkerboard artefact. It will be shown via a stability analysis that an optimum value of the weighting factor exists such that the critical CFL number obtained is larger than the original delayed difference scheme. In the case of multi-dimensional

problems, we use the recently introduced isotropic lattice differential operators and then show that weighted combination of delayed and non-delayed difference schemes can result in significant improvement in CFL criteria for multi-dimensional problems.
In this work, the diffusion equation is used as the model problem to test different methods as such a consideration would come without any loss of generality. This can be understood by analyzing the stability implication of different terms in a PDE like advection, diffusion, etc. It can be shown that respecting stability of the diffusion term would make $\Delta t \propto \Delta x^2$ whereas for the advection term, it would require $\Delta t \propto \Delta x$. Since Δx is normally chosen to be small to obtain better spatial accuracy, it is the diffusion term that imposes severe restrictions on the time step. Hence, the analysis is primarily focused on the diffusion equation, particularly for one-dimensional systems in Chapter 3 of this thesis and for multi-dimensional systems in Chapter 4.

Essentially, the present thesis is organized as follows:

- In chapter 2, we report some of the recent developments in computational science related to numerical solution of parabolic equations.
- In chapter 3, we present in detail the one-dimensional implementation of the current algorithm followed by performance comparison with the standard explicit methods to solve diffusion equation.
- In chapter 4, we extend the implementation of current algorithm to multi-dimensional problems. A detailed derivation for the stability of current numerical scheme and effective PDE is included along with numerical experiments to validate the solver. The efficacy of our current approach will also be assessed through the solution of Poisson's equation in a pseudo-transient framework where we compare the numerical results with conventional iterative methods. Weak and Strong scaling analysis of the parallel version of the three-dimensional code is also included.
- In chapter 5, we provide a summary of findings from the current work with suggestions for future extension.

Chapter 2

Literature Review

2.1 Background

As described in the previous chapter, the focus of the current work is on development of fast and efficient numerical techniques for solving transient partial differential equations. Specifically, our primary effort will be devoted towards enhancing the stability criteria associated with explcit time integrators of diffusion equation. Accordingly, we present a brief review of literature in this chapter with an emphasis on recent developments in numerical solution of parabolic partial differential equations. To begin with, we look at different attempts that have been made in the recent past to overcome the stability restrictions in explicit methods that include super time stepping methods [20] and the asynchronous delayed difference method [14, 15]. This is followed by some of the recent works on the spatial discretization of differential operators where specific attention has been given with regard to isotropy in discrete representation of continuum operators such as laplacian and their implication on the stability of the temporal schemes [19, 21].

2.2 Explicit super time-stepping methods

Multiscale nature of transient dynamics is a regular feature in complex physical systems which often have processes acting on wide range of time-scales. Capturing the detailed evolution of these wide range of time-scales in numerical simulations is a daunting task due to the finer time-resolutions required. As seen already, numerical stability restrictions associated with conventional explicit methods can limit the time-steps making the simulations computationally intractable for complex problems. Hence, it is desirable to develop efficient methods that enhance the stability limit while operating on such stiff systems.

Notable of different approaches in the literature for stability enhancement include the relatively lesser known scheme termed as "super time-stepping" methods which have been used in astrophysical magnetohydrodynamics (MHD) codes [5, 22]. The main idea behind this method is that the restrictive stability condition of the forward Euler can be relaxed if we do not require stability at the end of each time-step but rather at the end of N time-steps of unequal length. This cycle of N time-steps leads to a Runge-Kutta like method where stages are added for stability rather than accuracy.

W. Gentzsch [20], in his pioneering work, has shown that the stability restrictions of one-step explicit methods can be relaxed by demanding stability not for a single step but for two or more time steps taken together. This can be illustrated by considering the example of 1-D diffusion equation for which the one-step explicit method (FTCS),

$$\phi_i^{n+1} = \phi_i^n + \frac{\kappa \Delta t}{\Delta x^2}\left(\phi_{i+1}^n - 2\phi_i^n + \phi_{i-1}^n\right)$$

is known to be stable, if

$$\left| 1 - 4\frac{\kappa \Delta t}{\Delta x^2}\sin^2\left(\frac{\mathrm{k}_x \Delta x}{2}\right)\right| \leq 1 \implies \frac{\kappa \Delta t}{\Delta x^2} \leq \frac{1}{2}.$$

If one were to make this time marching in two steps, involving CFL numbers of α_1 and α_2 i.e.,

$$\phi_i^{n+1} = \phi_i^n + \alpha_1\left(\phi_{i+1}^n - 2\phi_i^n + \phi_{i-1}^n\right)$$

and,

$$\phi_i^{n+2} = \phi_i^{n+1} + \alpha_2\left(\phi_{i+1}^{n+1} - 2\phi_i^{n+1} + \phi_{i-1}^{n+1}\right)$$

where $\alpha_1 = \frac{\kappa \Delta t_1}{\Delta x^2}$ and $\alpha_2 = \frac{\kappa \Delta t_2}{\Delta x^2}$. Then, the stability criteria at the end of two time steps can be obtained as

$$|(1-4\alpha_1)(1-4\alpha_2)| \leq 1.$$

where

$$\alpha_1 = \frac{\kappa \Delta t_1}{\Delta x^2} = 1 - \frac{\sqrt{2}}{2} \text{ and } \alpha_2 = \frac{\kappa \Delta t_2}{\Delta x^2} = 1 + \frac{\sqrt{2}}{2}.$$

Hence, it follows that $\alpha_1 + \alpha_2 = 2$ instead of $2\alpha = 2 \times 0.5 = 1$ that is typically obtained in the case of forward-euler with constant time-steps. Thus, one super-step, $\Delta T = \Delta t_1 + \Delta t_2 = \frac{\Delta x^2(\alpha_1+\alpha_2)}{\kappa} = \frac{2\Delta x^2}{\kappa}$ is two-times larger than $\frac{\Delta x^2}{\kappa}$ i.e. two FTCS steps with constant Δt. So, in general if N is the number of time-steps or stages belonging to a single "super-step" it will result in a speed up of N-times over the conventional explicit method.

Alexiades *et al* [12] subsequently presented an improvement over Gentzch's original method. Their super time stepping explicit scheme was achieved by requiring stability at the end of a cycle with N time-steps of different sizes $\tau_1, \tau_2, ..., \tau_N$ that are chosen to maximize the total duration, $\Delta T = \sum_{i=1}^{N} \tau_i$. A detailed derivation was given by making use of the optimality properties of Chebyshev polynomials and were thus termed as Runge-Kutta-Chebychev (RKC) methods. They subsequently performed numerical studies on couple of benchmark problems involving heat equation (linear) and Stefan problem (non-linear) with super-time stepping scheme and compared with standard versions of Crank-Nicolson and fully implicit schemes using SOR and Newton-iterations [13, 23]. Results obtained with STS methods were found to be comparable with implicit methods both in terms of accuracy and speed [3, 12, 13, 23]. At the same time, STS methods retained the ease of implementation of conventional explicit methods.

It was pointed out by Meyer *et al* [24] that RKC methods are not monotonic in nature and they produce spurious oscillations when the diffusion coefficient is either spatially varying or solution dependent. This was demonstrated through numerical experiments for the simple case of one-dimensional thermal conduction between two different materials. Further for non-linear diffusion operators, it becomes difficult to assess the stability of RKC schemes and often the RKC schemes are augmented by an additional damping factor to help stabilize the schemes for non-linear problems. This behavior was explained based on the properties of Chebyshev polynomials and eigen spectrum analysis of the spatial discretization operator. This can be understood from the diffusion equation written in a semidiscrete ODE form as

$$\frac{d\phi}{dt} = \mathbf{M}\phi(t) \tag{2.1}$$

where $\mathbf{M}$ is a symmetric, constant coefficient matrix resulting from the discretization of parabolic diffusion operator (assuming the problem is linear). The eigen values of $\mathbf{M}$ are real and non-positive. For a n-stage scheme with an overall super-step τ, the update can be expressed in terms of the stability polynomial referred to as the amplification factor, R_n, i.e.

$$\phi(t+\tau) = R_n(\tau\mathbf{M})\phi(t). \tag{2.2}$$

The scheme is stable if $\mid R_n(\tau\lambda) \mid \leq 1$ for the entire set of λ between 0 and maximum negative eigenvalue of $\mathbf{M}$. The RKC methods are designed using shifted Chebyshev polynomials since it is known that the absolute value of these polynomials is always ≤ 1. Thus, the magnitude of amplification factor at the end of each super-time step is ≤ 1. It may be noted that RKC-STS method requires damping for complicated diffusion problems since the stability polynomials and hence the amplification factor can become equal to 1. In order to address the above issue with RKC methods, another variant of the super-time-stepping schemes was developed by Meyer *et al* [24] based on the shifted Legendre polynomials and was hence, called as Runge-Kutta-Legendre (RKL) methods. The stability polynomial for a general n-stage RKL scheme can be written as

$$R_n(\tau\lambda) = a_n + b_n P_n(w_0 + w_1\tau\lambda). \tag{2.3}$$

The parameter w_0 is equal to 1 for all RKL schemes. First and second-order accurate formulations of RKL methods, called as RKL1 and RKL2, were derived by enforcing the stability polynomial at sub-stage j to be $a_j + b_j P_j(\tau\mathbf{M})$ and using the property that Legendre polynomials are bounded by unity in magnitude when their argument lies in the range $(-1,1)$. It was shown that RKL schemes have superior stability as compared to RKC owing to the fact that Legendre polynomials, unlike Chebyshev polynomials, are always smaller than unity in magnitude when their argument lies in $(-1,1)$. This also makes RKL schemes more robust than RKC and eliminates the need of an additional damping parameter for complicated problems. The RKL schemes have a convex monotonicity preserving property and are thus useful in parabolic problems with variable diffusion coefficients. These properties were demonstrated by the authors for a wide range of linear and non-linear test cases in one, two and three dimensions.

In their recent work, Caplan *et al* [22] have compared the performance of the second-order Runge-Kutta Legendre (RKL2) STS method with the implicit backward Euler scheme computed using preconditioned conjugate gradient solver (BE + PCG) for integrating parabolic operators in thermodynamic MHD models of solar corona. Both the methods were tested on a production-level coronal relaxation simulation of MAS code with special emphasis given to the performance and scaling of the methods. It was found that RKL2 scales much better for high core counts than the backward Euler and matches or exceeds the performance for low number of cores. Despite their better performance and scaling, it was found that RKL2 methods have some drawbacks. Specifically, analysis of the methods showed that the amplification factor of RKL2 scheme doesn't generally monotonically decrease for increasing wave numbers and is quite high for the highest modes. The inability of the RKL2 scheme to adequately damp high wave modes was reflected in results obtained from numerical solutions where it failed to damp grid-level oscillations. As such, RKL2 methods can exhibit slowly-growing grid-level oscillations in a few localized portions of the domain and their unchecked growth can lead to unphysical solutions that could rapidly propagate into the global solution. The authors have suggested some methods of avoiding these errors but each would slow down the algorithm although they wouldn't adversely affect the scaling.

B Vaidya *et al* [5] have compared various STS schemes in the context of anisotropic thermal conduction in an MHD plasma where it was also found that RKL-STS performs better than RKC-STS (AAG-STS). The presence of large gradients in anisotropic diffusion can lead to non-monotonic behaviour of temperature and conventionally limiters have been useful in preventing negative temperatures in such cases. It was found that although the use of limiters improves the robustness of all schemes, they reduce the accuracy of STS schemes for a large number of sub-steps. Their studies also confirmed that STS schemes based on Chebyshev polynomials require an artificial damping for numerical stability while RKL-STS schemes are more robust with sufficient inherent damping and can better exploit the STS strategy. Parallel computation studies performed with RKL-STS schemes have shown an excellent strong scaling with an efficiency of 80 percent for number of processors upto 10^4 on modern distributed Petascale supercomputers.

Apart from astrophysical applications, super-stepping methods have also been employed in other diverse areas such as image processing [25], electroanalytical digital simulations in computational electrochemistry [26], simulations of isotropic/anisotropic diffusion on curved biological surfaces [27], simulations of deformable body dynamics for computer animation and design [28].

2.3 Studies on Asynchronous schemes

An alternate approach to developing explicit methods capable of handling large time steps has emerged from the attempts to solve PDEs in an asynchronous fashion [14–18]. These works were primarily concerned with the optimization of data communication in stencil-based computation methods like finite differencing. Note that for stencil computations, bulk of the communication is spent on updating/communicating information from the neighbouring nodes. The above methods aim to optimize the solution procedure by appropriately interleaving and overlapping communication with computation and memory access.

Dheevatsa *et al* [14] have shown that it is possible to derive a class of efficient asynchronous schemes, known as the delayed difference schemes, by introducing a temporal delay within the difference approximation. In this approach, the derivatives are estimated at a time step earlier than the regular explicit procedure. The idea was illustrated by considering the diffusion equation $\partial_t \phi = \kappa \partial_x^2 \phi$ for which the general form of delayed difference scheme is given as,

$$\phi_i^{n+1} = \phi_i^n + \frac{\alpha}{(m+1)^2}\left(\phi_{i+m+1}^{n-m} + \phi_{i-m-1}^{n-m} - 2\phi_i^{n-m}\right), \tag{2.4}$$

where $\alpha = \frac{\kappa \Delta t}{\Delta x^2}$. For $m = 0$, this scheme reduces to the standard FTCS scheme. It was found that the stability limit of delayed difference scheme ($m = 1$) was enhanced by a factor of 2 as compared to the FTCS scheme. In addition, the parallel efficiency of delayed scheme is better than FTCS because with the introduction of a time delay $(m > 0)$ the data used in communication are from the $(n-m)$-th step rather than the n-th step. Also with such an implementation, boundary synchronization is required only after $m+1$ steps where m corresponds to the time delay introduced in the difference equations. Thus, it was concluded that increasing the scalability of the code is a direct consequence of relaxing the data dependence from within an iteration to across iterations. This also improves the scope for overlapping communication with computations especially for large scale scientfic applications.

An important consideration that needs to be taken care for efficient resource utilization in a parallel computing framework is to minimize the processor(s) idle time. Conventionally, this issue has been dealt with by a uniform load balancing technique which requires the computational work to be equally distributed, at least in an approximate sense, among the different processors. Asynchronous methods have been conceived as an alternate approach to minimize/avoid the processor idle time where the central idea is to eliminate as much synchronizations as possible to reduce the time a processor has to wait for information from other processors [14, 15, 29]. This approach is becoming popular with the increasing use of heterogenous workstation clusters where the computational power of individual processors can be unpredictable and in such cases each processor can be allowed to advance at its own speed. Some of the works where the idea of asynchronous iterations has been applied for the solution of partial differential equations can be found in [18, 30, 31].

Aditya *et al* [16] in their recent studies have focused on the deterioration in accuracy of numerical solutions while solving partial differential equations in an asynchronous fashion to relax the synchronization between processors. The authors have presented a general methodology to develop finite difference schemes which can maintain their accuracy when synchronizations are relaxed.

In contrast to the idea of modifying difference equations, Ankita *et al* [17] have proposed modifying the governing equation as a function of delay. The central idea in their work involved modifying the transport equation at processor boundaries to reduce asynchronous errors. A numerical test case was presented involving one-dimensional advection diffusion equation.

Sauro Succi and collaborators [32] have recently put forward time-delayed version of Lattice Boltzmann as one of the tentative ideas in prospective exascale LB applications. Their central idea is again overlapping computation and communication (non-blocking) as the delay provides a longer lapse for data communication between processors which can be utilized for doing additional work and thus eliminating processor idle time. As suggested in their work, the streaming step for a M-level delayed version of LB would have the form,

$$f_i\left(\vec{x}, t+1\right) = \sum_{m=0}^{M} w_m f_i\left(\vec{x} - m\vec{c}_i, t-m\right), \tag{2.5}$$

where $f_i\left(\vec{x}, t\right)$ are a set of discrete distribution functions (populations in LB jargon) expressing the probability of finding a particle at position $\vec{x}$ and time t, and w_m are appropriate normalized weights. Unexpectedly, the authors have pointed out that the delayed version is subjected to more severe stability restrictions than the standard single-time level implementation.

2.4 Studies on development of discrete spatial operators

It is well known that the maximum allowable time step for a given explicit time integrator is larger for an isotropic spatial discretization than the corresponding value for conventional discretization. Shukla *et al* [21], in their work on isotropic finite volume methods, have reported an improvement in stability by a factor of 1.5 and 1.36 for diffusion and advection equations respectively in two-dimensions over the conventional finite volume method.

In the recent work by Mahan *et al* [19], the use of mimetic isotropic laplacian ($\tilde{\nabla} \cdot \tilde{\nabla}$) coupled with the delayed time integration approach also resulted in an improvement in stability by a factor of about 3.6 for two-dimensional diffusion equation compared to the conventional FTCS discretization. Since, our ultimate interest is to enhance the stability restrictions for explicit time integrators and noting that the use of isotropic differential operators can lead to an improvement in CFL stability criteria, we review some of the recent works on the development of isotropic discretization of differential operators in this section.

Simple lower order finite differences schemes still remain a popular choice for spatial discretization of differential operators [3, 8, 33]. With the conventional finite difference discretization, Taylor's series expansion are used to formulate the spatial derivatives in one-dimension and the same is extended to multi-dimensions in a straight forward manner. Although, such a discretization based on tensor product stencils offers ease of implementation for multi-dimensional problems but it also leads to loss of isotropy. This can be verified by considering the example of the laplacian operator [34]. The most widely used approximation to the laplacian (1D) based on central difference discretization is given by,

$$\frac{d^2\phi}{dx^2} \approx \frac{\phi\left(x+\Delta x\right) - 2\phi\left(x\right) + \phi\left(x-\Delta x\right)}{\Delta x^2} \tag{2.6}$$

Similarly in 2D, the discrete laplacian $\tilde{\nabla}^2$ of a scalar field $\phi\left(x,y\right)$ based on standard central difference is given as,

$$\tilde{\nabla}^2\phi_{i,j} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2} \tag{2.7}$$

where Δx and Δy are the grid spacing in x- and y-directions and $\phi_{i,j}$ is the value of ϕ at the point (i,j) in space. By performing a Taylor series expansion about the point (i,j) it can be shown that the discrete laplacian approximates the continous laplacian as

$$\tilde{\nabla}^2 \approx \nabla^2 + \frac{h^2}{12}\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right) \tag{2.8}$$

where $\Delta x = \Delta y = h$. In the above Eq.(2.8), the leading order truncation error is anisotropic [34]. This anisotropic behavior is quite evident in the Fourier space where the Fourier transform of the discrete laplacian exhibits an angular dependence. i.e.,

$$\tilde{\mathbf{L}}\left(\mathbf{k}\right) = \mathbf{L}\left(\mathbf{k}\right) + \frac{h^2}{12}k^4\left(1 - 2\cos^2\theta\sin^2\theta\right). \tag{2.9}$$

The normalized truncation error in the above equation has been plotted in Fig. 2.1 from which it is clear that the central difference operator has an angular dependence as the error is not the same at all angles. Thus, it can be concluded that conventional method of constructing the laplacian by taking central difference operator in each of the dimension separately leads to loss of isotropy in higher dimensions. Isotropy, which is an inherent property of the continuum laplacian operator, is a desirable feature even for the discrete representation. Specific attention to isotropy of discrete laplacian operator can be found in the works of Kumar [35], Thampi *et al.*[1], Patra *et al.* [36].

Kumar and Patra *et al.* have addressed this issue by providing algebraic methods for constructing isotropic laplacians. In contrast to this, Thampi *et al.* [1] have shown that lattices

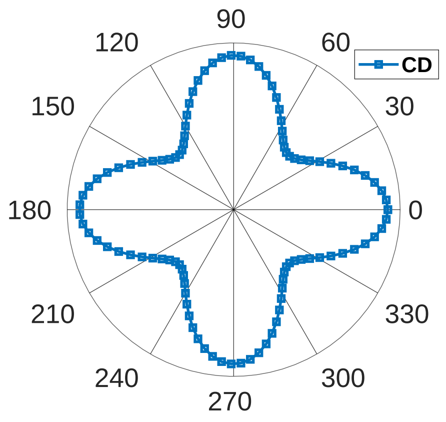


Fig. 2.1 Polar plot of Eq. 2.9. Radius of the plot represents the normalized error $\frac{\tilde{\mathbf{L}}(\mathbf{k})-\mathbf{L}(\mathbf{k})}{h^2k^4}$.

and associated weights commonly used in lattice hydrodynamics simulations, naturally provide discrete Laplacians with isotropic discretization error at the leading order. The important feature of the isotropy of lattice Maxwellians (discrete form of the Maxwell-Boltzmann velocity distribution) which is also a necessary condition for Navier-Stokes hydrodynamics on the lattice was exploited to generate discrete isotropic Laplacians in two and three dimensions. It was shown that for a DnQm lattice hydrodynamic model in n dimensions with m velocities, the Laplacian $\mathbf{L}(\mathbf{r})$ for a scalar function $\phi(\mathbf{r})$ defined on the lattice i.e. $\mathbf{L}(\mathbf{r}) \equiv \nabla^2\phi(\mathbf{r})$ has the form,

$$\mathbf{L}(\mathbf{r}) = \frac{2}{T}\left[\sum_{j=0}^{3}\sum_{i=1}^{N_j} w_i^j \phi\left(\mathbf{r}+\mathbf{c_i^j}\right) - \phi(\mathbf{r})\right] + O\left(\nabla^4\right) \tag{2.10}$$

where i labels the i-th discrete velocity in the j-th energy shell with N_j velocities (see Fig.2.2) and w_i^j is the corresponding weight factor and $T = \frac{1}{3}$. The values of weights w_i^j and the discrete velocities N_j for various DnQm models can be found in Table 1 in [1].

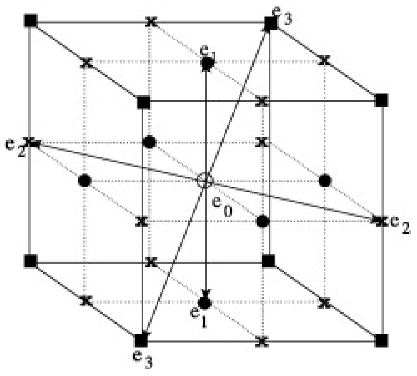


Fig. 2.2 Ordering of points on a cubic unit lattice according to "energy shells" as expalined in [1]. e_0, e_1, e_2, e_3 represent the energy shells corresponding to $j = 0, 1, 2, 3$.

A more general approach to construct isotropic discrete operators from lattice kinetic models such as gradient, divergence, curl and laplacian can be found in the recent work by Rashmi *et al.* [34]. It is beyond the scope of this chapter to go into a detailed description of the underlying method. For the sake of completeness, we list down the basic discrete operators viz. Gradient ($\tilde{\nabla}$), Divergence ($\tilde{\nabla}\cdot$) and Curl($\tilde{\nabla}\times$) for a given vector field ϕ on the lattice as given below.

$$\tilde{\nabla}^{iso}\phi = \frac{1}{\Delta x}\sum_{i=1}^{N} w_i \hat{c}_i \phi(r_i + c_i) \tag{2.11}$$

$$\tilde{\nabla}^{iso}\cdot\phi = \frac{1}{\Delta x}\sum_{i=1}^{N} w_i \hat{c}_i \cdot \phi(r_i + c_i) \tag{2.12}$$

$$\tilde{\nabla}^{iso}\times\phi = \frac{1}{\Delta x}\sum_{i=1}^{N} w_i \hat{c}_i \times \phi(r_i + c_i) \tag{2.13}$$

where c_i represents the set of connecting vectors (discrete velocities in LBM framework) on the lattice with $i = \{1, 2, ...N\}$ and N is the total number of neighbouring points in the stencil. In addition, weights w_i are determined from the set of constraints given below.

$$\sum_{i}^{N} w_i = 1, \quad \sum_{i}^{N} w_i c_{i\alpha} = 0, \quad \sum_{i}^{N} w_i c_{i\alpha} c_{i\beta} = A\delta_{\alpha\beta}, \sum_{i}^{N} w_i c_{i\alpha} c_{i\beta} c_{i\kappa} c_{i\eta} = B\Delta_{\alpha\beta\kappa\eta}$$

where $\delta_{\alpha\beta}$ and $\Delta_{\alpha\beta\kappa\eta}$ are the second and fourth order isotropic Kronecker deltas, respectively. The above constraints on weights ensure that the resulting discrete operators are isotropic at the leading order.

A detailed study on the anisotropy of schemes in multi-dimensions can be found in the work of Lele [37] where the differencing errors are analyzed in terms of phase error (dispersion) and anisotropy (in multi-dimensions). Arguing from the isotropic nature of acoustic waves unlike the entropy and vorticity waves which are highly directional, Tam and Webb [38] had also addressed the issue of anisotropy in finite difference CFD codes (1992) and developed finite difference schemes for computational acoustics that have the same dispersion relations as original PDE's.

It is well known that the process of discretization can lead to loss of information about the original problem and its structure because of the inability of discrete operators to retain all the properties and symmetries of their continuum counterparts. In one of the earlier works, Bochev and Hyman [39] have addressed this issue via mimetic discretizations to develop compatible or mimetic algebraic models that can mimic fundamental properties of continuum equations including conservation laws. In connection to the numerical solution of Maxwell's equation and magnetic field diffusion, Hyman and Shaskov [40] have also demonstrated that construction of discrete analog of differential operators in a mimetic framework satisfies the continuum "divergence-gradient-curl" identities and the theorems of vector and tensor calculus in the discrete limit. The numerical solutions obtained using this approach were found to be physically consistent, retain the symmetries and mimic the fundamental properties of the original underlying physical problem.

2.5 Summary of literature

From all the works discussed above, it is clearly evident that there exists a number of methodologies for the numerical integration of diffusion equation including conventional explicit and implicit finite difference methods, super time-stepping schemes and delayed difference methods. The super-time stepping schemes and delayed difference approach are very attractive because their time step is larger than the conventional explicit methods while incurring only a slight loss in accuracy. At the same time, these methods retain the ease of implementation of explicit methods and are therefore straight forward to parallelize. Nevertheless, these methods are fraught with their own issues like inability to damp high frequency error for the STS method and the occurrence of oscillatory solution pattern in the delayed difference method. Hence, it is evident that we need improvements in these methods to overcome these issues.

Isotropic differential operators which respect isotropy and symmetries of the continuous operators also provide an improvement in CFL criteria over the conventional central difference discretization when coupled with an explicit time-integration approach. The performance of such a solver will however depend on various factors including the computational complexity of isotropic laplacian operator and the overall improvement in CFL.

With regard to the methods employed for solution of multi-dimensional diffusion equations, the existing methodologies need improvements to enhance the CFL criteria for explicit schemes. The use of larger time steps is justifiable from the fact that solution of diffusion equation is very smooth and changes slowly and therefore larger time steps are desirable for computationally efficient algorithms without significant deterioration in accuracy.

2.6 Objective of the current work

The main objectives of the current work are as follows:

1. Develop an algorithm for efficient solution of diffusion equation by improving the CFL stability criteria over classical explicit finite difference schemes.

2. Study the effect of spatial discretization on stability through eigenvalue analysis of differential operators.

3. Assessment of effectiveness and accuracy of the current approach through numerical experiments involving different boundary conditions and comparing the results with the corresponding analytical solutions.

4. Test the parallel scaling and performance of the current algorithm on state-of-the-art HPC systems.

Furthermore, we demonstrate the effectiveness of the algorithm when implemented in a pseudo-transient framework to obtain solution of Poisson's equation. We estimate the asymptotic convergence rate by performing an eigenvalue analysis of the amplification matrix. The theoretical analysis is backed with results obtained from numerical experiments and the performance of current approach is compared against standard iterative solvers.

Chapter 3

One-dimensional Transient Diffusion Solver

3.1 Introduction

In this chapter, we present discretizations for the 1D diffusion equation. FTCS discretization was already introduced in Chapter 1 and the stability criteria for this approach when applied to diffusion equation were also discussed in Chapter 1. In the following, we first consider the delayed difference scheme for diffusion equation and discuss the desirable features and shortcomings associated with both of these methods. Based on the study of these two algorithms, we subsequently present a weighted scheme, obtained by taking a convex combination of FTCS and delayed difference scheme. This method retains the desirable properties of both the above methods and provides an improvement in stability over the individual approaches. Our emphasis will be on investigating the performance of algorithms for large problems for which the data size usually exceeds the sum of all the caches on a representative modern workstation.

3.2 Delayed Difference Scheme

To begin with, we briefly review the delayed difference scheme [14, 19] and highlight the occurrence of a checker board instability for certain conditions while using the scheme. We consider the unsteady, one-dimensional diffusion equation with constant diffusivity. This is given as

$$\frac{\partial \phi}{\partial t} = \kappa \nabla^2 \phi. \tag{3.1}$$

A typical finite difference approximation of the above equation involving forward in time and central in space discretization results in a discrete equation of the form

$$\phi_i^{n+1} = \phi_i^n + \frac{\kappa \Delta t}{\Delta x^2}\left(\phi_{i+1}^n - 2\,\phi_i^n + \phi_{i-1}^n\right). \tag{3.2}$$

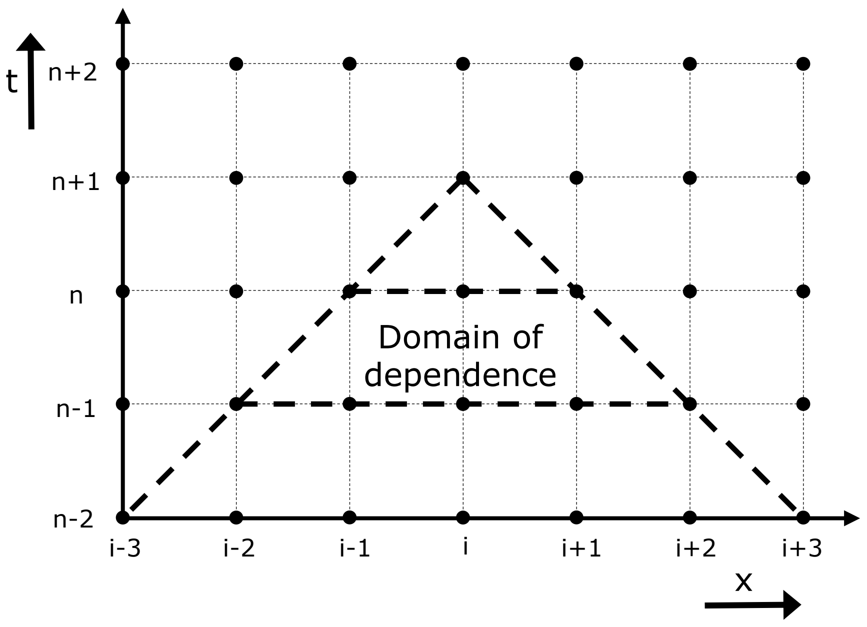


Fig. 3.1 Domain of dependence and stencil for delayed difference scheme

Here, i ranges from 0 to N in a domain of size $0 \leq x \leq 1$ and hence, $\Delta x = 1/N$. It may be noted that the evolution of the above equation will be stable only when the CFL number (α), defined as $\alpha = \kappa\left(\Delta t/\Delta x^2\right)$, is less than 0.5. In other words, for a given grid spacing, Δx, the maximum allowable time step is $\Delta t = \Delta x^2/2\kappa$. The delayed difference scheme, relaxes this stringent constraint by considering a wider stencil that allows for a larger time step. At the same time, the method compensates for the error arising from wider stencil by estimating the temporal derivative at an earlier time. The essential rationale behind such a consideration can be understood by observing the domain of dependence for a typical explicit scheme. As shown in the Fig.3.1, the domain of dependence is a pyramid of points for an explicit method. From this, one can envisage a general explicit scheme wherein the temporal derivative for updating values at each point, i, and time level, n is extrapolated from any earlier time level 'n-m'. Such a general scheme can be written as

$$\phi_i^{n+1} = \phi_i^n + \frac{\alpha}{(m+1)^2}\left[\phi_{i+m+1}^{n-m} + \phi_{i-m-1}^{n-m} - 2\,\phi_i^{n-m}\right]. \tag{3.3}$$

The benefit of such a representation can be inferred from the Von-Neumann stability analysis of Eq.(3.3) that gives the following relation for the error amplification factor (λ).

$$\lambda^{m+1} - \lambda^m + \frac{4\alpha}{(m+1)^2}\sin^2\left(\frac{(m+1)\mathrm{k}\Delta x}{2}\right) = 0. \tag{3.4}$$

For the particular case of m = 1, we arrive at the value of λ to be

$$\lambda = \frac{1}{2}\left(1 \pm \sqrt{1 - 4\,\alpha\,\sin^2(\mathrm{k}\Delta x)}\right), \tag{3.5}$$

wherein for the scheme to be stable, λ needs to be less than 1 and so should be α. Interestingly, the CFL limit of the delayed scheme (m = 1) is two times the value obtained for the general FTCS method (m = 0). The corresponding difference in the accuracies of the two methods can be inferred from the effective PDEs obtained through the Cauchy-Kowalewski procedure detailed in Appendix B. For a general scheme, the effective PDE is given as

$$\frac{\partial\phi}{\partial t} = \kappa\frac{\partial^2\phi}{\partial x^2} + \frac{\kappa\Delta x^2}{2}\frac{\partial^4\phi}{\partial x^4}\mathscr{I}^{\mathrm{CD2}}(m,\alpha) + O(\Delta x^4, \Delta t^2), \tag{3.6}$$

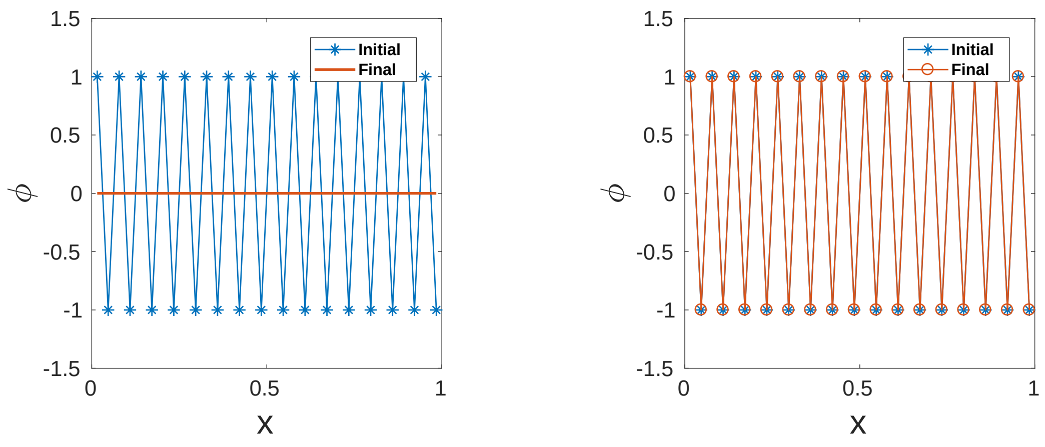
where the pre-factor associated with the biharmonic operator is

$$\mathscr{I}^{\mathrm{CD2}}(m,\alpha) = \left\{\frac{(m+1)^2}{6} - \alpha(2m+1)\right\}. \tag{3.7}$$

From the above equation, the trade-off in using the spatial "mimetic" stencil is obvious. The wider stencil has a lower accuracy as $|\mathscr{I}^{\mathrm{CD2}}(1,\alpha)| > |\mathscr{I}^{\mathrm{CD2}}(0,\alpha)|$ in the parameter range $0 \le \alpha \le 1/2$ where both the schemes are stable. Despite this minor drawback, the delayed schemes are still viable owing to its increased stability and the enhanced parallel efficiency. The latter arises from the possibility of improved overlap between data communication and computation.

Despite its many favourable features, a major drawback of the above delayed difference scheme is the occurrence of checker-board instability for certain conditions. This can be easily exemplified from the solution of a simple periodic diffusion system with the following initial condition.

$$\phi_i(t=0) = (-1)^i. \tag{3.8}$$



(a) FTCS **(b)** Delayed

Fig. 3.2 Checkboard instability associated with delayed difference schemes

Figure 3.2 shows the solution obtained by using both the FTCS and delayed difference schemes for the above initial condition. Evidently, the original high frequency information of the initial condition is perpetually maintained in the delayed difference scheme whereas this is correctly damped out in the FTCS scheme. This behaviour is essentially due to the odd-even decoupling that happens during derivative estimation in the delayed scheme. In the present work, we propose a remedy to the undesirable oscillations by considering a weighted averaging scheme which is described in the ensuing section.

3.3 Weighted Delayed difference

It is evident that both the delayed and the FTCS schemes have their respective favourable features. While the former allows for larger time steps, the latter is effective in wiggling out high frequency errors. Such an observation leads to the idea of a hybrid scheme that is formed by combining the FTCS and delayed difference schemes. Interestingly, we show that a weighted combination of FTCS and the delayed methods has many more desirable properties including further increase in the CFL limit. The new weighted method essentially generalises the scheme of Eq. 3.3 further by allowing the derivative at time level ‘n’ to be extrapolated from more than one time levels. In the present work, we consider a weighted scheme between time levels ‘n’ and ‘n-1’ which is of the form

$$\phi_i^{n+1} = \phi_i^n + w_1\alpha\left(\phi_{i+1}^n + \phi_{i-1}^n - 2\phi_i^n\right) + w_2\frac{\alpha}{4}\left(\phi_{i+2}^{n-1} + \phi_{i-2}^{n-1} - 2\phi_i^{n-1}\right), \tag{3.9}$$

where $w_1 + w_2 = 1$, $w_1 \geq 0$, and $w_2 \geq 0$. To understand the benefits of this scheme, we first examine the stability of this scheme via the Von-Neumann stability analysis. The resulting characteristics equation is given as

$$\lambda^2 - \lambda\left(1 - 4w_1\alpha z\right) + 4w_2\alpha z(1-z) = 0, \tag{3.10}$$

where $z = \sin^2\left(\frac{k_x\Delta x}{2}\right)$ and it ranges between 0 and 1. For stability, the roots of this quadratic equation should lie within a unit circle. This is guaranteed when the coefficients of the equation satisfy the necessary and sufficient conditions given below. For detailed derivation of these conditions, refer to Appendix A.

$$\begin{aligned} |4w_2\alpha z(1-z)| &\leq 1, \\ 4w_1\alpha z &\leq 4w_2\alpha z(1-z) + 2, \\ -w_1 z &\leq w_2 z(1-z). \end{aligned} \tag{3.11}$$

Here, w_1, w_2, z, and α are all positive and the first equation can be modified as

$$4w_2\alpha\,\mathrm{Max}(z(1-z)) \leq 1. \tag{3.12}$$

Since $\mathrm{Max}(z(1-z))$ is 0.25, we can obtain the first condition for stability as follows

$$\alpha\, w_2 \leq 1. \tag{3.13}$$

In the second equation, one can observe that when z is maximum (=1) on the l.h.s, the first term on the r.h.s is zero. This correspondingly yields the condition that

$$\alpha\, w_1 \leq \frac{1}{2}. \tag{3.14}$$

Thus, we have the sufficient conditions for stability as

$$\alpha \leq \frac{3}{2} \;\;\&\;\; w_1 \leq \frac{1}{3} \tag{3.15}$$

and the final scheme to be

$$\phi_i^{n+1} = \phi_i^n + \frac{\alpha}{3}\left(\phi_{i+1}^n + \phi_{i-1}^n - 2\phi_i^n\right) + \frac{\alpha}{6}\left(\phi_{i+2}^{n-1} + \phi_{i-2}^{n-1} - 2\phi_i^{n-1}\right) \tag{3.16}$$

Note that the stability limit of the new scheme is $\alpha \leq 3/2$. Incidentally, this improvement has been obtained by considering a weighted averaging of two methods that have lower stability limit and this is the central message of this work. We now assess other features of the present weighted scheme, with $w_1 = 1/3$ and $w_2 = 2/3$, in comparison to the other methods.

3.3.1 Low Wave-number Asymptotics

One of the important factors that needs to be considered while choosing an explicit scheme is its ability to handle/control low frequency errors that normally requires large number of iterations to converge. In this regard, we now assess the decay behaviour of various modes by expanding ϕ through the following periodic basis functions.

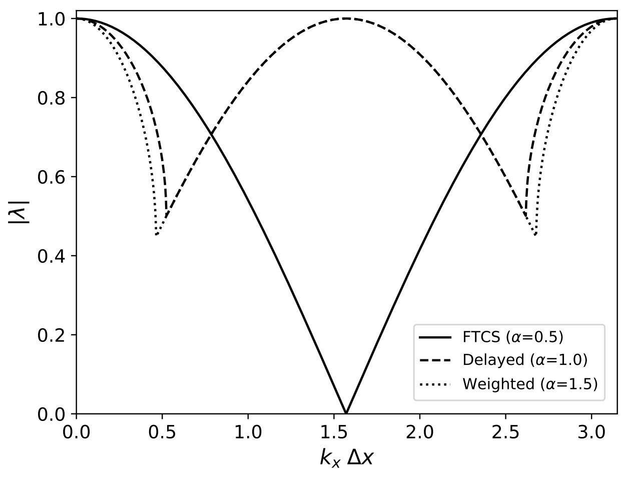


Fig. 3.3 $|\lambda|$ vs k$_x\Delta x$.

$$\phi_j^n \sim \lambda^n \exp\left(Ijk_x\Delta x\right). \tag{3.17}$$

This essentially leads to the characteristics equation, Eq. 3.10, whose formal solution is given as

$$\lambda = \frac{(1-4w_1\alpha z) \pm \sqrt{1 - 8w_1\alpha z + 16w_1^2\alpha^2 z^2 - 16w_2\alpha z(1-z)}}{2}. \tag{3.18}$$

In the low wave number limit, this solution reduces to

$$\lambda \approx 1 - \frac{\alpha k_x^2}{N^2} \quad \& \quad \lambda \approx w_2 \alpha k_x^2 \Delta x^2. \tag{3.19}$$

The second mode written above is an unwanted spurious mode having no connection with the original continuous partial differential equation. It is an artefact of the discretization scheme and in fact, it is well known that multi-step finite difference methods using central difference scheme usually result in such spurious modes. The corresponding error is often of the same order of magnitude as that of the truncation error and they vanish with a rate equal to the order of the scheme for any numerical method as long as it is stable. Even in the present scheme, the oscillations are of second order and they disappear up on grid refinement. Hence, this is not a matter of big concern.

On the other hand, we gain significant information from the first physical mode. Specifically, we look at the damping rate of the mode with lowest wave number, $k_x = \pi$. This is given as

$$\lambda \approx 1 - \frac{\alpha \pi^2}{N^2}. \tag{3.20}$$

This decay rate formula is same for all the methods that have been considered here namely: FTCS, delayed, and weighted schemes. Note that larger the CFL number, lower would be the error magnification factor and this clearly indicates that the weighted scheme should operate better on low wave number modes as it allows for the highest value of CFL (1.5). Figure 3.3 shows the magnitude of the largest eigenvalue for each method evaluated at their largest available CFL. Evidently, the damping rate is lowest for the current weighted scheme when we approach the high and low wavenumber limits and this reflects on its better convergence capabilities.

3.3.2 Effective Scheme

Apart from convergence, understanding the implication of the present weighted scheme on the temporal accuracy of field evolution is also very essential. In this regard, we once again resort to the Cauchy-Kowalewski procedure (refer to Appendix B) for obtaining the effective PDE that is being solved by the current implementation. For the present weighted scheme with $w_1 = 1/3$ and $w_2 = 2/3$, the effective PDE simplifies as

$$\frac{\partial \phi}{\partial t} = \kappa \frac{\partial^2 \phi}{\partial x^2} + \frac{\kappa \Delta x^2}{2} \left(\frac{1}{2} - \frac{7}{3} \alpha \right) \frac{\partial^4 \phi}{\partial x^4} + O(\Delta x^4, \Delta t^2). \tag{3.21}$$

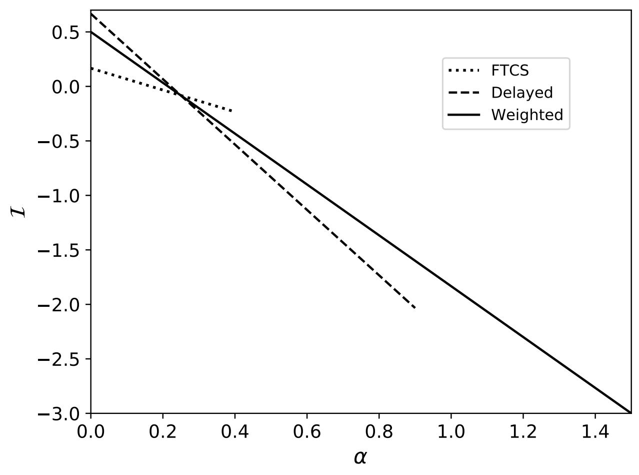


Fig. 3.4 Biharmonic prefactor for different methods

The variation of the biharmonic pre-factor with CFL for different schemes has been presented in Fig. 3.4. Generally, the pre-factor indicates the ability of the discrete formulation to effectively mimic the original PDE. In other words, lower the magnitude of the pre-factor, closer will be the effective PDE to the original one and lower would be the discretization error. From Fig. 3.4, it is evident that the pre-factor of the FTCS method is generally lower in the range of its stability i.e. $\alpha \leq 0.5$. Within the same range, the delayed difference scheme has the highest pre-factor value and the weighted scheme has values in between the FTCS and the delayed schemes. At their highest CFL, weighted scheme has a large pre-factor value as compared to the delayed scheme and this would typically indicate a significant discretization error. However, this might not pose a big threat in a cumulative sense as the weighted scheme would require less number of steps in any time marching procedure. Note that two steps of weighted scheme would cover three steps of the delayed method or six steps of the FTCS scheme. Hence, it is difficult to arrive at a clear picture of the discretization error using the effective PDE and in this regard, we now evaluate these methods through numerical experiments.

3.4 Numerical Experiments

Apart from understanding the discretization error pattern, it is also essential to estimate the actual speed-up one may achieve by using the delayed and weighted schemes. From the critical CFL number, one would anticipate two times speed from the delayed scheme over FTCS and three times speed-up from the weighted scheme. However in reality, much of the speed-up depends on the number of mathematical operations involved in a method, pattern of read/write synchronization, buffer size of the compute system, and more importantly, the data transfer speed. In order to assess these factors comprehensively, we now consider different scenarios of 1D diffusion in this section.

3.4.1 Periodic Test Case

We begin our analysis with a sample 1D diffusion problem with periodic domain of size, $L = 1$. The initial field distribution is given as a combination of low and high frequency information as follows and is also represented in Fig. 3.5.

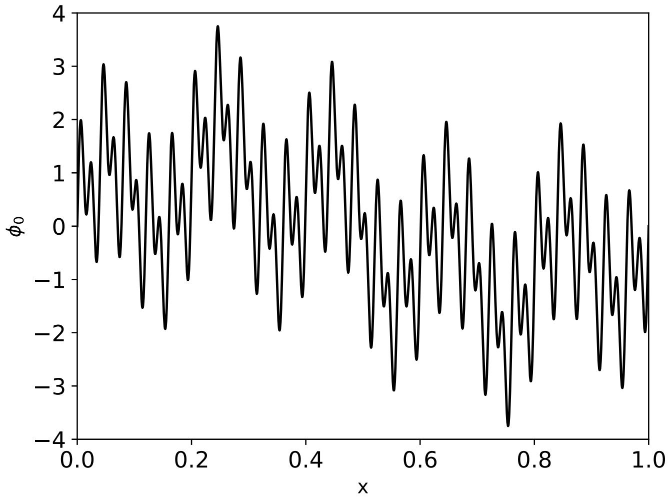


Fig. 3.5 Initial ϕ distribution

$$\phi_0 = a_1 sin(2\pi x) + a_2 sin(10\pi x) + a_3 sin(50\pi x) + a_4 sin(100\pi x) \tag{3.22}$$

The corresponding analytical solution for the evolution of ϕ in time is given as

$$\begin{aligned}\phi(t) \quad = \quad & a_1 e^{-4\pi^2\kappa t} sin(2\pi x) + a_2 e^{-100\pi^2\kappa t} sin(10\pi x) \\ & + a_3 e^{-2500\pi^2\kappa t} sin(50\pi x) + a_4 e^{-10000\pi^2\kappa t} sin(100\pi x)\end{aligned} \tag{3.23}$$

For the sake of simplicity, the coefficients a_1, a_2, a_3, and a_4 have been considered here to be unity. The numerical software for all the three solvers have been written in C++ language and were compiled using gnu compilers (gcc version 9.2.0) with -O3 optimization. The simulations were run on a single core of AMD workstation equipped with AMD Ryzen Threadripper 2970WX processor of 3.0GHz and 128 GB RAM.

Here, we perform the analysis for two different ranges of problem size: one when the problem being solved completely fits within the L3 cache of the system and the other when it exceeds the L3 cache. For the first case, we record the CPU-time (in seconds) taken by different solvers to perform time integration starting from $t = 0$ to $t = 0.1 \times t_{\text{diff}}$, where $t_{\text{diff}} = L^2/\kappa$, for a grid resolution ranging from $N = 512$ to $N = 65536$. CFL values used in the simulations for different solvers are given in Table 3.1.

Solver	CFL
FTCS	0.49
Delayed	0.99
Weighted	1.49

Table 3.1 CFL values used for simulations for different solvers.

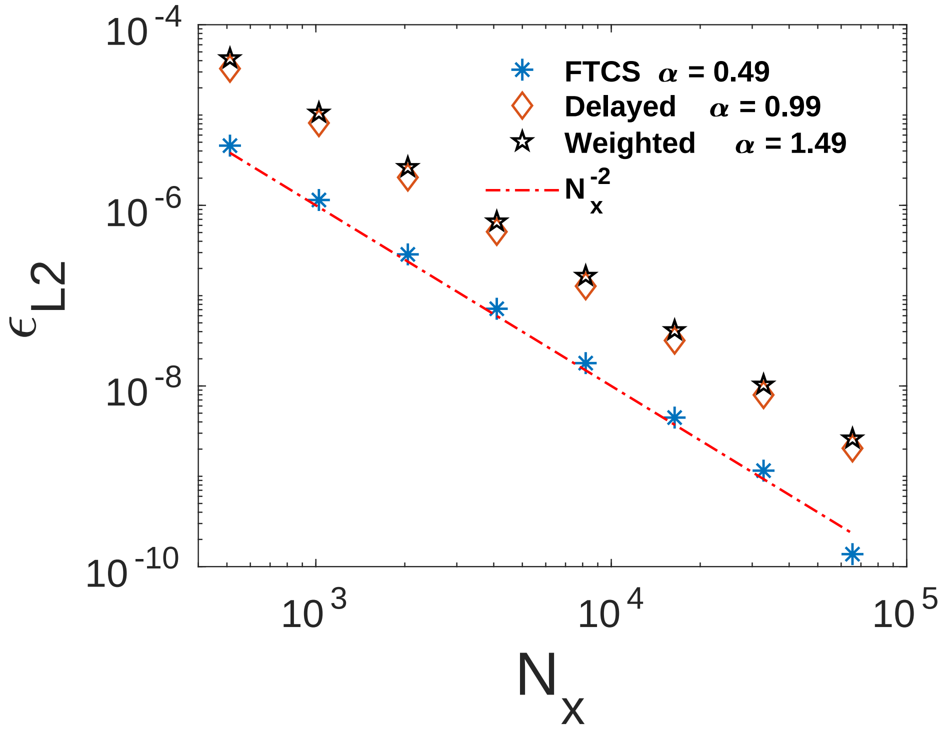


Fig. 3.6 Error analysis for 1D Periodic test case using different explicit schemes. L_2 norm of error with different grid resolution N_x for FTCS, delayed and weighted schemes. The red solid line gives the ideal scaling for second-order convergence.

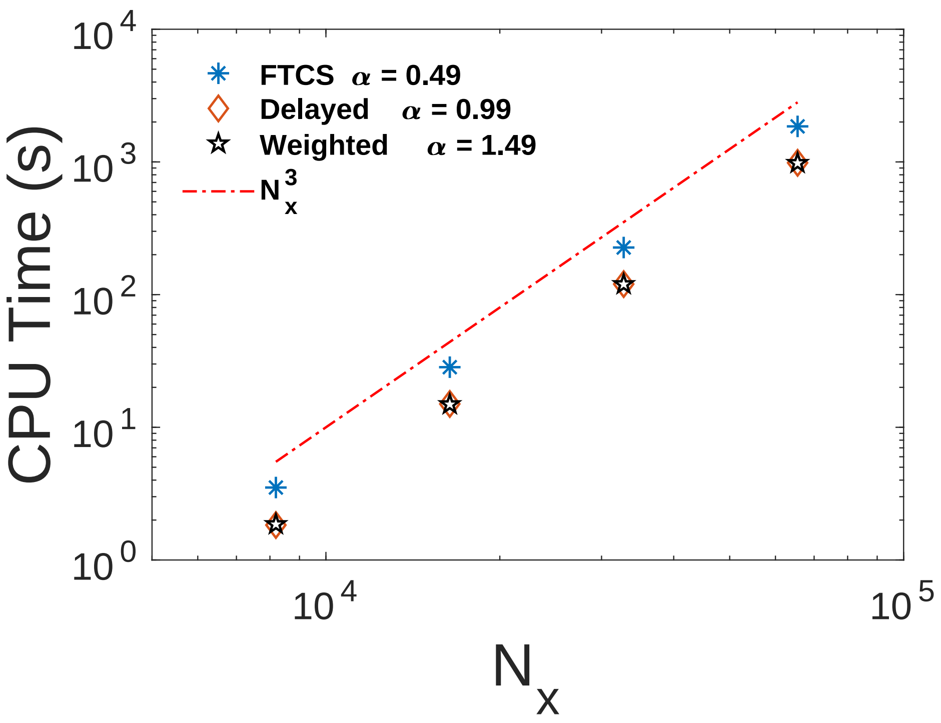


Fig. 3.7 Computational cost analysis for 1D Periodic test case using different explicit schemes. CPU time for $0.1\times t_{\text{diffusion}}$. Computational cost in terms of CPU time follows the expected scaling of N_x^3.

Fig. 3.6 shows the errors obtained for FTCS, delayed, and weighted schemes. From the obtained results, we can see that all the methods show second-order convergence. However, the weighted scheme has a slightly larger error as compared to other methods owing to the larger values of time-step used in the simulations. In Fig. 3.7, we have plotted the computational cost measured in terms of CPU time as a function of grid point count N_x. For convenience, the CPU times (in seconds) have also been listed in Table 3.2. It can be seen that for all the methods, the CPU time, to arrive at the final time, scales as N_x^3 with the grid refinement. Also, the delayed and weighted schemes provide improvement in CPU time over FTCS owing to their larger CFL. However, the weighted scheme does not show any improvement in CPU time over the delayed scheme although the weighted scheme has been solved with a higher CFL value. In order to understand the reason for this behaviour, we now look at the implementation of these algorithms in a typical computer code. A simple implementation of the three schemes are given as

- Update rule for FTCS scheme,

```
for(i=start; i<=end; i++)
    fNew[i] = (1.0-2.0*cfl)*f[i] + cfl*(f[i+1] + f[i-1]);
```

Grid Size (N_x)	FTCS	Delayed	Weighted
512	0.0017	0.00070	0.0010
1024	0.0089	0.0054	0.0038
2048	0.083	0.047	0.036
4096	0.43	0.24	0.23
8192	3.51	1.83	1.85
16384	28.38	14.99	14.88
32768	226.44	120.026	119.62
65536	1853.078	984.76	979.79

Table 3.2 Comparison of CPU-time (in seconds) with different solvers for grid resolution ranging from 512 to 65536. 1-D transient diffusion test case with periodic boundaries.

- Update rule for delayed difference scheme,

```
for(i=start; i<=end; i++)
    fNew[i] = f[i] + cfl*(fOld[i+2] + fOld[i-2] - 2*fOld[i]);
```

- Update rule for weighted difference scheme,

```
for(i=start; i<=end; i++){
    fNew[i] = (1.0-2.0*cfl)*f[i] + w1*(f[i+1] + f[i-1])
        + w2*(fOld[i+2] + fOld[i-2] - 2*fOld[i]); }
```

In the current scenario, where the problem size is small enough to fit within the cache, the performance is governed by the number of floating point computations (arithmetic operations) rather than memory operations. It can be seen from the code snippets presented above that the number of floating point computations is more for weighted scheme (4 multiplications and 5 additions) than delayed scheme (2 multiplications and 3 additions). Hence, the CPU execution time for every update is larger for weighted solver as can be seen from Fig.3.8 where the CPU time per iteration has been plotted for delayed and weighted methods. These results were obtained by performing simulations with 8192, 16384, 32768, and 65536 grid points, respectively. So, the gain in CFL for the weighted scheme is neutralised by the increased number of floating point computations and the overall CPU execution time is almost the same for delayed and weighted schemes as seen earlier in Fig.3.7 and Table 3.2.

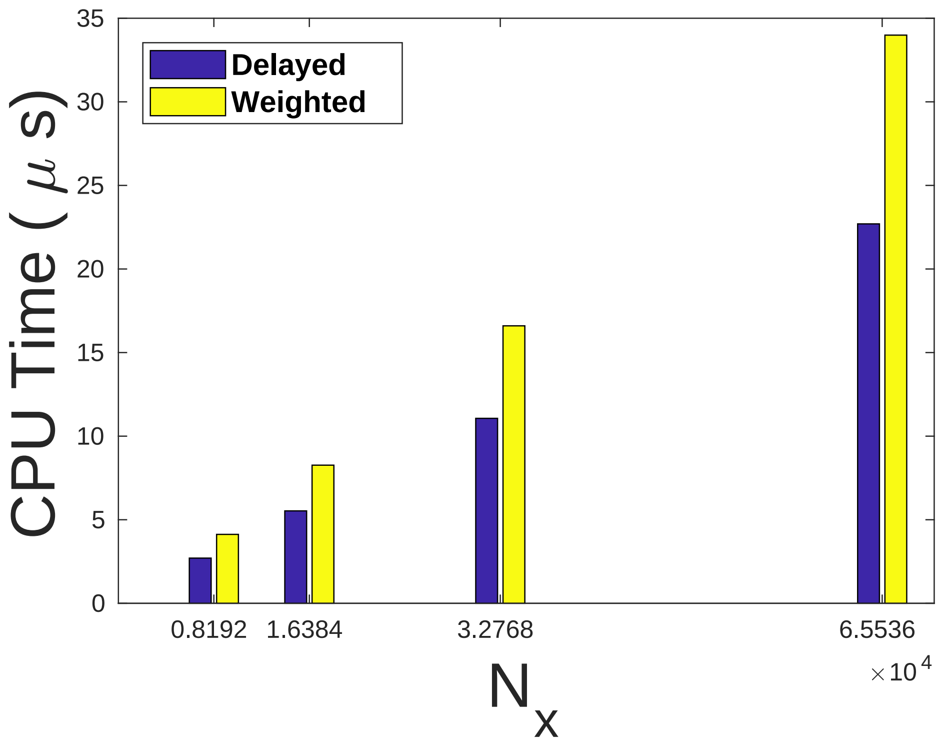


Fig. 3.8 Comparison of CPU execution time (in μs) per time-step update for delayed and weighted schemes when problem size $<$ Cache size.

At this stage, it is worth considering the number of memory operations for each of these methods. It can be seen from the code snippets that for each grid point update, we essentially perform 2 reads and 1 write for the delayed and weighted solvers. To understand this better, consider the delayed scheme where the cost for memory read is effectively associated with reading the variables `f[i]` and `fOld[i+2]` from the main memory as the other variables, `fOld[i-2]` and `fOld[i]`, would already be available in the local cache memory on account of their usage during the previous grid point updates (`fNew[i-2]` and `fNew[i-1]`). The cost of writing is associated with updating the value of `fNew[i]` in the main memory. Similarly, it can be seen that the memory access cost associated with weighted scheme also includes two reads (`fOld[i+2], f[i+1]`) and one write (`fNew[i]`). Hence, both the delayed and weighted schemes effectively involve same number of memory operations during the execution of program. The implications of this equality would be evident in the ensuing part of this section where the behaviour of large size systems have been discussed.

In the scenarios where the problem size exceeds the last level cache, the behaviour of the schemes is notably different. Figure 3.9 shows the overall CPU execution time for the three methods (FTCS, delayed and weighted schemes) at grid resolutions varying from 1 million to 4 millions. Here the computations have been performed from $t = 0$ to $t = 2 \times 10^{-6}s$. For convenience, the numerical values have also been listed in the Table 3.3. From the data, it is evident that the weighted scheme has the best performance as compared to all the other schemes. The behaviour directly stems from the higher CFL value of the weighted scheme. Since the problem size exceeds the cache memory size here, the major cost of computation essentially comes from the read/write latency and the cost associated with the mathematical operations is comparatively smaller. Since the read and write operation counts is same for the above version of the delayed and weighted schemes, the benefit of using higher CFL in the latter makes it a faster approach. This can be corroborated from the CPU execution time per iteration (in micro-seconds) data plotted in Fig.3.10 for grid resolutions varying from one million to four millions. It is evident that the computation cost per every iteration is same for the delayed and weighted scheme owing to the same dominant read/write operation cost and the trivial arithmetic cost. Consequently, with higher CFL number, the weighted schemes arrives faster at the final time.

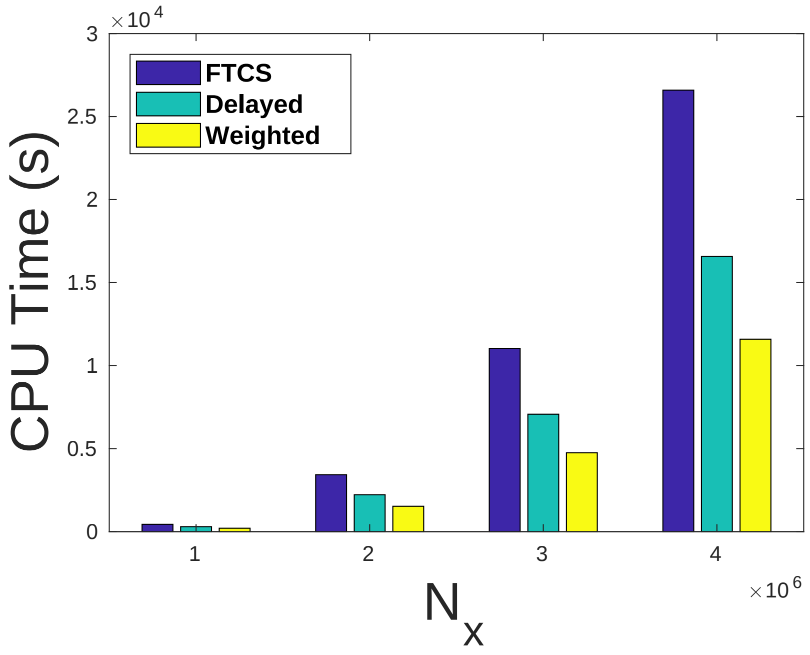


Fig. 3.9 Comparison of overall CPU execution time (in seconds) for FTCS, delayed and weighted schemes with problem size exceeding last level cache size. The simulations were performed with grid resolution 1M, 2M, 3M, 4M.

Grid Size (N_x)	FTCS	Delayed	Weighted
1000000	443.25	301.58	207.70
2000000	3428.11	2222.93	1530.72
3000000	11043.27	7076.48	4748.76
4000000	26595.30	16578.15	11595.56

Table 3.3 Comparison of overall CPU-time (in seconds) with different solvers for grid resolution ranging from 1M to 4M. 1-D transient diffusion test case with periodic boundaries.

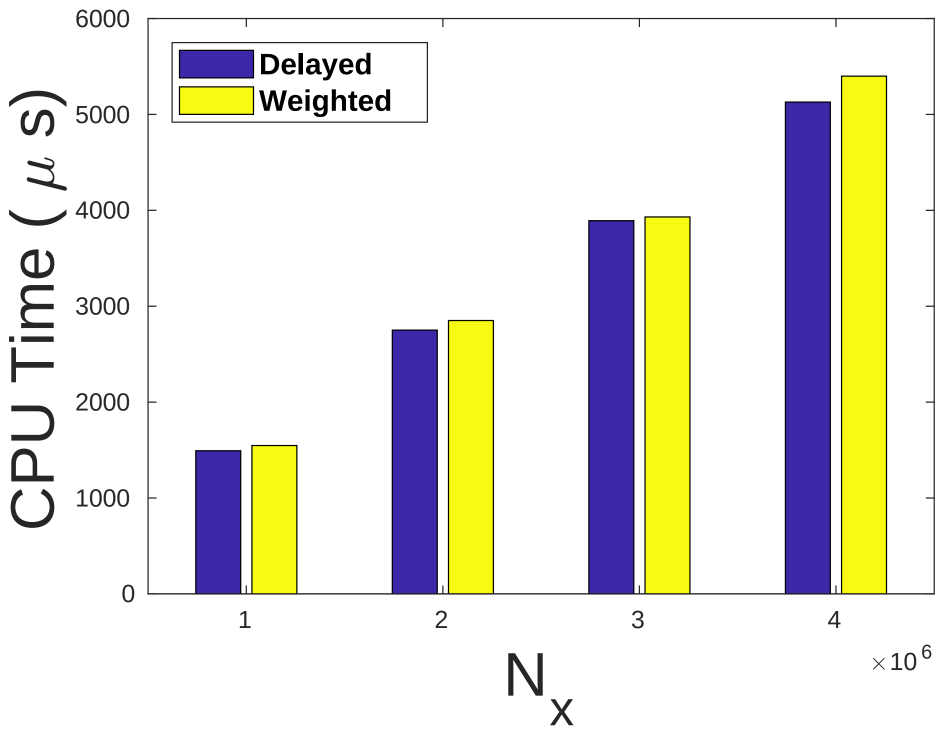


Fig. 3.10 Comparison of CPU execution time (in μs) per time-step update for delayed and weighted schemes when problem size $>$ Cache size

3.5 Code Optimization

The numerical experiments carried out till now were performed with a very basic implementation of the schemes. However, the actual benefit of using the delayed and weighted algorithms can be realized only when we reduce/optimize the number of memory (read and write) operations. In order to facilitate this, we have optimized the basic implementation of the code for all the three solvers (FTCS, delayed and weighted). The optimized code is implemented in such a way that the memory access cost for performing each grid point update involves only 1 read and 1 write operations for every time step. The optimized version

Solver	1M		2M		3M		4M	
	V1	V2	V1	V2	V1	V2	V1	V2
FTCS	443.2	313.5	3428.1	3115.5	11043.2	10645.9	26595.3	24570.9
Delayed	301.5	143.6	2222.9	1199.5	7076.4	3890.6	16578.1	9131.4
Weighted	207.7	146.9	1530.7	1154.2	4748.7	3745.1	11595.5	8801.4

Table 3.4 Comparison of overall CPU-time (in seconds) with different solvers for grid resolution ranging from 1M to 4M. Basic Version (V1) and Optimized Version(V2). 1-D transient diffusion test case with periodic boundaries.

of the weighted code has been provided in Appendix C. In the process of optimizing the code, we have made use of temporary variables. The use of temporary variables allows for data residing in cache to be reused in further computations. This approach essentially minimizes the data fetch from main memory thereby reducing the memory bandwidth pressure that bottlenecks performance for large stencil based computations.

We now present the improvement in CPU time obtained with the optimized version of the code (V2) over the basic version (V1). A comparison of overall CPU execution time for FTCS, delayed and weighted solvers with four different grid resolutions, 1M, 2M, 3M, and 4M, for the basic and optimized version of the code has been tabulated in Table 3.4. The same information is depicted graphically using bar plots in Fig.3.11.

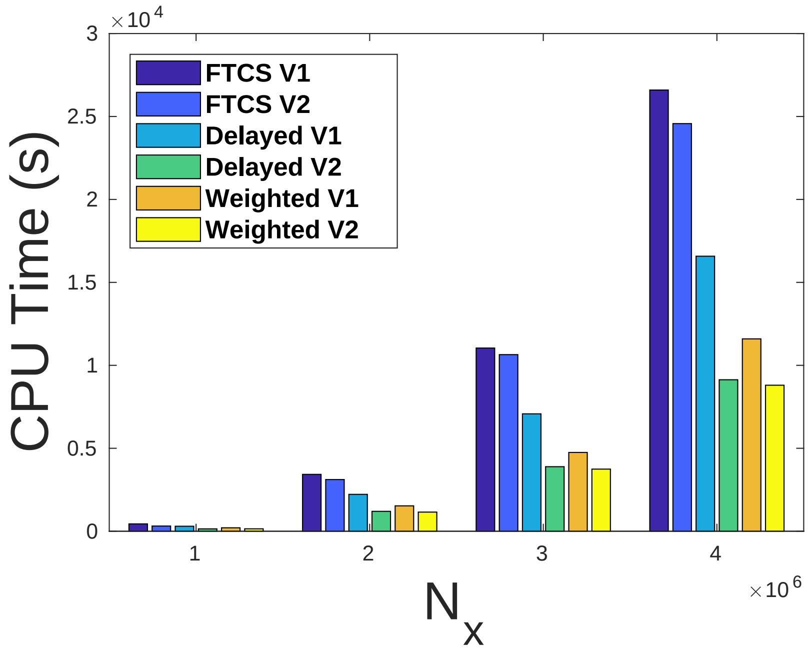


Fig. 3.11 CPU time comparison, Basic (V1) and Optimized (V2) code.

It can be seen that with the optimized implementation there is a significant improvement in overall CPU time at least for delayed and weighted solvers. For the delayed scheme, we have obtained upto 50% reduction in CPU time whereas the reduction is comparatively lesser for the weighted scheme. The reason for the latter behaviour is not very obvious though both the codes have the same read/write operation count. Nevertheless, it shows that effective scheduling of read/write operations in a program can lead to significant benefits from these approaches for large size problems. One could see the benefits evolve in the case of multi-dimensional problems that have been discussed in the ensuing chapter.

3.6 Closure

In this chapter, we have presented a numerical scheme for solving one-dimensional diffusion equation. The weighted difference scheme presented here is an extension of the existing FTCS and the delayed difference schemes. We have carried out a systematic analysis involving different aspects like stability, low wavenumber asymptotics etc., on the new weighted scheme. Subsequently, we have compared the performance of FTCS, delayed and weighted schemes on a simple periodic test case. The major conclusions of this chapter are as follows:

- The delayed difference scheme exhibit checkerboard instability for certain initial conditions whereas the FTCS scheme does not have such spurious behaviour. Subsequently, the weighted difference scheme has been derived as a convex combination of FTCS and delayed scheme and has certain desirable characteristics of both the schemes.
- The maximum allowable CFL limit for weighted scheme is $\alpha \leq 1.5$. This is an improvement by a factor of 3 over the conventional FTCS scheme for which $\alpha \leq 0.5$.
- The asymptotic analysis suggested that the weighted scheme has better damping properties than FTCS and delayed scheme and should operate better on low frequency modes.
- Using the one-dimensional periodic test case, we have shown that the weighted scheme shows a second-order convergence rate by performing a grid resolution study and the CPU time follows expected scaling of N^3.
- The recorded CPU execution time for delayed and weighted schemes show an improvement over the conventional FTCS scheme for small and large sized problems. However, the weighted scheme does not provide any further improvement over delayed scheme when the problem size fits within computer's cache memory.

- Lastly, the weighted scheme outperforms FTCS and delayed schemes when the problem size exceeds the last level cache yielding the actual benefit from CFL improvement. Moreover, the optimized code implementation yielded up to 50% reduction in CPU time for delayed scheme and up to 30% reduction in CPU time for weighted scheme over the basic implementation.

Chapter 4

Multi-dimensional Isotropic Transient Solver

4.1 Introduction

In the preceding chapter, the foundations of delayed and weighted difference schemes were presented and the methods were applied to diffusion equation in one-dimension. In this chapter, we extend the discussion and implementation of weighted scheme to multi-dimensional problems on an orthogonal (cartesian) grid.

In the conventional finite difference scheme, the discretizations are generally carried out along the principal (coordinate) directions on a cartesian grid. Though the implementation of this approach is very simple, it leads to the loss of isotropy in numerical solutions even for simple diffusion problems. An undesirable effect correspondingly is the inverse dependence of stability of explicit schemes (i.e. CFL number) on the dimension of problem i.e. CFL $\propto \frac{1}{D}$. In the present chapter, we first show that the problem of dependence of CFL on dimension can be mitigated by making use of the recently developed lattice differential operators [1, 19, 34] for which the leading order error is isotropic. This is established through a spectral analysis of the discrete diffusion operator which provides the eigenvalue spectrum of central difference and isotropic disretization templates. Next, we show that the weighted approach of delayed and non-delayed difference schemes results in substantial improvement in CFL for 2D and 3D diffusion equation over the conventional FTCS schemes. Finally, we implement the transient solvers in a pseudo transient framework to obtain solution of Poisson equations and compare the performance of the current method against standard iterative solvers.

4.2 The multi-dimensional Diffusion Equation

In order to begin our discussion in multi-dimension, let us first consider the case of diffusion equation in two dimensions which is given as,

$$\frac{\partial \phi}{\partial t} = \kappa \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right), \tag{4.1}$$

where $\phi \equiv \phi(x,y,t)$. The discrete analog of the above equation obtained by employing explicit FTCS discretization can be written as,

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{n} + \kappa \Delta t \left(\frac{\phi_{i+1,j}^{n} - 2\phi_{i,j}^{n} + \phi_{i-1,j}^{n}}{\Delta x^2} + \frac{\phi_{i,j+1}^{n} - 2\phi_{i,j}^{n} + \phi_{i,j-1}^{n}}{\Delta y^2} \right) \tag{4.2}$$

If we substitute the ansatz,

$$\phi_{i,j}^{n} = V^n e^{I\left(\mathrm{k}_x x_i + \mathrm{k}_y y_j\right)} \tag{4.3}$$

the following relation for the error amplification factor, λ, is obtained.

$$\lambda = 1 - 4\alpha \sin^2\left(\frac{\mathrm{k}_x \Delta x}{2}\right) - 4\beta \sin^2\left(\frac{\mathrm{k}_y \Delta y}{2}\right) \tag{4.4}$$

where $\alpha = \frac{\kappa \Delta t}{\Delta x^2}$ and $\beta = \frac{\kappa \Delta t}{\Delta y^2}$. Thus for stability, i.e. $|\lambda| \leq 1$, the following condition,

$$\alpha + \beta \leq \frac{1}{2}. \tag{4.5}$$

needs to be identically satisfied and it imposes a limitation on the time step to be

$$\Delta t \leq \frac{\Delta x^2 \Delta y^2}{2\kappa \left(\Delta x^2 + \Delta y^2\right)}. \tag{4.6}$$

Further, if we assume $\Delta x = \Delta y$ then,

$$\Delta t \leq \frac{\Delta x^2}{4\kappa}. \tag{4.7}$$

This limitation is more restrictive than the one-dimensional case.

Following the same procedure, it can be shown that for the three-dimensional diffusion equation, the CFL crtiteria for stability imposes the following restriction on time-step.

$$\Delta t \leq \frac{\Delta x^2}{6\kappa}. \tag{4.8}$$

This is even more restrictive than the two-dimensional case. Hence, it can be seen that with the conventional FTCS discretization, the maximum allowable time-step becomes smaller with the increase in number of degrees of freedom. The corresponding implication is that the simulations become computationally intractable and this makes it extremely difficult to solve large scientific problems of interest using explicit methods particularly in 3-D. However, knowing the benefits of explicit schemes in the existing massively parallel computing architectures, it is important to develop new multi-dimensional explicit schemes which can overcome this issue.

4.3 The discrete spatial operators

In the last chapter, we have seen that delayed and weighted schemes resulted in an improvement in the stability criteria of discretized diffusion equation in one dimension. Hence, it is of present interest to see how these methods can be utilised to overcome the above stability restrictions of the FTCS method in higher dimensions. Before we proceed on to the actual characterisation, it is important to understand the changes that are required in the spatial discretization strategy for harnessing better efficiency from these methods.

It was shown by Mahan *et al* [19] that for the multi-dimensional extension of delayed scheme, i.e.,

$$\phi_i^{n+1} = \phi_i^n + \kappa \Delta t \Delta \phi_i^{n-1}, \tag{4.9}$$

improvement in stability required further restrictions on the form of discrete operators. It was shown that the multi-dimensional delayed scheme given by Eq.(4.9) has better stability than the conventional FTCS scheme when the discrete Laplacian is isotropic, at least at the leading order error. However, as seen in Chapter 2, the discrete laplacian constructed using the standard central difference stencil do not satisfy this criteria [1, 41, 42]. It is worth recalling that the central difference approximation for the 2D Laplacian is given as,

$$\tilde{\nabla}^2 \approx \nabla^2 + \frac{(\Delta x)^2}{12}\left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4}\right). \tag{4.10}$$

Here, the anisotropy [35] is evident in the Fourier space where the Fourier transform in the polar coordinates exhibits an angular dependence [34],

$$\tilde{\mathbf{L}}(k) = \mathbf{L}(k) + \frac{(\Delta x)^2}{12} k^4 (1 - 2\,\cos^2\theta\,\sin^2\theta). \tag{4.11}$$

4.3.1 Isotropic differential operators

In order to resolve the issue of anisotropy associated with the conventional central difference operators [35], we now make use of the lattice differential operators. We will then understand the implication of these differential operators on the stability by considering their eigen spectra. The detailed description of the methods for deriving the isotropic operators can be found in the papers of Thampi *et al* [1] and Rashmi *et al* [34]. In this approach, the discrete Gradient (∇) and Divergence ($\nabla\cdot$) operators on the lattice are formulated as

$$\nabla^{iso}\phi = \frac{1}{\Delta x}\sum_{i=1}^{N} w_i \hat{c}_i \phi(r_i + c_i) \tag{4.12}$$

and

$$\nabla^{iso}\cdot\phi = \frac{1}{\Delta x}\sum_{i=1}^{N} w_i \hat{c}_i \cdot \phi(r_i + c_i), \tag{4.13}$$

where c_i represents the set of connecting vectors (discrete velocities in LBM framework) on the lattice and N, being the total number of neighbouring points in the stencil. Here, weights w_i are determined from the set of constraints,

$$\sum_{i}^{N} w_i = 1, \quad \sum_{i}^{N} w_i c_{i\alpha} = 0, \quad \sum_{i}^{N} w_i c_{i\alpha} c_{i\beta} = A\delta_{\alpha\beta}, \sum_{i}^{N} w_i c_{i\alpha} c_{i\beta} c_{i\kappa} c_{i\eta} = B\Delta_{\alpha\beta\kappa\eta}, \tag{4.14}$$

where $\delta_{\alpha\beta}$ and $\Delta_{\alpha\beta\kappa\eta}$ are the second and fourth order isotropic Kronecker-delta, respectively. The above constraints on weights ensure that the resulting discrete operators are isotropic at the leading order.

Let us first consider the 2-D case for which the discrete operators are constructed on a lattice as shown in Fig.4.1a. Using Eq.(4.12), we can obtain the explicit expression for the components of discrete gradient operator as

$$\nabla_x\phi = \frac{1}{3\Delta x}\left(\phi_{i+1,j} - \phi_{i-1,j}\right) + \frac{1}{12\Delta x}\left(\phi_{i+1,j+1} - \phi_{i-1,j+1} + \phi_{i+1,j-1} - \phi_{i-1,j-1}\right), \tag{4.15}$$

$$\nabla_y\phi = \frac{1}{3\Delta y}\left(\phi_{i,j+1} - \phi_{i,j-1}\right) + \frac{1}{12\Delta y}\left(\phi_{i+1,j+1} + \phi_{i-1,j+1} - \phi_{i+1,j-1} - \phi_{i-1,j-1}\right). \tag{4.16}$$

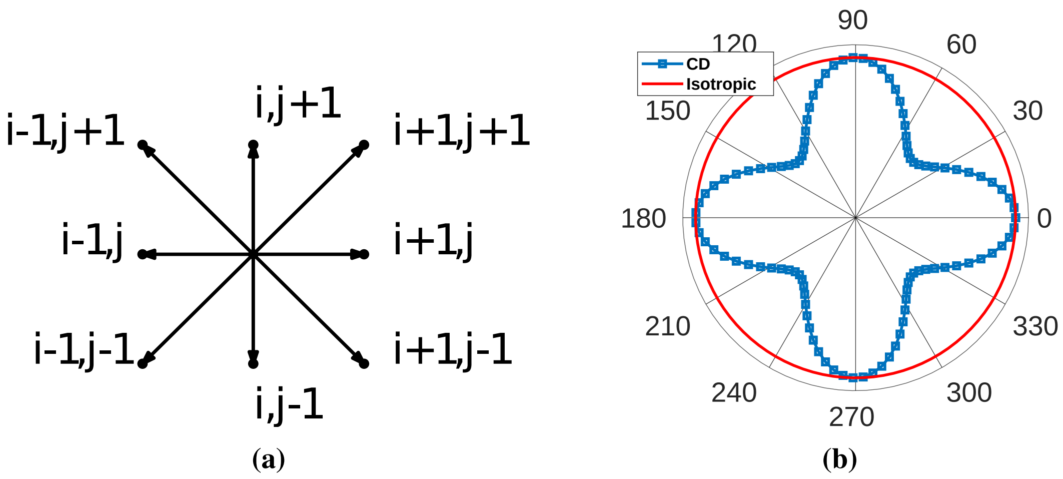


Fig. 4.1 Stencil for Isotropic laplacian in 2D (left) and error distribution for isotropic and CD laplacian (right).

Using the defnition of gradients as in Eq.(4.15) & (4.16) and using $\Delta = \nabla \cdot \nabla$ as the definition of Laplacian, we can formulate the discrete isotropic Laplacian in 2-D as

$$\begin{aligned}
\nabla \cdot \nabla \phi_{i,j} \quad = \quad & \frac{\phi^{n-1}_{i+2,j} + \phi^{n-1}_{i-2,j} + \phi^{n-1}_{i,j+2} + \phi^{n-1}_{i,j-2} - 4\phi^{n-1}_{i,j}}{9\Delta x^2} \\
& - \frac{\phi^{n-1}_{i,j+1} + \phi^{n-1}_{i,j-1} + \phi^{n-1}_{i+1,j} + \phi^{n-1}_{i-1,j} - 4\phi^{n-1}_{i,j}}{9\Delta x^2} \\
& + \frac{\phi^{n-1}_{i+2,j+2} + \phi^{n-1}_{i-2,j+2} + \phi^{n-1}_{i+2,j-2} + \phi^{n-1}_{i-2,j-2} - 4\phi^{n-1}_{i,j}}{72\Delta x^2} \\
& + \frac{\phi^{n-1}_{i+2,j+1} + \phi^{n-1}_{i-2,j+1} + \phi^{n-1}_{i+2,j-1} + \phi^{n-1}_{i-2,j-1} - 4\phi^{n-1}_{i,j}}{18\Delta x^2} \\
& + \frac{\phi^{n-1}_{i+1,j+2} + \phi^{n-1}_{i+1,j-2} + \phi^{n-1}_{i-1,j+2} + \phi^{n-1}_{i-1,j-2} - 4\phi^{n-1}_{i,j}}{18\Delta x^2}.
\end{aligned} \tag{4.17}$$

The leading order truncation error for the above laplacian is plotted in Fig. 4.1b from where it can be seen that the truncation error is isotropic at leading order whereas the error associated with central difference laplacian operator clearly exhibits an angular dependence. The Fourier transform of this Laplacian is

$$\nabla^2(\mathrm{k}_x, \mathrm{k}_y) = -\frac{1}{9}\left[\sin^2(\mathrm{k}_x \Delta x)(\cos(\mathrm{k}_y \Delta y) + 2)^2 \; + \; (\cos(\mathrm{k}_x \Delta x) + 2)^2 \sin^2(\mathrm{k}_y \Delta y)\right], \tag{4.18}$$

and is obviously negative semi-definite which is also true for the Fourier transform of the continuum Laplacian operator.

Similarly in 3-D, we can write the components of gradient vector based on D3Q19 lattice model, shown in Fig. 4.2, as

$$\begin{aligned}\nabla_x\phi &= w_1\left(\phi_{i+1,j,k}-\phi_{i-1,j,k}\right)+w_2\big(\phi_{i+1,j+1,k}-\phi_{i-1,j+1,k}\\ &\quad+\phi_{i+1,j-1,k}-\phi_{i-1,j-1,k}+\phi_{i+1,j,k+1}-\phi_{i-1,j+1,k}+\phi_{i+1,j,k-1}-\phi_{i-1,j,k-1}\big),\end{aligned} \tag{4.19}$$

$$\begin{aligned}\nabla_y\phi &= w_1\left(\phi_{i,j+1,k}-\phi_{i,j-1,k}\right)+w_2\big(\phi_{i,j+1,k+1}-\phi_{i,j-1,k+1}\\ &\quad+\phi_{i,j+1,k-1}-\phi_{i,j-1,k-1}+\phi_{i+1,j+1,k}-\phi_{i+1,j-1,k}+\phi_{i-1,j+1,k}-\phi_{i-1,j-1,k}\big),\end{aligned} \tag{4.20}$$

$$\begin{aligned}\nabla_z\phi &= w_1\left(\phi_{i,j,k+1}-\phi_{i,j,k-1}\right)+w_2\big(\phi_{i+1,j,k+1}-\phi_{i+1,j,k-1}\\ &\quad+\phi_{i-1,j,k+1}-\phi_{i-1,j,k-1}+\phi_{i,j+1,k+1}-\phi_{i,j+1,k-1}+\phi_{i,j-1,k+1}-\phi_{i,j-1,k-1}\big),\end{aligned} \tag{4.21}$$

where $w_1=\frac{1}{18}$ and $w_2=\frac{1}{36}$.

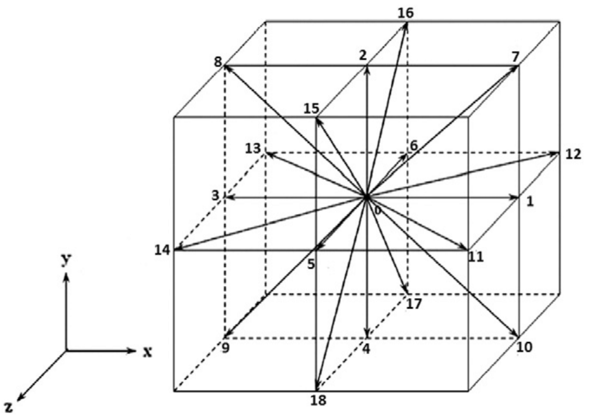


Fig. 4.2 D3Q19 lattice model for computation of discrete gradient in 3D

Discrete isotropic $\nabla\cdot\nabla$ laplacian constructed using above relations has the following form in Fourier space,

$$\begin{aligned}\nabla^2(\mathrm{k}_x,\mathrm{k}_y,\mathrm{k}_z) \;=\; &-\frac{1}{9}\Big[\sin^2(\mathrm{k}_x\Delta x)\;\big\{1+\cos(\mathrm{k}_y\Delta y)+\cos(\mathrm{k}_z\Delta z)\big\}^2\\ &+\sin^2(\mathrm{k}_y\Delta y)\;\{1+\cos(\mathrm{k}_x\Delta x)+\cos(\mathrm{k}_z\Delta z)\}^2\\ &+\sin^2(\mathrm{k}_z\Delta z)\;\big\{1+\cos(\mathrm{k}_x\Delta x)+\cos(\mathrm{k}_y\Delta y)\big\}^2\Big],\end{aligned} \tag{4.22}$$

which also is a negative semi-definite quantity.

4.4 Eigenvalue spectrum of laplacian operators

Having defined the isotropic operators, we now focus our efforts towards characterizing their eigenvalue spectra. This is essential due to the fact that the error amplification factor in an explicit scheme using these operators is closely linked to their eigenvalues. In fact, it will be shown that the maximum allowable time-step in these explicit schemes is inversely proportional to the maximum eigenvalue values of the spatial operators. Thus, smaller the maximum eigenvalue, larger the time step for which the computations would be stable. Correspondingly, we now analyze the eigenvalue spectrum of the two spatial discretization procedures viz. central difference and isotropic discretization in the context of pure diffusion. We also consider an interim case where the central difference Laplcian is considered via the $\nabla \cdot \nabla$ representation. In order to simplify the analysis, we consider a problem with spatial periodicity. The immediate consequence of such a consideration is that the eigenvectors of the system are the Fourier modes themselves and we can obtain an expression for the eigenvalues in a straightforward manner following the general methodology applied for the Von Neumann analysis.

In general, the eigenvalues for a given spatial discretization operator $\mathscr{L}$ in a periodic context can be obtained from

$$\mathscr{L} \cdot e^{I\vec{k}\cdot\vec{x}} = \Omega e^{I\vec{k}\cdot\vec{x}}, \tag{4.23}$$

where Ω are the eigenvalues of the space discretization matrix.

Now, considering the Laplacian in 2D, we obtain the following for the central difference discretization,

$$\begin{aligned}
\mathscr{L} \cdot e^{I\left(\mathrm{k}_x \cdot i\Delta x + \mathrm{k}_y \cdot j\Delta y\right)} &= \kappa \left(\frac{e^{I\mathrm{k}_x\Delta x} - 2 + e^{-I\mathrm{k}_x\Delta x}}{\Delta x^2} + \frac{e^{I\mathrm{k}_y\Delta y} - 2 + e^{-I\mathrm{k}_y\Delta y}}{\Delta y^2} \right) e^{I\left(\mathrm{k}_x \cdot i\Delta x + \mathrm{k}_y \cdot j\Delta y\right)} \\
&= \frac{\kappa}{\Delta x^2} \left\{ 2\cos\left(\mathrm{k}_x\Delta x\right) - 2 + 2\cos\left(\mathrm{k}_y\Delta y\right) - 2 \right\} e^{I\left(\mathrm{k}_x \cdot i\Delta x + \mathrm{k}_y \cdot j\Delta y\right)} \\
&= \frac{2\kappa}{\Delta x^2} \left\{ \cos\left(\mathrm{k}_x\Delta x\right) + \cos\left(\mathrm{k}_y\Delta y\right) - 2 \right\} e^{I\left(\mathrm{k}_x \cdot i\Delta x + \mathrm{k}_y \cdot j\Delta y\right)} \\
&\equiv \Omega\left(\mathrm{k}_x, \mathrm{k}_y\right) e^{I\left(\mathrm{k}_x \cdot i\Delta x + \mathrm{k}_y \cdot j\Delta y\right)}
\end{aligned} \tag{4.24}$$

Thus, we have the following relation for the eigenvalues of central difference laplacian operator.

$$\Omega\left(\mathrm{k}_x, \mathrm{k}_y\right) = \frac{2\kappa}{\Delta x^2} \left\{ \cos\left(\mathrm{k}_x\Delta x\right) + \cos\left(\mathrm{k}_y\Delta y\right) - 2 \right\}, \tag{4.25}$$

which can be written as,

$$\Omega\left(\mathrm{k}_x,\mathrm{k}_y\right) = -\frac{4\kappa}{\Delta x^2}\left\{\sin^2\left(\frac{\mathrm{k}_x\Delta x}{2}\right)+\sin^2\left(\frac{\mathrm{k}_y\Delta y}{2}\right)\right\}. \tag{4.26}$$

It can be seen that the eigenvalues are real, negative and covers the range $\left(-8\kappa/\Delta x^2, 0\right)$. As seen earlier, Eq.(4.4), the error amplification factor, λ, for FTCS discretization of 2-D diffusion equation is given by

$$\lambda = 1-4\alpha\sin^2\left(\frac{\mathrm{k}_x\Delta x}{2}\right)-4\beta\sin^2\left(\frac{\mathrm{k}_y\Delta y}{2}\right) \tag{4.27}$$

where $\alpha=\frac{\kappa\Delta t}{\Delta x^2}$ and $\beta=\frac{\kappa\Delta t}{\Delta y^2}$.
Using Eq.(4.26), Eq.(4.27), and $\Delta x=\Delta y$, we can arrive at the following relation for amplification factor.

$$\lambda = 1+\Omega\Delta t. \tag{4.28}$$

Thus, for stability we have,

$$\left|1+\Omega\Delta t\right| \le 1.$$

$$\implies \Delta t \le \frac{2}{\left|\Omega\right|}. \tag{4.29}$$

Thus, it can be concluded that the maximum allowable timestep for an explicit time integration method varies inversely with the spectral radius of laplacian operator i.e.

$$\Delta t_{\max} \propto \frac{1}{\Omega_{\max}}. \tag{4.30}$$

Following the procedure outlined above, one can also obtain the eigenvalues of $\nabla\cdot\nabla$ central difference laplacian and isotropic $\nabla\cdot\nabla$ laplacian operators. In summary, the eigenvalue spectrum of standard central difference laplacian (Ω_{CD}), $\nabla\cdot\nabla$ central difference laplacian ($\Omega_{\mathrm{CD}}^{\nabla\cdot\nabla}$) and isotropic $\nabla\cdot\nabla$ laplacian ($\Omega_{\mathrm{Iso}}^{\nabla\cdot\nabla}$) are as follows:

$$\Omega_{\mathrm{CD}} = -\frac{4\kappa}{(\Delta x)^2}\left[\sin^2\left(\frac{\mathrm{k}_x\Delta x}{2}\right)+\sin^2\left(\frac{\mathrm{k}_y\Delta y}{2}\right)\right] \tag{4.31}$$

$$\Omega_{\mathrm{CD}}^{\nabla\cdot\nabla} = -\frac{\kappa}{(\Delta x)^2}\left[\sin^2\left(\mathrm{k}_x\Delta x\right)+\sin^2\left(\mathrm{k}_y\Delta y\right)\right] \tag{4.32}$$

$$\Omega_{\mathrm{Iso}}^{\nabla\cdot\nabla} = -\frac{\kappa}{9\,\Delta x^2}\left[\sin^2(\mathrm{k}_x\Delta x)(\cos(\mathrm{k}_y\Delta y)+2)^2\;+\;(\cos(\mathrm{k}_x\Delta x)+2)^2\sin^2(\mathrm{k}_y\Delta y)\right] \tag{4.33}$$

Correspondingly, we compare the eigenvalue distribution for the above three discretizations as a function of phase angle (kΔx) in Fig.4.3. Evidently, the isotropic $\nabla\cdot\nabla$ laplacian has lower magnitude of eigenvalues compared to the standard central difference laplacian and $\nabla\cdot\nabla$ central difference laplacian. This leads to an improvement in the maximum allowable CFL value when associated with conditionally stable explicit time integration methods.

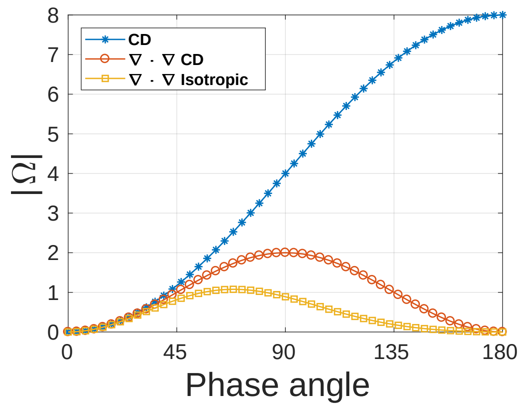


Fig. 4.3 Eigen spectra of different Laplacian operators in 2D on a periodic domain.

We can repeat the above analysis to determine the eigenvalue spectrum for three-dimensional laplacian operator in periodic systems. In this case, the eigenvalue spectrum for standard central difference laplacian, $\nabla\cdot\nabla$ central difference laplacian and isotropic $\nabla\cdot\nabla$ laplacian are obtained as:

$$\Omega_{\text{CD}} = -\frac{4\kappa}{(\Delta x)^2}\left[\sin^2\left(\frac{\text{k}_x\Delta x}{2}\right)+\sin^2\left(\frac{\text{k}_y\Delta y}{2}\right)+\sin^2\left(\frac{\text{k}_z\Delta z}{2}\right)\right] \tag{4.34}$$

$$\Omega_{\text{CD}}^{\nabla\cdot\nabla} = -\frac{\kappa}{(\Delta x)^2}\left[\sin^2(\text{k}_x\Delta x)+\sin^2(\text{k}_y\Delta y)++\sin^2(\text{k}_z\Delta z)\right] \tag{4.35}$$

$$\begin{aligned}\Omega_{\text{Iso}}^{\nabla\cdot\nabla} \quad = \quad & -\frac{\kappa}{9\,\Delta x^2}\Big[\sin^2(\text{k}_x\Delta x)\left\{1+\cos(\text{k}_y\Delta y)+\cos(\text{k}_z\Delta z)\right\}^2 \\ & +\sin^2(\text{k}_y\Delta y)\left\{1+\cos(\text{k}_x\Delta x)+\cos(\text{k}_z\Delta z)\right\}^2 \\ & +\sin^2(\text{k}_z\Delta z)\ \left\{1+\cos(\text{k}_x\Delta x)+\cos(\text{k}_y\Delta y)\right\}^2\Big]\end{aligned} \tag{4.36}$$

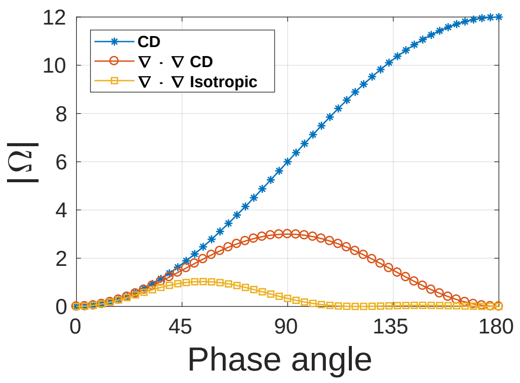


Fig. 4.4 Eigen spectra of different Laplacian operators in 3D on a periodic domain.

Once again, it can be seen from Fig. 4.4 that the spectral radius of the isotropic $\nabla \cdot \nabla$ laplacian is smaller than the central difference laplacian. Since, the maximum allowable time-step for an explicit scheme varies inversely with the spectral radius of the laplacian operator, i.e. $\Delta t \sim \frac{1}{\Omega_{max}}$, it can be expected that explicit schemes constructed with isotropic $\nabla \cdot \nabla$ operators should result in an improvement of time-step restrictions over the conventional standard central difference schemes. In this regard, we now consider the delayed $\nabla \cdot \nabla$ scheme of Mahan *et al* [19] for multi-dimensional diffusion equation and analyze its stability in the following section.

4.5 Delayed $\nabla \cdot \nabla$ solver

The general form of delayed difference solver for multi-dimensional diffusion equation can be written as [19],

$$\phi_i^{n+1} = \phi_i^n + \kappa \Delta t \nabla \cdot \nabla \phi_i^{n-1}. \tag{4.37}$$

Here, the discrete laplacian evaluated at time level 'n-1' is constructed as a composite of discrete isotropic gradient and divergence operators so that the identity $\nabla^2 = \nabla \cdot \nabla$ is satisfied by construction in the discrete form [19]. The resulting discrete isotropic Laplacian operator is negative semi-definite with a trivial null-space: the only eigenvector with a zero eigenvalue is the constant. A distinct advantage of delayed time integration scheme, deriving from the trivial null space of the Laplacian, is that it does not produce spurious checker board modes in two- and three- dimensional spaces [19], unlike what has been seen in the earlier chapter for one-dimensional case. It may be noted that the stencil employed for the computation of $\nabla \cdot \nabla$ Laplacian is larger than those used for same order of accuracy commonly used in finite-difference methods. The implications of this large stencil has a profound effect on the stability of the scheme which we now analyse using the Von Neumann analysis.

We now begin with the analysis in the two-dimensional space where the discrete form of the scheme is given as,

$$\begin{aligned}
\phi_i^{n+1} \quad = \quad & \phi_i^n + \kappa \Delta t \left\{ \frac{\phi_{i+2,j}^{n-1} + \phi_{i-2,j}^{n-1} + \phi_{i,j+2}^{n-1} + \phi_{i,j-2}^{n-1} - 4\phi_{i,j}^{n-1}}{9\Delta x^2} \right. \\
& - \frac{\phi_{i,j+1}^{n-1} + \phi_{i,j-1}^{n-1} + \phi_{i+1,j}^{n-1} + \phi_{i-1,j}^{n-1} - 4\phi_{i,j}^{n-1}}{9\Delta x^2} \\
& + \frac{\phi_{i+2,j+2}^{n-1} + \phi_{i-2,j+2}^{n-1} + \phi_{i+2,j-2}^{n-1} + \phi_{i-2,j-2}^{n-1} - 4\phi_{i,j}^{n-1}}{72\Delta x^2} \\
& + \frac{\phi_{i+2,j+1}^{n-1} + \phi_{i-2,j+1}^{n-1} + \phi_{i+2,j-1}^{n-1} + \phi_{i-2,j-1}^{n-1} - 4\phi_{i,j}^{n-1}}{18\Delta x^2} \\
& \left. + \frac{\phi_{i+1,j+2}^{n-1} + \phi_{i+1,j-2}^{n-1} + \phi_{i-1,j+2}^{n-1} + \phi_{i-1,j-2}^{n-1} - 4\phi_{i,j}^{n-1}}{18\Delta x^2} \right\}. \qquad (4.38)
\end{aligned}$$

Using the ansatz as in Eq.(4.3), we obtain the characteristic equation for the amplification factor λ as,

$$\lambda^2 - \lambda + \frac{\kappa \Delta t}{\Delta x^2} \frac{1}{9} \left[\sin^2(\mathrm{k}_x \Delta x)(\cos(\mathrm{k}_y \Delta y) + 2)^2 \; + \; (\cos(\mathrm{k}_x \Delta x) + 2)^2 \sin^2(\mathrm{k}_y \Delta y) \right] = 0. \quad (4.39)$$

As mentioned in the appendix A, the stability limits can be derived by considering a quadratic equation of the form $x^2 - bx + c = 0$. The neccessary and sufficient stability conditions are satisfied when the roots lie inside the unit circle which leads to the following three conditions,

$$\begin{aligned}
|c| &\leq 1, \\
-b &\leq c + 1, \qquad (4.40) \\
b &\leq c + 1.
\end{aligned}$$

From Eq.(4.39) we observe that $b = 1$ and $c \geq 0$. Thus, the last two inequalities are always satisfied and we have the neccessary and sufficient condition for stability arising only from one condition, $\mid c \mid \leq 1$. Hence, from Eq.(4.39) and Eq.(4.40), we have

$$\left| \frac{\kappa \Delta t}{9\Delta x^2} \left\{ \sin^2(\mathrm{k}_x \Delta x)(\cos(\mathrm{k}_y \Delta y) + 2)^2 \; + \; (\cos(\mathrm{k}_x \Delta x) + 2)^2 \sin^2(\mathrm{k}_y \Delta y) \right\} \right| \leq 1, \qquad (4.41)$$

$$\frac{\kappa \Delta t}{\Delta x^2} \,\mathrm{Max.} \left| \frac{\sin^2(\mathrm{k}_x \Delta x)(\cos(\mathrm{k}_y \Delta y) + 2)^2 \; + \; (\cos(\mathrm{k}_x \Delta x) + 2)^2 \sin^2(\mathrm{k}_y \Delta y)}{9} \right| \leq 1. \qquad (4.42)$$

By comparing with Eq.(4.33), we can see that the term containing the transcendental functions in the above equation is the same expression for eigenvalues of the isotropic Laplacian operators. Thus, its maximum value should be equal to the spectral radius of laplacian operator which in this case is equal to 1.077 (refer to Fig.4.3).

The resulting CFL criteria obtained from the above condition is given by,

$$\alpha \leq 0.928. \tag{4.43}$$

where $\alpha = \frac{\kappa \Delta t}{\Delta x^2}$. Thus it is evident that the CFL stability limit of the delayed $\nabla \cdot \nabla$ scheme is 3.6 times the limit of 2-D FTCS scheme for which $\alpha \leq 0.25$.

In the case of three-dimensions, the final expression for discrete isotropic laplacian based on D3Q19 lattice model can be obtained by making use of Eqns. (4.12), (4.13), (4.19), (4.20) and (4.21). Once again, we perform the von-Neumann analysis for Eq.(4.37) which gives the following relation for amplification factor,

$$\begin{aligned}\lambda^2 - \lambda + \frac{\kappa \Delta t}{\Delta x^2}\frac{1}{9}\Big[\sin^2(\mathrm{k}_x \Delta x)\ \left\{1+\cos(\mathrm{k}_y \Delta y)+\cos(\mathrm{k}_z \Delta z)\right\}^2 \\ +\sin^2(\mathrm{k}_y \Delta y)\ \left\{1+\cos(\mathrm{k}_x \Delta x)+\cos(\mathrm{k}_z \Delta z)\right\}^2 \\ +\sin^2(\mathrm{k}_z \Delta z)\ \left\{1+\cos(\mathrm{k}_x \Delta x)+\cos(\mathrm{k}_y \Delta y)\right\}^2\Big] = 0.\end{aligned} \tag{4.44}$$

By repeating the analysis described above for the 2D scenario, we obtain the CFL stability criteria to be $\alpha \leq 0.9$. Once again, the delayed $\nabla \cdot \nabla$ scheme shows significant improvement in the CFL value. Infact, the improvement here is by a factor of about 6 over the 3D FTCS scheme for which $\alpha \leq \frac{1}{6}$.

Dimension	FTCS	$(\nabla \cdot \nabla)_{CD}$	$(\nabla \cdot \nabla)_{Iso}$
1D	1/2	1	**1.0**
2D	1/4	1/2	**0.928**
3D	1/6	1/3	**0.9**

Table 4.1 CFL criteria for FTCS (Column II), Delayed $(\nabla \cdot \nabla)_{CD}$ (Column III) and Delayed $(\nabla \cdot \nabla)_{Iso}$ (Column IV).

The above stability procedure has also been repeated here for the $(\nabla \cdot \nabla)_{CD}$ operator. Incidentally, the CFL gain is not as high as from the isotropic laplacian and this is evident from Table 4.1 where CFL criteria for different cases in 1-D, 2-D and 3-D have been tabulated. It can be noted that the CFL for conventional central difference operators varies inversely with dimension D i.e. $\alpha \sim \frac{1}{D}$. However, the use of isotropic differential operators alleviates any such dependency of CFL on dimension and we see that for isotropic $\nabla\cdot\nabla$ operators, $\alpha \sim 1$.

4.6 Weighted $\nabla\cdot\nabla$ solver

Having observed the improvement provided by the delayed difference scheme, we now focus on our weighted difference scheme to see if it is able to replicate its performance as observed in the 1D scenario. We recall that the general form of the weighted difference scheme is given as

$$\phi_i^{n+1} = \phi_i^n + \kappa\,\Delta t\left\{\underbrace{w\,\nabla\cdot\nabla\phi_i^n}_{non\;delayed} + \underbrace{(1-w)\,\nabla\cdot\nabla\phi_i^{n-1}}_{delayed}\right\}. \tag{4.45}$$

The important features of the above scheme are as follows:

- Convex combination of non-delayed and delayed scheme.
- Isotropic discrete differential operators.
- Discrete laplacian constructed as divergence of gradient i.e. $\nabla^2 \equiv \nabla\cdot\nabla$.

Effective PDE associated with Eq.(4.45) can be derived by performing a Taylor series expansion and replacing all the time and mixed derivatives with spatial derivatives which gives the effective differential equation at the leading order as,

$$\frac{\partial\phi}{\partial t} = \kappa\nabla^2\left[1+\Delta x^2\left\{a-\alpha\,(1-w)-\frac{\alpha}{2}\right\}\nabla^2\right]\phi + O(\Delta x^4,\Delta t^2). \tag{4.46}$$

Also, the characteristic equation for the error amplification factor, λ, obtained through von-Neumann stability analysis in 2-D (analog of Eq.(4.45)) is

$$\lambda^2-\lambda\left[1-\frac{\alpha w}{9}\left\{\sin^2 A\,(2+\cos B)^2+\sin^2 B\,(2+\cos A)^2\right\}\right] \\ +\frac{\alpha\,(1-w)}{9}\left\{\sin^2 A\,(2+\cos B)^2+\sin^2 B\,(2+\cos A)^2\right\}=0, \tag{4.47}$$

where $A = \mathrm{k}_x\Delta x$, $B = \mathrm{k}_y\Delta y$ and $\alpha = \frac{\kappa\Delta t}{\Delta x^2}$.
Once again, associating the above equation to the form, $x^2 - bx + c = 0$, we can identify the

neccessary and sufficient conditions for stability to be,

$$\begin{aligned} |c| &\leq 1, \\ -b &\leq c+1, \\ b &\leq c+1. \end{aligned} \tag{4.48}$$

Considering the first condition based on product of roots i.e. $\mid c \mid \leq 1$, we obtain,

$$\left| \frac{\alpha(1-w)}{9} \left\{ \sin^2(\mathrm{k}_x\Delta x)(\cos(\mathrm{k}_y\Delta y)+2)^2 \, + \, (\cos(\mathrm{k}_x\Delta x)+2)^2 \sin^2(\mathrm{k}_y\Delta y) \right\} \right| \leq 1 \tag{4.49}$$

$$\alpha(1-w) \text{ Max.} \left\{ \frac{\sin^2(\mathrm{k}_x\Delta x)(\cos(\mathrm{k}_y\Delta y)+2)^2 \, + \, (\cos(\mathrm{k}_x\Delta x)+2)^2 \sin^2(\mathrm{k}_y\Delta y)}{9} \right\} \leq 1 \tag{4.50}$$

This implies that

$$\alpha(1-w) \leq 0.928. \tag{4.51}$$

The condition based on sum of roots i.e. $-b \leq c+1$ yields

$$\begin{aligned} &-\left[1 - \frac{\alpha w}{9} \left\{ \sin^2(\mathrm{k}_x\Delta x)(\cos(\mathrm{k}_y\Delta y)+2)^2 \, + \, (\cos(\mathrm{k}_x\Delta x)+2)^2 \sin^2(\mathrm{k}_y\Delta y) \right\} \right] \\ &\leq 1 + \frac{\alpha(1-w)}{9} \left\{ \sin^2(\mathrm{k}_x\Delta x)(\cos(\mathrm{k}_y\Delta y)+2)^2 \, + \, (\cos(\mathrm{k}_x\Delta x)+2)^2 \sin^2(\mathrm{k}_y\Delta y) \right\}. \end{aligned} \tag{4.52}$$

This can be simplified further as

$$\frac{\alpha w}{9} \left\{ \sin^2(\mathrm{k}_x\Delta x)(\cos(\mathrm{k}_y\Delta y)+2)^2 \, + \, (\cos(\mathrm{k}_x\Delta x)+2)^2 \sin^2(\mathrm{k}_y\Delta y) \right\} \leq 2 \tag{4.53}$$

and therefore,

$$\alpha w \text{ Max.} \left\{ \frac{\sin^2(\mathrm{k}_x\Delta x)(\cos(\mathrm{k}_y\Delta y)+2)^2 \, + \, (\cos(\mathrm{k}_x\Delta x)+2)^2 \sin^2(\mathrm{k}_y\Delta y)}{9} \right\} \leq 2 \tag{4.54}$$

$$\text{i.e., } \alpha w \leq 1.856 \tag{4.55}$$

Combining Eq.(4.51) and Eq.(4.55), we obtain the conditions for stability as $w = \frac{2}{3}$, $1-w = \frac{1}{3}$ and $\alpha \leq \mathbf{2.784}$.

Note that the CFL limit is improved here by a factor of **11** as compared to 2D FTCS and by a factor of **3** in comparison with the pure delayed scheme. Also, we note that the weights obtained here are complementary to the 1D case i.e. $w_1 = \frac{2}{3}$ here as opposed to $w_1 = \frac{1}{3}$ in 1D.

Finally in 3D, the characteristic equation for the error amplification factor, λ, obtained through von-Neumann analysis is,

$$\lambda^2 - \lambda \left[1 - \frac{\alpha w}{9}\left\{\sin^2 A\,(1+\cos B+\cos C)^2 + \sin^2 B\,(1+\cos A+\cos C)^2 + \sin^2 C\,(1+\cos A+\cos B)^2\right\}\right] + \frac{\alpha(1-w)}{9}\left[\sin^2 A\,(1+\cos B+\cos C)^2 + \sin^2 B\,(1+\cos A+\cos C)^2 + \sin^2 C\,(1+\cos A+\cos B)^2\right] = 0,$$

where $\mathrm{k}_x\Delta x = A$, $\mathrm{k}_y\Delta y = B$, $\mathrm{k}_z\Delta z = C$ and $\alpha = \frac{\kappa\Delta t}{\Delta x^2}$.

Repeating the similar analysis as above we have the following conditions,

$$\mid c \mid \leq 1,$$

$$\left|\frac{\alpha(1-w)}{9}\left\{\sin^2 A\,(1+\cos B+\cos C)^2 + \sin^2 B\,(1+\cos A+\cos C)^2 + \sin^2 C\,(1+\cos A+\cos B)^2\right\}\right| \leq 1,$$

$$\frac{\alpha\,(1-w)}{9}\text{Max.}\left\{\sin^2 A(1+\cos B+\cos C)^2 + \sin^2 B(1+\cos A+\cos C)^2 + \sin^2 C(1+\cos A+\cos B)^2\right\} \leq 1,$$

$$\implies \alpha(1-w) \leq 0.9 \tag{4.56}$$

and,

$$-b \leq c+1,$$

$$-\left[1-\frac{\alpha w}{9}\left\{\sin^2 A\,(1+\cos B+\cos C)^2+\sin^2 B\,(1+\cos A+\cos C)^2 + \sin^2 C\,(1+\cos A+\cos B)^2\right\}\right] \le 1+\frac{\alpha(1-w)}{9}\left[\sin^2 A\,(1+\cos B+\cos C)^2 + \sin^2 B\,(1+\cos A+\cos C)^2+\sin^2 C\,(1+\cos A+\cos B)^2\right].$$

The above condition can be simplified further to yield,

$$\frac{\alpha w}{9}\left[\sin^2 A\,(1+\cos B+\cos C)^2+\sin^2 B\,(1+\cos A+\cos C)^2 + \sin^2 C\,(1+\cos A+\cos B)^2\right] \le 2$$

$$\frac{\alpha w}{9}\,\text{Max.}\left[\sin^2 A\,(1+\cos B+\cos C)^2+\sin^2 B\,(1+\cos A+\cos C)^2 + \sin^2 C\,(1+\cos A+\cos B)^2\right] \le 2$$

$$\implies \alpha w \le 1.8 \tag{4.57}$$

From Eq.(4.56) and Eq.(4.57), we obtain $w=\frac{2}{3}$, $1-w=\frac{1}{3}$ and $\alpha \le 2.7$.

Interestingly, the CFL limit is improved by a factor of **16** as compared to 3-D FTCS which required α to be ≤ 0.166 and a factor of **3** w.r.t. the delayed scheme.

Dimension	FTCS	Delayed $(\nabla\cdot\nabla)_{CD}$	Delayed $(\nabla\cdot\nabla)_{Iso}$	Weighted $(\nabla\cdot\nabla)_{Iso}$
1D	1/2	1.0	1.0	**1.5**
2D	1/4	1/2	0.928	**2.784**
3D	1/6	1/3	0.9	**2.7**

Table 4.2 CFL criteria for FTCS (Column II), Delayed $(\nabla\cdot\nabla)_{CD}$ (Column III), Delayed $(\nabla\cdot\nabla)_{Iso}$ (Column IV) and Weighted $(\nabla\cdot\nabla)_{Iso}$ (Column V).

A comparison of CFL criteria for different cases is given in Table 4.2 where it can be seen that similar to the case of 1D the weighted difference scheme provides significant improvement in CFL for 2D and 3D compared to FTCS scheme. Also, the use of isotropic operators with weighted difference scheme provides further improvement in CFL over pure delayed scheme.

Having obtained definite improvement in CFL from the weighted scheme along with isotropic operators, it is interesting to investigate whether this translates into an actual speed-up in CPU time for real test cases as the operation count both in terms of memory and arithmetic operations differ for each of the methods that have been considered. For instance, computation of central difference laplacian in 3D involves 7 memory operations (6 read and 1 write) whereas the computation of isotropic $\nabla \cdot \nabla$ laplacian involves 66 memory operations (60 read and 6 write). On the whole, CPU cache data reuse (to reduce cache misses), memory bandwidth, latency and communications (for parallel implementation) etc., are the factors that will ultimately determine if there is a substantial speed-up while using the pure delayed and weighted difference schemes. We now carry out a series of tests to understand the actual efficacy of the schemes as done in Chapter 3.

4.7 A three-dimensional transient test case

To begin with, we consider the following diffusion problem with periodic boundary conditions,

$$\frac{\partial \phi}{\partial t} = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2}, \tag{4.58}$$

with $(x, y, z) \in [0, 1]$ and the initial condition,

$$\phi(x, y, z, t = 0) = \sin(2\pi x)\ \sin(2\pi y)\ \sin(2\pi z).$$

The analytical solution for the above problem is,

$$\phi^*(x, y, z, t) = e^{-12\pi^2 t}\ \sin(2\pi x)\ \sin(2\pi y)\ \sin(2\pi z).$$

The L_2 norm of error normalized w.r.t. total number of grid points are defined as,

$$||\phi||_2 = \sqrt{\frac{\sum (\phi(x, y, z, t) - \phi^*(x, y, z, t))^2}{N_x * N_y * N_z}}, \tag{4.59}$$

where $\phi(x, y, z, t)$ is the computed solution and $\phi^*(x, y, z, t)$ is the analytical solution at any particular time instant. We record the CPU-time (in seconds) taken by different solvers to perform time integration starting from $t = 0$ to $t = 0.1 \times t_{\text{diffusion}}$ for different grid resolutions ranging from 48^3 to 256^3. CFL values used in the simulations for different solvers are given in Table 4.3.

Solver	CFL
FTCS	0.15
Delayed	0.90
Weighted	2.50

Table 4.3 CFL values used for simulations presented in Fig. 4.5

The numerical software for all the three solvers have been written in C++ language and were compiled using *gnu compilers* (gcc version 9.2.0) with -O3 optimization. The simulations were run on single core of AMD workstation equipped with AMD Ryzen Threadripper 2970WX processor and 128 GB RAM. We have optimized the solvers with different code optimization techniques namely SIMD, parallelization in time, loop and cache blocking technique, etc., to improve memory utilization by eliminating as many cache misses as possible. By using/reusing data in cache we try to reduce the memory fetch (reduce memory bandwidth pressure).

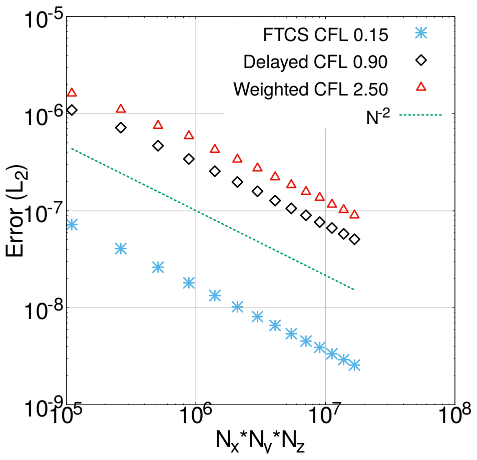
Grid Size	FTCS	Delayed $\nabla\cdot\nabla$	Weighted $\nabla\cdot\nabla$
48^3	0.227	0.252	0.217
64^3	0.773	1.122	0.745
80^3	2.991	3.179	2.292
96^3	8.216	8.323	5.489
112^3	25.538	17.233	10.799
128^3	48.189	34.024	22.817
144^3	59.332	53.314	37.143
160^3	147.377	101.467	61.186
176^3	169.744	154.619	109.657
192^3	236.568	218.153	150.914
208^3	518.407	352.083	221.044
224^3	573.256	497.492	376.754
240^3	706.880	705.456	477.748
256^3	1192.085	987.414	733.687

Table 4.4 Comparison of CPU-time with different solvers for grid resolution ranging from 48^3 and 256^3. 3-D transient diffusion test case with periodic boundaries.

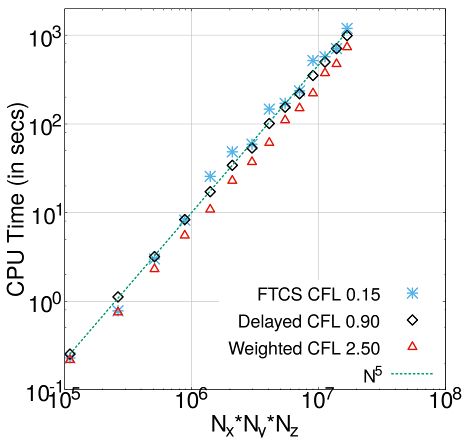
A comparison of CPU time for FTCS, Delayed and Weighted solvers has been presented in Table 4.4. It can be seen that for all the grid sizes, weighted scheme has better performance than FTCS and delayed schemes. However, the relative reduction in CPU time for weighted scheme as compared to FTCS was found to deteriorate slightly with the increase in grid size.

For instance, we observe 50% reduction in CPU time with weighted scheme as compared to FTCS for the grid size of 128^3 whereas for grid size of 256^3, the corresponding improvement is around 40%. We try to explain this behavior based on the experience gained with blocking optimization techniques. We found that best CPU time for all the solvers were obtained when we keep increasing the block size with the increase in grid size. At the same time, the chosen block size should be small enough to fit all the data into the cache. We observed that using a larger blocksize for larger grids yields better CPU time. However, it is important to note that with a large block size we have to fetch more data on a cache miss and the miss penalty also grows with increasing block size.

Next, shown in Fig. 4.5a are the L_2 norm of error as a function of the grid resolution varying from 48^3 to 256^3 for different methods. From the figure, we can verify that all the methods show second order convergence as expected (dotted line gives the ideal scaling for second-order schemes). It can also be seen that the error associated with delayed and weighted schemes are larger as compared to the FTCS scheme owing to the larger time-step (Δt) used for these methods. This is the minor cost one has to incur for the increased speed-up. CPU time for different methods has been plotted in Fig.4.5b for grid resolution varying from 48^3 to 256^3 and as expected computational time for all the methods scales as N^5 with the grid resolution, where N is the number of grid points in each direction.



(a) Grid size vs Error



(b) Grid size vs CPU time

Fig. 4.5 Plots for L_2 norm of error and CPU time (in seconds) for 3-D transient diffusion test case with periodic boundaries.

4.8 Parallel Scaling of 3D Transient Solvers

Proceeding further, we now focus our attention on parallel performance of these methods. Accordingly in this section, we illustrate the parallel scaling of FTCS, Delayed and Weighted solvers. Here, we use the same 3-D diffusion problem with periodic boundaries as in section 4.7. The domain is split up along processors in each direction yielding 3D block sub-domains. The simulations were carried out on an Intel Xeon Phi Knights Landing cluster. The hardware and software configuration used are given in Table 4.5. In order to time the simulations, we use calls to the MPI library's timing API routines.

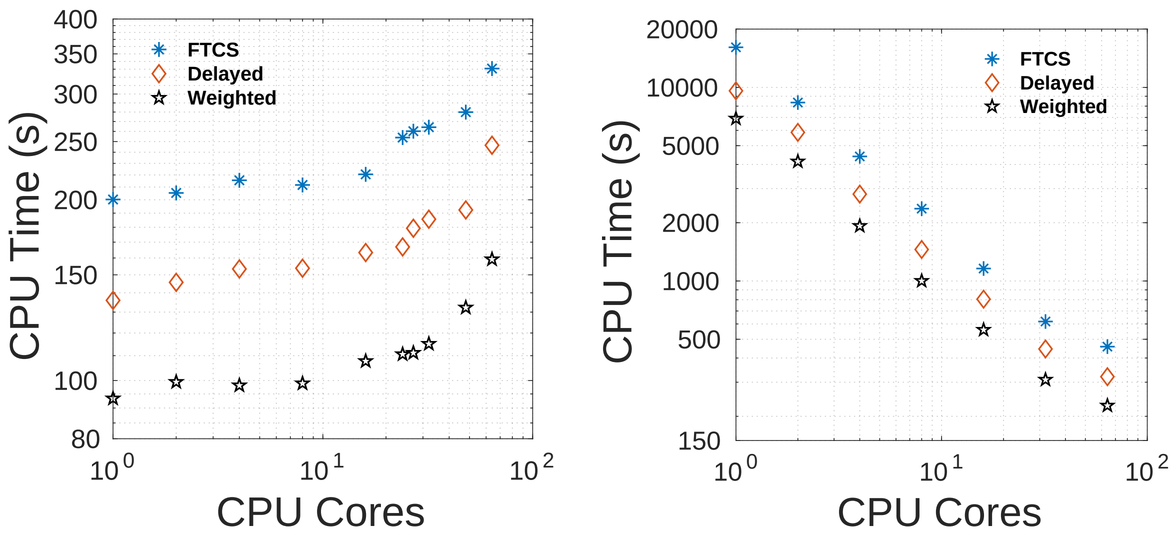
Processor Model	Intel Xeon E5-2620 v4
Clock Speed	2.1 GHz
CPU Cores/node	16
Memory Type	DDR4-2133MHz
Max no. of Memory Channels	4
Max Memory Bandwidth (DDR4)	68.3 GB/s
Network	InfiniBand
MPI Library	openmpi 1.1
Compiler	GCC

Table 4.5 System hardware and software configuration used for parallel scaling studies.

We have carried out the weak scaling studies using a 128^3 Cartesian grid assigned to each processor. In weak scaling, the workload assigned to each processor is kept constant and the problem size is increased in proportion with the number of processors. We perform time integration from $t = 0$ to $t = 0.1 \times t_{\text{diffusion}}$ on number of processors ranging from 1 to 64. Fig.4.6a shows the CPU time vs processor count for our weak scaling studies. As can be seen, all the solvers show approximate linear scaling upto 32 processors. Beyond 32 processors, we see that there is an increase in CPU time which occurs due to the MCDRAM bandwidth saturation. This can be essentially corroborated from the results obtained for STREAM triad benchmark. The MCDRAM bandwidth (in GB/s) measured with STREAM triad benchmark for number of processors ranging from 1 to 64 is presented in Table 4.6.

No. of processors	Bandwidth (GB/s)
1	12.026
2	23.374
4	46.603
8	92.564
16	181.130
24	256.293
32	306.900
48	263.107
56	300.347
64	331.470

Table 4.6 STREAM triad benchmark results for Intel Knights Landing.



(a) Weak Scaling with 128^3 grid points per MPI process

(b) Strong Scaling for a Cartesian grid with 512^3 grid points

Fig. 4.6 Weak Scaling (left) and Strong Scaling (right).

Further, we have carried out strong scaling studies using a 512^3 Cartesian grid. In the strong scaling scenario, the problem size remains fixed but the number of processors is increased progressively. A large enough problem size is used so that the communication cost does not dominate over the computational cost. Hence, we choose a grid size of 512^3 so that the problem size exceeds the sum of all the last level caches. We perform time integration from $t = 0$ to $t = 0.01 \times t_{\text{diffusion}}$ on number of processors ranging from 1 to 64. Fig.4.6b shows the CPU time vs processor count for our strong scaling studies. All the solvers show close to the inverse linear scaling with the processor count.

For both weak and strong scaling studies, CPU time behaves as expected with the weighted solver being atleast two times faster than the FTCS method.

4.9 Numerical experiments in 3D for Poisson's equation

Motivated by the recent developments on the fast iterative algorithms [11, 43, 10], we implement a pseudo-transient approach to solve the Poisson equations. Mathematically, by adding a pseudo-time derivative, we obtain the diffusion equation,

$$\frac{\partial \phi}{\partial t}(x,t) = \nabla \cdot (\kappa \nabla \phi (x,t)) + S(x). \tag{4.60}$$

The solution to the original boundary value problem can be recovered by performing a long time integration of Eq.(4.60) till the steady state when the time derivative will vanish. Here, the boundary conditions are kept fixed and hence, we expect the solution to "converge" to the steady state solution (i.e., $\frac{\partial \phi}{\partial t} = 0$). In this section, we consider a set of problems to assess the usefulness of our current weighted approach in solving Poisson equations in 3D. For the purpose of validation, we consider test cases with known analytical solutions for different boundary conditions. We define the error based on Euclidean norm normalized w.r.t. the total number of grid points as below,

$$||\phi||_2 = \sqrt{\frac{\sum (\phi(x,y,z,t) - \phi^*(x,y,z))^2}{N_x * N_y * N_z}} \tag{4.61}$$

where $\phi(x,y,z,t)$ is the computed solution at any particular time instant and $\phi^*(x,y,z)$ is the analytical solution at the final steady state. In our studies, the simulations are assumed to reach a converged solution when the error norm of ϕ between two successive iterations is less than 10^{-12}. All the codes have been written in C++ language and were compiled using *gnu compilers* (gcc version 9.2.0) with -O3 optimization. All the test cases presented in the following sub-sections were run on single core of AMD workstation equipped with AMD Ryzen Threadripper 2970WX processor and 128 GB RAM.

4.9.1 3-D Poisson equation with periodic boundaries

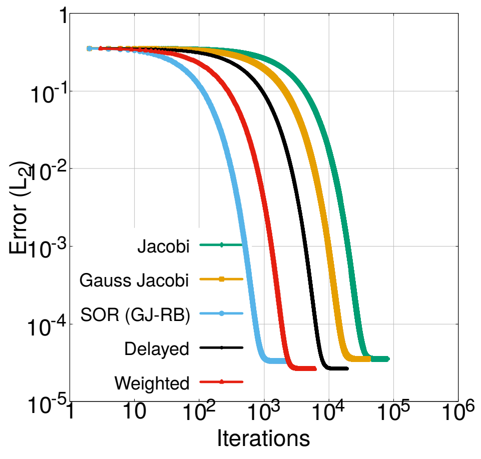
We first consider the following problem,

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} = -12\pi^2 \sin(2\pi x)\sin(2\pi y)\sin(2\pi z) \tag{4.62}$$

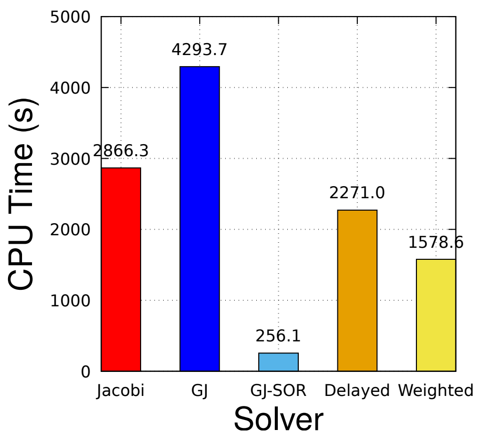
on a unit cube, $0 \leq x \leq 1$, $0 \leq y \leq 1$, $0 \leq z \leq 1$ with periodic boundaries. The domain is initialized to $\phi(x,y,z) = 0$. The analytical solution for the above problem is given by, $\phi_{analytical} = \sin(2\pi x)\sin(2\pi y)\sin(2\pi z)$. A mesh consisting of $256 \times 256 \times 256$ grid points has been used to perform the simulations. We employ the delayed $\nabla \cdot \nabla$ and weighted $\nabla \cdot \nabla$ schemes to solve Eq.(4.62) in a pseudo-transient framework. We compare the performance of these methods with the conventional Poisson solvers such as Jacobi, Gauss-Jacobi or red-black Gauss-Seidel, and Gauss-Jacobi with over-relaxation (SOR). The successive over-relaxation method is implemented here by taking an optimum value of over-relaxation factor given as $\omega_{opt} = 2/(1+\sin(\pi\Delta x))$ where Δx is the grid spacing [3].

The left hand panel in Fig.4.7 shows the L_2 norm of error vs number of iterations for different algorithms. As can be seen from the plot, the SOR method converges the fastest followed by the weighted scheme in terms of number of iterations to attain the steady state. However, it should be noted that in this case SOR technique is implemented using the optimum relaxation factor ω_{opt} and it is not always possible to obtain an optimum value of ω expect for few selected problems (such as those involving periodic and dirichlet boundaries) and there is no gurantee that simulations performed with ω_{opt} will always converge.

The right hand panel in Fig.4.7 shows a comparison of single core CPU-time for different methods. It is evident that the serial version of weighted scheme is computationally efficient than Jacobi, Gauss-Jacobi and delayed schemes. The serial version of SOR method is faster than the weighted scheme. However, it is well known that the SOR method isn't parallelizable whereas the current weighted solver is inherently parallelizable.

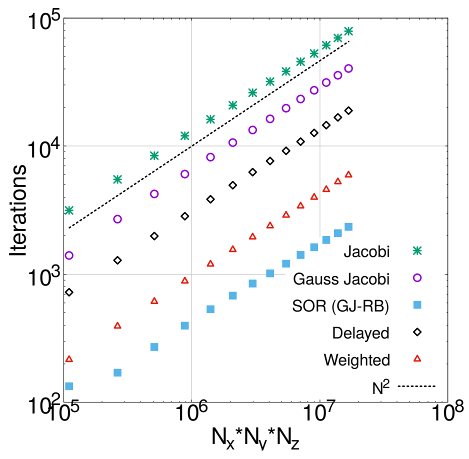


(a) L_2 norm of error vs number of iterations

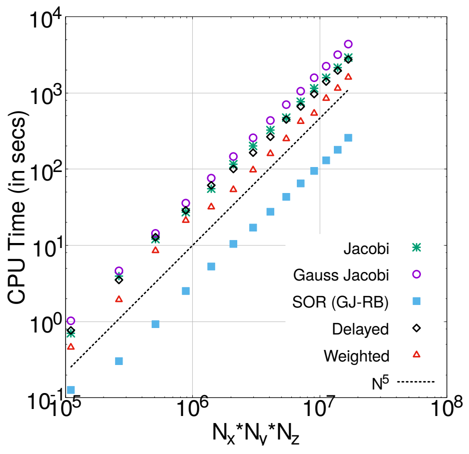


(b) CPU Time for different solvers.

Fig. 4.7 L_2 norm (left) and CPU time (right) comparison for periodic boundary conditions.



(a) Grid resolution vs number of iterations



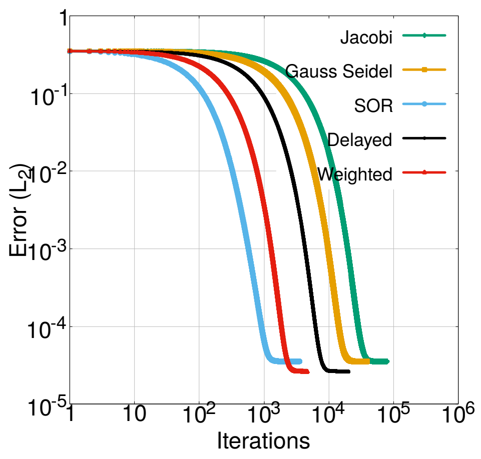
(b) Grid resolution vs CPU Time

Fig. 4.8 Scaling studies for number of iterations and CPU time with increasing grid resolution.

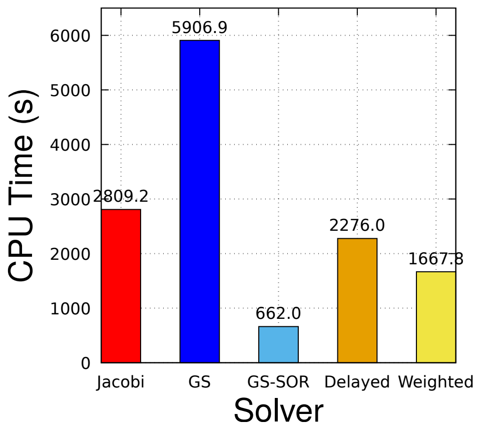
We have performed a grid resolution study to analyze the performance of different algorithms in terms of number of iterations to attain a given level of convergence and computational cost in terms of CPU time. Fig.4.8 shows the results obtained with grid resolution studies by varying the grid resolution from 48^3 to 256^3. The left hand panel shows that number of iterations required to obtain a given level of convergence is $O(N^2)$ as expected for a uniform $N \times N \times N$ grid where N is the number of grid points in each direction. Consequently, the CPU time scales as $O(N^2 \times N^3) \sim O(N^5)$ as can be seen from the right hand panel in Fig.4.8.

4.9.2 3-D Poisson equation with Dirichlet boundaries

In order to verify the universality of the above results for other boundary conditions, we once again consider Eq.4.62 with dirichlet boundary conditions specified on the domain boundaries: $\phi(0,y,z) = 0$, $\phi(1,y,z) = 0$, $\phi(x,0,z) = 0$, $\phi(x,1,z) = 0$, $\phi(x,y,0) = 0$, $\phi(x,y,1) = 0$. The interior grid points are initialized to $\phi(x,y,z) = 0$. Analytical solution for this case is $\phi_{analytical} = \sin(2\pi x)\sin(2\pi y)\sin(2\pi z)$. Numerical studies are carried out on a $256 \times 256 \times 256$ mesh. L_2 norm of error with various methods viz. Jacobi, Gauss-Seidel(GS), Gauss-Seidel with over-relaxation (SOR, $\omega = 1.9$) , delayed $\nabla \cdot \nabla$ and weighted $\nabla \cdot \nabla$ scheme have been included in Fig.4.9a. Once again, it can be seen that the weighted scheme is capable of producing results with comparable accuracy in lesser number of iterations than Jacobi, Gauss-Seidel and Delayed methods. From the right hand panel of Fig. 4.9b, it is also evident that the weighted scheme is computationally more efficient, in terms of serial CPU time, than Jacobi, Gauss-Seidel and delayed methods. Here again, Gauss-Seidel with successive over-relaxation performs better than weighted scheme in this case for serial implementation both in terms of number of iterations to attain steady state and single-core CPU time. We recall again that the SOR method is not parallelizable whereas the current weighted solver is inherently parallelizable.



(a) L$_2$ norm of error vs number of iterations



(b) CPU Time for different solvers.

Fig. 4.9 L_2 norm (left) and CPU time (right) comparison for Dirichlet boundary conditions.

4.9.3 3-D Poisson equation with mixed boundary conditions

Finally, we verify the consistency of the method's behaviour for 3D Poisson problem which has dirichlet boundary conditions in one direction and neumann conditions on the remaining boundaries i.e.,

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}=f(x,y,z) \tag{4.63}$$

where, (x,y,z) $\in \Omega = [0,1]\times[0,1]\times[0,1]$ subject to the boundary conditions,

$$\begin{aligned} \phi(0,y,z)&=0 & \phi(1,y,z)&=0\\ \frac{\partial\phi}{\partial y}(x,0,z)&=0 & \frac{\partial\phi}{\partial y}(x,1,z)&=0\\ \frac{\partial\phi}{\partial z}(x,y,0)&=0 & \frac{\partial\phi}{\partial z}(x,y,1)&=0. \end{aligned}$$

Method of manufactured solutions

We can see that the above mentioned boundary conditions are satisfied if we choose the steady state value of $\phi(x,y,z)$ to be,

$$\phi' = x(1-x)\left[\cos(\pi x)+\cos(\pi y)+\cos(\pi z)\right]/\pi$$

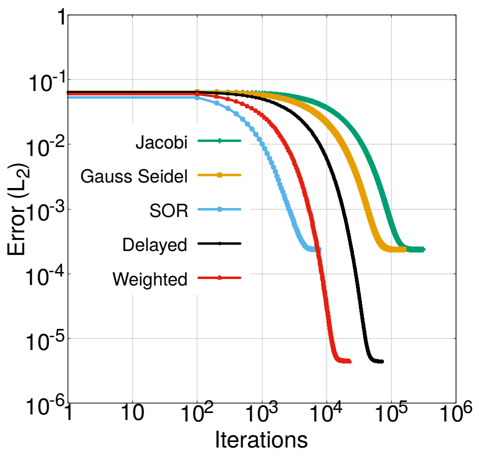
With the laplacian applied on $\phi'(x,y,z)$, we obtain the right hand side

$$\nabla^2\phi'(x,y,z) = -\left\{\frac{2}{\pi}+\pi x(1-x)\right\}\{\cos(\pi x)+\cos(\pi y)+\cos(\pi z)\} - 2(1-2x)\sin(\pi x).$$

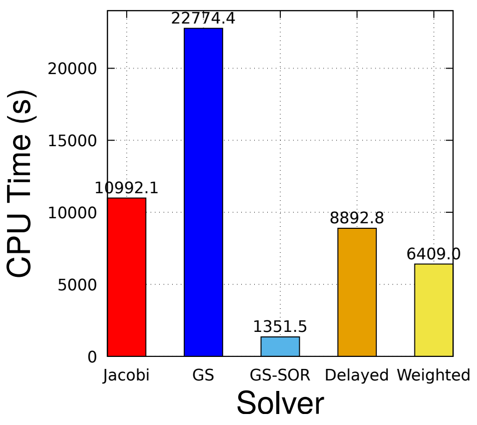
Now, with $f'(x,y,z) = \nabla^2\phi'(x,y,z)$ and solving,

$$\frac{\partial\phi}{\partial t}+\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2} = f'(x,y,z) \tag{4.64}$$

we obtain the numerical solution $\phi(x,y,z)$. A mesh consisting of $256\times256\times256$ grid points has been used to perform the simulations. Error norms are computed using the computed solution ϕ and the analytical solution $\phi_{analytical} = \phi'$ which are included in Fig.4.10a. The value of over-relaxation parameter ω used for SOR method is 1.9. As can be seen, delayed and weighted schemes provide a significant improvement in convergence error for mixed boundary conditions as compared to Jacobi, Gauss-Seidel and SOR methods. Also, it can be seen from Fig.4.10b that the weighted scheme significantly outperforms other methods except SOR when measured in terms of CPU time (single-core).



(a) L$_2$ norm of error vs number of iterations.



(b) CPU Time for different solvers.

Fig. 4.10 L_2 norm (left) and CPU time (right) comparison for mixed boundary conditions.

An interesting observation that can be made by looking at the CPU time values in Fig.4.7b, 4.9b and 4.10b for the above three test cases is that CPU time for Gauss-Seidel is more than Jacobi method although the former requires almost half the number of iterations to attain the same level of convergence error. Conventionally, for serial applications, Gauss-Seidel method are known to be faster than Jacobi method for solving Poisson equations or elliptic partial differential equations in general. This belief was derived when computers were designed very differently than today. On modern computers, vectorization allows in-core parallelism for operations on arrays (vectors) of data. Auto-vectorization (enabled with -O3 optimization on *gnu compilers*) transforms sequential code to exploit the SIMD instructions (singe instruction multiple data) within the processor to speed up execution times. However, whether an algorithm can be vectorized or not depends on many factors including primarily the data dependence. A statement X is said to be data dependent on statement Y if,

- Y executes before X in the original sequential program,
- X and Y access the same data item,
- atleast one of the accesses is a write.

Specific to our case of stencil computation, there can be data dependencies arising when a variable is written in one iteration and read in a subsequent iteration i.e. read after write. For example, let us consider the following,

```
A[0] = 0;
for(int i=1; i<MAX; i++)
        A[i] = A[i-1] + 1;
```

This is equivalent to,

```
A[1] = A[0] + 1;
A[2] = A[1] + 1;
A[3] = A[2] + 1;
A[4] = A[3] + 1;
```

The above loop cannot be vectorized safely because if the first two iterations are executed simultaneously by a SIMD instruction, the value of A[1] may be used by the second iteration before it has been calculated by the first iteration which could lead to incorrect results. This is indeed the case for Gauss-Seidel iteration where we use the latest values for the neighbors which have been already updated and thus owing to the read after write data dependencies Gauss-Seidel algorithm cannot be vectorized. On the other hand, Jacobi iterations doesn't have any such dependency and so it can be vectorized by the compilers at ease. Thus, it is essentially interesting to look at how modern computer architecture has changed the way in which the conventional algorithms have been looked at, primarily from the point of data dependence.

4.10 Closure

We have presented a numerical scheme for solving diffusion equation in multi-dimensions. The weighted schemes are capable of computing explicit updates to diffusion problems with considerably higher computational efficiency than explicit forward-Euler methods of FTCS schemes. The results of the chapter are summarized as follows:

- The conventional FTCS methods are not computationally efficient for multi-dimensional diffusion problems due to prohibitively smaller time-steps. The CFL limit for these schemes varies inversely with dimensions.
- We have made use of isotropic laplacian operator for which truncation error is isotropic at leading order. CFL limit for explicit schemes with isotropic $\nabla \cdot \nabla$ laplacian operator was found to be $\sim$ 1 i.e., 2.784 in 2D and 2.7 in 3D, and therefore, the curse of dimensionality is alleviated with the current approach. The corresponding improvement is by a factor 11 and 16 in 2D and 3D.
- We performed a spectral analysis of the laplacian operators to explain the difference between the two discretization techniques. The lower magnitude of spectral radius or maximum eigen value of the isotropic laplacian operator in comparison to the central difference operators results in the improvement in maximum allowable time-step.
- Results obtained from grid resolution studies show that weighted schemes consistenly perform at design accuracy and expected computational complexity.
- The weighted schemes show a good scaling on modern distributed architecture (Intel KNL in present study). The wallclock times recorded for weighted method is less than FTCS by about 50%. This is true for both weak and strong scaling studies.
- Results obtained from solution of Poisson equations for different boundary conditions show that the weighted methods inherit the simplicity and parallel efficiency of the Jacobi method while retaining the fast convergence property of Gauss-Seidel method.

Chapter 5

Conclusions

A number of scientific and engineering systems are governed by PDEs which have solutions comprising of wide range of spatial and temporal scales. The accompanying details can only be captured by high-fidelity simulations on high performance computational systems. Although, advances in computing technology have made it possible to carry out intensive simulations on massively parallel computers, the presence of order of magnitude gaps in spatial and temporal scales is a major obstacle. Thus, the development of PDE solvers which can bridge these order of magnitude gaps in spatial and temporal scales is a major challenge for computational physics and a central topic for high-performance computing. In this thesis, we have presented a general methodology to analyze and derive explicit schemes that can overcome the issue of small time-steps by allowing some tunable level of asynchrony i.e. using spatial data from present and previous time levels for computing derivatives.

The concept relies on finite differences to approximate derivatives using values of the function from neighboring points and the realization that wider stencils have better stability. The performance of explicit solvers is stalled due to the time-step restrictions imposed by the stability criteria which is more severe for a problem with higher number of degrees of freedom. In the earlier works by Dheevatsa *et al* [14] and Mahan *et al* [19], it was shown that the stability limit for an explicit solver of the diffusion equation was doubled in the case of delayed scheme where the computations used values from past time levels. In this work, we have highlighted a form of checkerboard instability in the formulation of delayed difference scheme for 1D diffusion systems. In order to overcome this issue and attain a better scheme, we generalize the concept further by considering a scheme that is weighted between the delayed and the conventional explicit schemes. We have established via stability analysis that an optimum value of the weighting factor exists such that the critical CFL number obtained is larger than the original delayed difference scheme. These schemes are referred as weighted difference schemes.

The multidimensional extension of weighted differencing approach for diffusion equation incorporated isotropic differential operators to compute the spatial derivatives. By analyzing in detail the effective differential equation, we demonstrated the consistency and order of accuracy of weighted scheme. We have provided a general framework in which stability of schemes can be accessed by analyzing the eigen value spectra of the diffusion operator for different discretization procedure namely central difference and isotropic operators. Fourier analysis of the diffusion operator revealed that isotropic discretization, $\nabla \cdot \nabla$, alleviates the dependency of CFL on dimension of problem as opposed to the conventional central difference discretization. With the weighted difference approach presented in this work, we have obtained CFL improvement by factors exceeding 11 and 16 for two-dimensional and three-dimensional diffusion equation, respectively, over the conventional FTCS scheme. Moreover, the weighted difference scheme also provided further CFL improvement over the multi-dimensional delayed scheme of Mahan *et al* [19]. To conclude our discussion regarding the improvement in CFL for different cases a summary of CFL values for different schemes is presented in Table 5.1 below:

Dimension	FTCS	Delayed $(\nabla \cdot \nabla)_{CD}$	Delayed $(\nabla \cdot \nabla)_{Iso}$	Weighted $(\nabla \cdot \nabla)_{Iso}$
1D	1/2	1.0	1.0	**1.5**
2D	1/4	1/2	0.928	**2.784**
3D	1/6	1/3	0.9	**2.7**

Table 5.1 CFL improvement obtained with weighted difference scheme for diffusion equation.

Theoretical predictions on the accuracy and convergence rates of weighted schemes were corroborated with numerical experiments for transient and steady problems. Good agreement was found across different initial and boundary conditions. For serial applications involving three-dimensional diffusion systems, we have obtained upto about 50% reduction in CPU execution time with weighted scheme over FTCS scheme. With the parallel implementation of the 3D code, our weak and strong scaling results on a distributed architecture (Intel KNL) were close to ideal linear scaling with weighted solver being at least two times faster than FTCS solver. We have also demonstrated the efficacy of weighted scheme through a pseudo-transient framework to obtain solution of Poisson's equation for different boundary conditions. Again, the performance of weighted scheme was found to be better than Jacobi and Gauss-Seidel methods in terms of CPU execution time and number of iterations to attain steady state. We conclude our discussion with the following remarks:

- Long time integration can be efficiently performed using the **weighted** $\nabla \cdot \nabla$ solver for multi-dimensional diffusion systems as it enhances the stability without a significant deterioration in accuracy.

- Weighted algorithm is inherently **parallel** and well suited for domain decomposition while providing significant improvement in computation time over standard FTCS scheme. We found for our specific case, weighted scheme gave about 100% speed-up over FTCS scheme.
- Computational efficiency of **weighted** $\nabla \cdot \nabla$ solver is better than Jacobi and Gauss Seidel methods.
- **Weighted** $\nabla \cdot \nabla$ solver can be used as a pre-conditioner to aid the performance of Conjugate gradient, Multigrid methods for solving elliptic partial differential equations (e.g. Poisson equation).

Hence, the work presented here provides a strong foundation for deriving schemes which inherits the simplicity and parallel efficiency of the Jacobi method while retaining the fast convergence property of the Gauss-Seidel method.

5.1 Suggestions for Future Work

Finally, we mention some important extensions to the present work. The studies presented here were based on uniform grid spacing, mainly for the purpose of clarity in exposition. However, in many problems non-uniform meshes are required. This can be straightforwardly incorporated, for example, by the means of an analytical mapping between a non-uniform mesh and a uniform computational mesh. In this case, derivatives are affected by the Jacobian of the transformation matrix. Alternatively, the second approach is to perform the computations directly on the non-uniform grid. This can be more tedious as the time-step will be limited by the smallest grid spacing in the domain. The order of scheme is expected to stay unaffected for sufficiently slowly varying grid spacings.

Another interesting direction would be to formulate hybrid schemes, which combine the advantages of the multiple methodologies. One such possibility is to extend the stability limit of weighted difference schemes further by implementing these schemes in a super-time stepping framework where stability is enforced at the end of N time-steps. Research into the application of STS-type weighted difference schemes for diffusion and advection equations is of particular interest, as it could yield a scalable alternative to existing semi-implicit operators.

It is also possible to extend our weighted difference approach to other discretization methods such as finite volume approach where fluxes across control volume surface can be evaluated using data from earlier time step. The stability of weighted difference schemes remains the same for finite volume implementation.

References

[1] Sumesh P. Thampi, Santosh Ansumali, R. Adhikari, and Sauro Succi. Isotropic discrete laplacian operators from lattice hydrodynamics. *Journal of Computational Physics*, 234:1 – 7, 2013.

[2] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[3] Charles Hirsch. *Numerical Computation of Internal and External Flows, The Fundamentals of Computational Fluid Dynamics*. Elsevier, 2/e. edition, 2007.

[4] Zoran Mikić, Jon A. Linker, Dalton D. Schnack, Roberto Lionello, and Alfonso Tarditi. Magnetohydrodynamic modeling of the global solar corona. *Physics of Plasmas*, 6(5):2217–2224, 1999.

[5] Bhargav Vaidya, Deovrat Prasad, Andrea Mignone, Prateek Sharma, and Luca Rickler. Scalable explicit implementation of anisotropic diffusion with runge-kutta-legendre super-time-stepping. *Monthly Notices of the Royal Astronomical Society*, 472, 02 2017.

[6] Sergey Astanin and Luigi Preziosi. *Multiphase Models of Tumour Growth*, pages 1–31. Birkhäuser Boston, Boston, 2008.

[7] Bruce Finlayson and L.E. Scriven. The method of weighted residuals - a review. *Appl. Mech. Rev.*, 19:735–748, 01 1966.

[8] J.H.Ferziger and M.Peric. *Computational Methods for Fluid Dynamics*. Springer, 3/e. edition, 2002.

[9] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A MultiGrid Tutorial*. SIAM, 2/e. edition, 2000.

[10] Jun Zhang. Fast and high accuracy multigrid solution of the three dimensional poisson equation. *Journal of Computational Physics*, 143(2):449 – 461, 1998.

[11] Xiyang I.A. Yang and Rajat Mittal. Acceleration of the jacobi iterative method by factors exceeding 100 using scheduled relaxation. *Journal of Computational Physics*, 274:695 – 708, 2014.

[12] Vasilios Alexiades, Geneviève Amiez, and Pierre-Alain Gremaud. Super-time-stepping acceleration of explicit schemes for parabolic problems. *Communications in Numerical Methods in Engineering*, 12(1):31–42, 1996.

[13] V. Alexiades. Overcoming the stability restriction of explicit schemes via super-time-stepping. *Dynamic Systems and Applications*, 2:39–44, 04 1996.

[14] Mudigere Dheevatsa, Sherlekar Sunil D., and Ansumali Santosh. Delayed difference scheme for large scale scientific simulations. *Phys. Rev. Lett.*, 113:218701, Nov 2014.

[15] Andreas Frommer and Daniel B. Szyld. On asynchronous iterations. *Journal of Computational and Applied Mathematics*, 123(1):201 – 216, 2000. Numerical Analysis 2000. Vol. III: Linear Algebra.

[16] Konduri Aditya and Diego A. Donzis. High-order asynchrony-tolerant finite difference schemes for partial differential equations. *Journal of Computational Physics*, 350:550 – 572, 2017.

[17] Ankita Mittal and Sharath Girimaji. Proxy-equation paradigm: A strategy for massively parallel asynchronous computations. *Phys. Rev. E*, 96:033304, Sep 2017.

[18] Dganit Amitai, Amir Averbuch, Moshe Israeli, and Samuel Itzikowitz. On parallel asynchronous high-order solutions of parabolic pdes. *Numerical Algorithms*, 12(1):159–192, Mar 1996.

[19] Mahan Raj Banerjee, Sauro Succi, Santosh Ansumali, and R Adhikari. Isotropic finite-difference discretization of stochastic conservation laws preserving detailed balance. *Journal of Statistical Mechanics: Theory and Experiment*, 2017(10):103202, oct 2017.

[20] W. Gentzsch. *Numerical Solution of Linear and Non-Linear Parabolic Differential Equations by a Time-Discretisation of Third Order Accuracy*, pages 109–117. Vieweg+Teubner Verlag, Wiesbaden, 1980.

[21] Ratnesh K. Shukla and Pritam Giri. Isotropic finite volume discretization. *Journal of Computational Physics*, 276:252 – 290, 2014.

[22] R. M. Caplan, Z. Mikić, J. A. Linker, and R. Lionello. Advancing parabolic operators in thermodynamic MHD models: Explicit super time-stepping versus implicit schemes with krylov solvers. *Journal of Physics: Conference Series*, 837:012016, may 2017.

[23] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2/e. edition, 2003.

[24] Chad D. Meyer, Dinshaw S. Balsara, and Tariq D. Aslam. A stabilized runge–kutta–legendre method for explicit super-time-stepping of parabolic and mixed equations. *Journal of Computational Physics*, 257:594 – 626, 2014.

[25] J. Weickert. Theoretical foundations of anisotropic diffusion in image processing. In W. Kropatsch, R. Klette, F. Solina, and R. Albrecht, editors, *Theoretical Foundations of Computer Vision*, pages 221–236, Vienna, 1996. Springer Vienna.

[26] D. Barnaś and L.K. Bieniasz. Utility of super-time-stepping for electroanalytical digital simulations by explicit finite difference methods. part 1: Spatially one-dimensional models. *Journal of Electroanalytical Chemistry*, 815:210 – 219, 2018.

[27] Ivo F. Sbalzarini, Arnold Hayer, Ari Helenius, and Petros Koumoutsakos. Simulations of (an)isotropic diffusion on curved biological surfaces. *Biophysical Journal*, 90(3):878 – 885, 2006.

[28] Minchen Li, Ming Gao, Timothy Langlois, Chenfanfu Jiang, and Danny M. Kaufman. Decomposed optimization time integrator for large-step elastodynamics. *ACM Trans. Graph.*, 38(4), July 2019.

[29] D. Chazan and W. Miranker. Chaotic relaxation. *Linear Algebra and its Applications*, 2(2):199 – 222, 1969.

[30] Dganit Amitai, Amir Averbuch, Moshe Israeli, and Samuel Itzikowitz. Implicit-explicit parallel asynchronous solver of parabolic pdes. *SIAM J. Sci. Comput.*, 19(4):1366–1404, July 1998.

[31] Wen Shangmeng and Li Xiaomei. A class of stable difference schemes for linear elliptic pdes and their asynchronous parallel computation. *Wuhan University Journal of Natural Sciences*, 1(3):553–556, Dec 1996.

[32] S. Succi, G. Amati, M. Bernaschi, G. Falcucci, M. Lauricella, and A. Montessori. Towards exascale lattice boltzmann computing. *Computers and Fluids*, 181(Phys Rev E 92 4 2015):107–115, 2019.

[33] Culbert B. Laney. *Computational Gasdynamics*. Cambridge University Press, 1998.

[34] Rashmi Ramadugu, Sumesh P. Thampi, Ronojoy Adhikari, Sauro Succi, and Santosh Ansumali. Lattice differential operators for computational physics. *EPL (Europhysics Letters)*, 101(5):50006, mar 2013.

[35] Anand Kumar. Isotropic finite-differences. *Journal of Computational Physics*, 201:109–118, 2004.

[36] Michael Patra and Mikko Karttunen. Stencils with isotropic discretization error for differential operators. *Numerical Methods for Partial Differential Equations*, 22(4):936–953, 2006.

[37] Sanjiva K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16 – 42, 1992.

[38] Christopher K.W. Tam and Jay C. Webb. Dispersion-relation-preserving finite difference schemes for computational acoustics. *Journal of Computational Physics*, 107(2):262 – 281, 1993.

[39] Pavel B. Bochev and James Mac Hyman. Principles of mimetic discretizations of differential operators. 2006.

[40] James M. Hyman and Mikhail Shashkov. Mimetic discretizations for maxwell's equations. *Journal of Computational Physics*, 151(2):881 – 909, 1999.

[41] Y. Oono and S. Puri. Study of phase-separation dynamics by use of cell dynamical systems. i. modeling. *Phys. Rev. A*, 38:434–453, Jul 1988.

[42] Y. Oono and S. Puri. Computationally efficient modeling of ordering of quenched phases. *Phys. Rev. Lett.*, 58:836–839, Feb 1987.

[43] Dhiraj V. Patil, Kannan N. Premnath, and Sanjoy Banerjee. Multigrid lattice boltzmann method for accelerated solution of elliptic equations. *Journal of Computational Physics*, 265:172 – 194, 2014.

Appendix A

Quadratic Equation with Roots within the Unit Circle

In this appendix, we look at the necessary and sufficient conditions for the roots of a quadratic equation to have magnitude less than unity. For this purpose, we shall consider a quadratic equation of the generic form,

$$x^2 - bx + c = 0. \tag{A.1}$$

Its roots are given as

$$x_1 = \frac{b - \sqrt{b^2 - 4c}}{2} \;\; \& \;\; x_2 = \frac{b + \sqrt{b^2 - 4c}}{2}. \tag{A.2}$$

If the above roots are imaginary, then the necessary and sufficient conditions follow as

$$|x_1| = |x_2| \leq 1 \implies |c| \leq 1 \tag{A.3}$$

However if the roots are real, then the obvious conditions,

$$|c| \leq 1 \;\; \& \;\; -2 \leq b \leq 2, \tag{A.4}$$

are not sufficient to ensure the boundedness of the roots, i.e. $|x_1| \leq 1$ and $|x_2| \leq 1$. It thus becomes pertinent to directly enforce these conditions by means of the relations,

$$(i) - 2 \leq b - \sqrt{b^2 - 4c} \leq 2 \;\; \& \;\; (ii) - 2 \leq b + \sqrt{b^2 - 4c} \leq 2. \tag{A.5}$$

They can be simplified as

$$(i) - M \leq \sqrt{b^2 - 4c} \leq N \;\; \& \;\; (ii) - N \leq \sqrt{b^2 - 4c} \leq M, \tag{A.6}$$

where $M = 2 - b$ and $N = 2 + b$. Note that if b is positive, then $M < N$, and vice-versa. Correspondingly, the range for each of the conditions changes with the sign of b, as shown in the Fig. A.1 below.

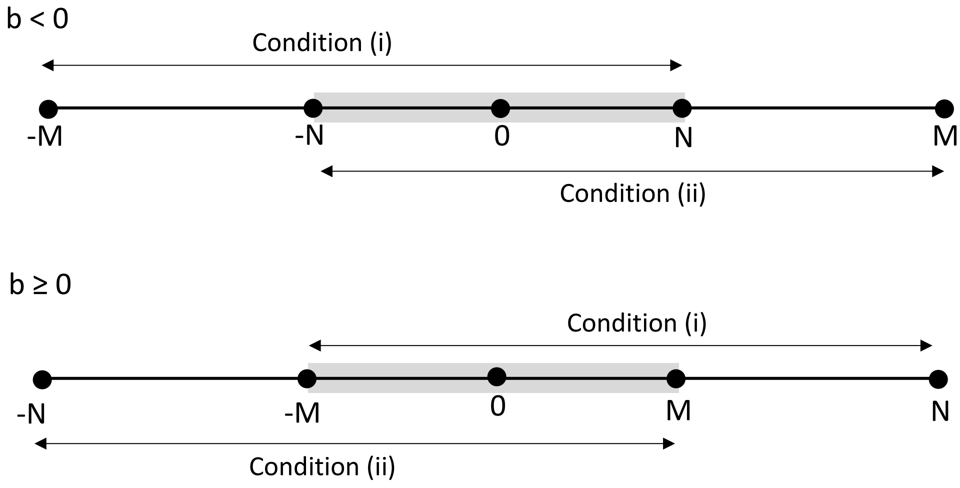


Fig. A.1 Range of $\sqrt{b^2 - 4c}$ for different b

In both the above cases, the desired value of $\sqrt{b^2 - 4c}$ should lie within the overlapping (shaded) region of the two conditions. See that we can accordingly tweak the original conditions as

$$(i) - M \leq \sqrt{b^2 - 4c} \leq M \;\; \& \;\; (ii) - N \leq \sqrt{b^2 - 4c} \leq N \tag{A.7}$$

Based on b, one of the above conditions needs to be primarily satisfied while other condition is implicitly adhered. Simplifying these conditions further, we obtain,

$$\begin{aligned} b^2 - 4c \leq M^2 &\implies b \leq c + 1, \\ b^2 - 4c \leq N^2 &\implies -b \leq c + 1. \end{aligned} \tag{A.8}$$

Thus, we have the necessary and sufficient conditions given as

$$\begin{aligned} |c| &\leq 1, \\ -b &\leq c+1, \\ b &\leq c+1. \end{aligned} \tag{A.9}$$

Appendix B

The Effective PDE

In this section, we derive the effective PDE that is being solved by the proposed weighted difference scheme using the Cauchy-Kowalewski procedure. For this, we first write the general Taylor series expansion of variable ϕ in both time and space as

$$\begin{aligned}\phi_{i+l}^{n-m} &= \phi_i^n + \left(l\Delta x\frac{\partial \phi}{\partial x} - m\Delta t\frac{\partial \phi}{\partial t}\right) + \frac{1}{2}\left(l^2\Delta x^2\frac{\partial^2 \phi}{\partial x^2} + m^2\Delta t^2\frac{\partial^2 \phi}{\partial t^2} - 2ml\Delta x\Delta t\frac{\partial^2 \phi}{\partial t\partial x}\right)\\ &+ \frac{1}{6}\left(l^3\Delta x^3\frac{\partial^3 \phi}{\partial x^3} - m^3\Delta t^3\frac{\partial^3 \phi}{\partial t^3} - 3ml^2\Delta x^2\Delta t\frac{\partial^3 \phi}{\partial t\partial x^2} + 3m^2 l\Delta x\Delta t^2\frac{\partial^3 \phi}{\partial t^2\partial x}\right)\\ &+ \frac{1}{24}\left(l^4\Delta x^4\frac{\partial^4 \phi}{\partial x^4} + m^4\Delta t^4\frac{\partial^4 \phi}{\partial t^4} + 6m^2l^2\Delta x^2\Delta t^2\frac{\partial^4 \phi}{\partial x^2\partial t^2} - 4ml^3\Delta x^3\Delta t\frac{\partial^4 \phi}{\partial t\partial x^3} - 4m^3l\Delta x\Delta t^3\frac{\partial^4 \phi}{\partial t^3\partial x}\right)\\ &+ Higher\,order\,terms\end{aligned} \tag{B.1}$$

Using the above expression, one can write the general central difference formula as

$$\frac{\phi_{i+l}^{n-m} + \phi_{i-l}^{n-m} - 2\phi_i^{n-m}}{l^2\Delta x^2} = \frac{\partial^2 \phi}{\partial x^2} - m\Delta t\frac{\partial^3}{\partial t\partial x^2}\phi + \frac{1}{12}\left(l^2\Delta x^2\frac{\partial^4}{\partial x^4} + 6m^2\Delta t^2\frac{\partial^4}{\partial x^2\partial t^2}\right)\phi. \tag{B.2}$$

For the present weighted scheme,

$$\phi_i^{n+1} = \phi_i^n + w_1\,\frac{\kappa\Delta t}{\Delta x^2}\,\left(\phi_{i+1}^n + \phi_{i-1}^n - 2\phi_i^n\right) + w_2\frac{\kappa\Delta t}{4\Delta x^2}\left(\phi_{i+2}^{n-1} + \phi_{i-2}^{n-1} - 2\phi_i^{n-1}\right), \tag{B.3}$$

we obtain the effective equation as,

$$\frac{\partial \phi}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 \phi}{\partial t^2} \approx \kappa\frac{\partial^2 \phi}{\partial x^2} - \kappa\Delta t w_2\frac{\partial^3 \phi}{\partial t\partial x^2} + \kappa\frac{\Delta x^2}{12}\left(w_1 + 4w_2\right)\frac{\partial^4 \phi}{\partial x^4} + O(\Delta x^4, \Delta t^2). \tag{B.4}$$

Here, ϕ_i^{n+1} on the L.H.S. of Eq. B.4 has been expanded about ϕ_i^n as follows.

$$\phi_i^{n+1} = \phi_i^n + \Delta t \frac{\partial \phi}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \phi}{\partial t^2} + \frac{\Delta t^3}{6}\frac{\partial^3 \phi}{\partial t^3} + ... \tag{B.5}$$

In order to simply further, we need to eliminate the cross-derivatives in Eq. B.4. For this sake, we differentiate the equation twice w.r.t x and once w.r.t t and eliminate higher order terms to obtain the following relations.

$$\frac{\partial^3 \phi}{\partial x^2 \partial t} \approx \kappa \frac{\partial^4 \phi}{\partial x^4} + O(\Delta x^2, \Delta t), \tag{B.6}$$

$$\frac{\partial^2 \phi}{\partial t^2} \approx \kappa \frac{\partial^3 \phi}{\partial t \partial x^2} + O(\Delta x^2, \Delta t). \tag{B.7}$$

The above equations can be merged as

$$\frac{\partial^2 \phi}{\partial t^2} \approx \kappa^2 \frac{\partial^4 \phi}{\partial x^4} + O(\Delta x^2, \Delta t) \tag{B.8}$$

Finally, substituting Eq. B.6 and Eq. B.8 back in Eq. B.4, we obtain,

$$\frac{\partial \phi}{\partial t} + \frac{\Delta t}{2}\kappa^2 \frac{\partial^4 \phi}{\partial x^4} = \kappa \frac{\partial^2 \phi}{\partial x^2} - \kappa^2 \Delta t w_2 \frac{\partial^4 \phi}{\partial x^4} + \kappa \frac{\Delta x^2}{12}(w_1 + 4w_2)\frac{\partial^4 \phi}{\partial x^4} + O(\Delta x^4, \Delta t^2), \tag{B.9}$$

and upon further simplification, this yields the effective equation as

$$\frac{\partial \phi}{\partial t} = \kappa \frac{\partial^2 \phi}{\partial x^2} + \frac{\kappa \Delta x^2}{2}\left(\frac{w_1 + 4w_2}{6} - (1 + 2w_2))\,\alpha\right)\frac{\partial^4 \phi}{\partial x^4} + O(\Delta x^4, \Delta t^2). \tag{B.10}$$

Appendix C

Code Optimization

We present here the optimized code implementation for the weighted scheme. The optimized version of the weighted solver is presented in the code listing below.

```
while(curr_time<=final_time)
{
  curr_time += (2*time_step);
  iter = iter+2;
  double temp = f[nX-5], tempO = fOld[nX-5];
  double tm1[2], tm2[2], tm3[2];
  double tn1[2], tn2[2], tn3[2], tn4[4];
  double tk1[2], tk2[4], tk3[4];
  tm3[0] = fOld[1]; tm3[1] = fOld[2];
  tm2[0] = fOld[3]; tm2[1] = fOld[4];
  tn3[0] = f[0]; tn3[1] = f[1];
  tn2[0] = f[2]; tn2[1] = f[3];

  tk3[0] = fOld[1] = (factor*f[nX-3])+(w1*(f[nX-2]+f[nX-4]))
         + (w2*(fOld[nX-1] + fOld[nX-5] - (2*fOld[nX-3])));

  tk2[0] = fOld[2] = (factor*f[2]) + (w1*(f[3] + f[1]))
         + (w2*(fOld[0] + fOld[4] - (2*fOld[2])));

  for(i=3; i<nX-3; i=i+2)
  {
     for(int k=0;k<2;k++)
            tm1[k]=fOld[i+2+k];         # 1 read
```

```
    for(int k=0;k<3;k++)
        tn4[k]=f[i+k];                    # 1 read

    tn1[0] = tn4[1]; tn1[1] = tn4[2];

    for(int k=0;k<2;k++)
    {
        tk1[k] = (factor*tn4[k]) + (w1*(tn1[k] + tn2[k]))
                  + (w2*(tm1[k] + tm3[k] - (2*tm2[k])));

        # 1 write
        f[i-1+k] = (factor*tk2[k]) + (w1*(tk1[k]+tk3[k]))
                    + (w2*(tn1[k] + tn3[k] - (2*tn2[k])));

        tk3[1-k] = tk2[k];
        tk2[1-k] = tk1[k];
    }

    for(int k=0;k<2;k++)
        fOld[i+k] =tk1[k];                # 1 write

    for(int k=0;k<2;k++)
    {
       tm3[k] = tm2[k];
       tm2[k] = tm1[k];
    }

    for(int k=0;k<2;k++)
    {
        tn3[k] = tn2[k];
        tn2[k] = tn1[k];
    }
}

if(nX%2==0)
```

```
    {
        fOld[nX-3] = (factor*f[nX-3]) + (w1*(f[nX-2]+f[nX-4]))
                     + (w2*(fOld[nX-1]+temp0-2*fOld[nX-3]));

        f[nX-4] = (factor*fOld[nX-4]) + (w1*(fOld[nX-3]
                  + fOld[nX-5])) + (w2*(f[nX-2]+tn3[0]
                  -(2*f[nX-4])));
    }

    periodic_boundary_condition(fOld,nX);

    f[nX-3] = (factor*fOld[nX-3]) + (w1*(fOld[nX-2]
                 + fOld[nX-4]))
              + (w2*(f[nX-1] + temp - (2*f[nX-3])));


    periodic_boundary_condition(f,nX);
}
```

In the above code implementation, we are reading values from `fOld` (ϕ^{n-1}) and `f` (ϕ^{n}) and we are updating `fOld` (ϕ^{n+1}) and `f` (ϕ^{n+2}). So, effectively we are doing 2 read and 2 write operations for two consecutive time step updates and hence basically 1 read and 1 write operation for updating each grid point per time step. In the process of optimizing the code, we have made use of temporary variables. The use of temporary variables allows for data residing in cache to be reused in further computations. This approach essentially minimizes the need to go to main memory for fetching data thereby reducing the memory bandwidth pressure which is the performance bottleneck for stencil based computations.