2014

Richardson Extrapolation-Based High Accuracy High Efficiency Computation for Partial Differential Equations

Ruxin Dai
University of Kentucky, ruxin.dai@uky.edu

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained and attached hereto needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine).

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless a preapproved embargo applies. I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's dissertation including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Ruxin Dai, Student

Dr. Jun Zhang, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

RICHARDSON EXTRAPOLATION-BASED HIGH ACCURACY HIGH EFFICIENCY COMPUTATION FOR PARTIAL DIFFERENTIAL EQUATIONS

DISSERTATION

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the College of Engineering at the University of Kentucky

By

Ruxin Dai

Lexington, Kentucky

Director: Jun Zhang, Ph.D., Professor of Computer Science

Lexington, Kentucky

2014

ABSTRACT OF DISSERTATION

RICHARDSON EXTRAPOLATION-BASED
HIGH ACCURACY HIGH EFFICIENCY COMPUTATION
FOR PARTIAL DIFFERENTIAL EQUATIONS

In this dissertation, Richardson extrapolation and other computational techniques are used to develop a series of high accuracy high efficiency solution techniques for solving partial differential equations (PDEs).

A Richardson extrapolation-based sixth-order method with multiple coarse grid (MCG) updating strategy is developed for 2D and 3D steady-state equations on uniform grids. Richardson extrapolation is applied to explicitly obtain a sixth-order solution on the coarse grid from two fourth-order solutions with different related scale grids. The MCG updating strategy directly computes a sixth-order solution on the fine grid by using various combinations of multiple coarse grids. A multiscale multigrid (MSMG) method is used to solve the linear systems resulting from fourth-order compact (FOC) discretizations. Numerical investigations show that the proposed methods compute high accuracy solutions and have better computational efficiency and scalability than the existing Richardson extrapolation-based sixth order method with iterative operator based interpolation.

Completed Richardson extrapolation is explored to compute sixth-order solutions on the entire fine grid. The correction between the fourth-order solution and the extrapolated sixth-order solution rather than the extrapolated sixth-order solution is involved in the interpolation process to compute sixth-order solutions for all fine grid points. The completed Richardson extrapolation does not involve significant computational cost, thus it can reach high accuracy and high efficiency goals at the same time.

There are three different techniques worked with Richardson extrapolation for computing fine grid sixth-order solutions, which are the iterative operator based interpolation, the MCG updating strategy and the completed Richardson extrapolation. In order to compare the accuracy of these Richardson extrapolation-based sixth-order methods, truncation error analysis is conducted on solving a 2D Poisson equation. Numerical comparisons are also carried out to verify the theoretical analysis.

Richardson extrapolation-based high accuracy high efficiency computation is extended to solve unsteady-state equations. A higher-order alternating direction im-

plicit (ADI) method with completed Richardson extrapolation is developed for solving unsteady 2D convection-diffusion equations. The completed Richardson extrapolation is used to improve the accuracy of the solution obtained from a high-order ADI method in spatial and temporal domains simultaneously. Stability analysis is given to show the effects of Richardson extrapolation on stable numerical solutions from the underlying ADI method.

KEYWORDS: Partial differential equations, high-order compact schemes, Richardson extrapolation, multiple coarse grids, multiscale multigrid method

Ruxin Dai

May 6, 2014

RICHARDSON EXTRAPOLATION-BASED HIGH ACCURACY HIGH EFFICIENCY COMPUTATION FOR PARTIAL DIFFERENTIAL EQUATIONS

By

Ruxin Dai

Jun Zhang, Ph.D.

Director of Dissertation

Miroslaw Truszczynski, Ph.D.

Director of Graduate Studies

May 6, 2014

Date

ACKNOWLEDGEMENTS

The past five years' PhD study has been an important and memorable time in my life. Without the guidance from my advisor and other committee members, help from friends, and support from my family, I would never have been able to finish my dissertation.

First of all, I would like to express my deepest appreciation to my academic advisor, Dr. Jun Zhang, for his excellent guidance, patience, encouragement, and caring. It is Dr. Zhang who led me to enter the computational science field and trained me to become a researcher. Dr. Zhang has been a great mentor on every account. Because of his broad knowledge and keen professional insight, I can start my research in a correct and clear direction and finally form the contents of my dissertation. In the dissertation writing process, Dr. Zhang also spends plenty of time reviewing my work and providing many constructive suggestions.

Next, I would like to thank the other faculty members of my Advisory Committee: Dr. Grzegorz Wasilkowski (Department of Computer Science), Dr. Dakshnamoorthy Manivannan (Department of Computer Science), Dr. Cai-Cheng Lu (Department of Electrical and Computer Engineering) for their helpful comments on my dissertation.

Then, I would like to thank my research collaborators: Dr. Yin Wang, Department of Math and Computer Science, Lawrence Technological University, Michigan, for his outstanding work on building multiscale multigrid computational framework and his continuous help and encouragement during my PhD study; Dr. Yongbin Ge, Institute of Applied Mathematics and Mechanics, Ningxia University, China, for his great mentoring on high order compact difference schemes and multigrid methods; and Mr. Qiao Liang, Department of Mathematics, University of Kentucky, Kentucky, for his helpful discussions on stability analysis for unsteady-state partial differential equations.

Also, I would like to thank Dr. Jun Zhang, Dr. Grzegorz Wasilkowski, Dr. Debby

Keen, and Dr. Yin Wang for their kind assistance with writing recommendation letters and helping me in my job search.

Thanks also go to all members in the Laboratory for High Performance Scientific Computing & Computer Simulation and Laboratory for Computational Medical Imaging & Data Analysis during my study: Dr. Yin Wang, Dr. Xuwei Liang, Dr. Changjiang Zhang, Dr. Dianwei Han, Dr. Ning Cao, Dr. Zhenhuan Zhou, Dr. Yongbin Ge, Dr. Lingjuan Li, Dr. Yingjin Lu, Dr. Jue Wu, Dr. Nirmal Thapa, Mr. Lian Liu, Mr. Qi Zhuang, Mr. Xiwei Wang and Mr. Pengpeng Lin. I want to thank them for their helpful suggestions and creating a friendly working environment.

At last, I would like to thank my family. I thank my father for his continuous support and encouragement. I thank my mother for giving me my life and her endless love.

Table of Contents

List of Tables

List of Figures

1 Introduction

Computational science and engineering (CSE) is a rapidly growing multidisciplinary field that deals with the development and application of computational models and simulations to solve complex physical problems, such as global weather forecasting, ocean modeling, combustion simulations, automobile crash studies, fluid dynamics, and oil and gas exploration. Computer modeling and simulation is essential because it provides the capability to enter fields that are either inaccessible or prohibitively expensive to carry out traditional experimentation. Since partial differential equations (PDEs) such as Poisson equation, convection-diffusion equations, and Navier-Stokes equations form the governing equations of most CSE modeling and simulation applications, numerical solutions of PDEs, as a key issue, have been the topic of research interest for many years.

There are two typical steps to solve PDEs through numerical methods. The first step is to discretize a PDE to obtain a linear system, which changes a continuous problem into a discrete problem. The commonly used numerical techniques include finite difference methods, finite element methods, and finite volume methods. The selection of discretization method for PDEs is application-oriented. The finite difference methods are useful for simple geometry domains because they are easy to implement and can reach higher-order accuracy. The finite element methods and finite volume methods are often applied to complex domains because they are allowed to use unstructured meshes. The second step is to solve the linear system from the discretized PDE. Since the resulting linear systems are usually large scale sparse linear systems, iterative methods stand out because of their easily implementation on high performance computers and they are faster for large systems if they converge fast [61]. The solvers for sparse linear systems mainly include basic iterative methods (Jacobi, Gauss-Seidel, and Successive Overrelaxation), Krylov subspace methods, and

multigrid methods.

The main goal of the numerical computation is to seek an approximate solution with acceptable accuracy in the least amount of computing time. As for numerical solutions of PDEs, the discretization method controls the solution accuracy because the discretization error becomes dominant when iterative methods converge. The linear system solver dictates the overall computing time because majority of the run-time for solving PDEs is spent on solving resulting linear systems. In general, people who study discretization methods and linear system solvers have their exclusive goals in mind. The motivation of my research work is to develop and analyze numerical algorithms for solving PDEs with both accuracy and efficiency goals in mind.

This dissertation mainly focuses on seeking high accuracy and high efficiency numerical techniques for solving PDEs over simple geometry domains with finite difference methods. For the purpose of high accuracy, Richardson extrapolation and high-order discretization schemes, especially high-order compact (HOC) difference schemes, are utilized. To obtain high efficiency, multiscale multigrid (MSMG) computation, multiple coarse grid (MCG) computation and alternating direction implicit (ADI) method are involved. The following parts introduce these numerical methods and computational techniques.

1.1 Richardson Extrapolation Technique

Richardson extrapolation is a sequence acceleration method used to improve the rate of convergence of a sequence, which was introduced by Lewis Fry Richardson in the early of the 20th century [56]. In introductory courses of numerical methods, it is taught as the basis of Romberg integration [11]. To increase the order of accuracy of numerical approximation through Richardson extrapolation, the numerical approximations using related discretization can be combined to remove the leading order error term and thus obtain a higher-order numerical approximation.

Assume $U(h)$ is a numerical approximation of order p to an exact solution U^*. The objective is to obtain the exact solution as h goes to 0. With the assumption, the numerical approximation can be expanded as

$$U(h) = U^* + Ah^p + O(h^{p+1})$$

with p being some known constant, A being some other (usually unknown) expression and independent of h, and $O(h^{p+1})$ being a sum of terms of order $p+1$ and higher on h. In order to remove the leading order error term Ah^p and obtain a more accurate approximation $\tilde{U}(h)$, consider another numerical approximation with discretization size rh (r as a given refinement ratio and usually $0 < r < 1$)

$$U(rh) = U^* + Ar^p h^p + O(h^{p+1}).$$

By multiplying $U(h)$ by r^p and subtracting off $U(rh)$, the Richardson extrapolation formula for the improved approximation $\tilde{U}(h)$ is

$$\tilde{U}(h) = \frac{r^p U(h) - U(rh)}{r^p - 1} = U^* + O(h^{p+1}). \tag{1.1}$$

If the original numerical scheme does not have an error term of the form h^{p+1}, then the order of accuracy of the extrapolated approximation $\tilde{U}(h)$ is based on the error term of the next lowest order on h.

The quantity being approximated in the simple formula (1.1) can be anything, such as an integral, a derivative, a solution to an ordinary differential equation or a solution to a partial differential equation. It does not require knowledge of the underlying methodology, except that the order of accuracy must be known. Just like "black boxes" for many modern computational tools, Richardson extrapolation can be viewed as a manipulation tool for the input or output of these black boxes without interfering with the details of the implementation within the black box [9]. Therefore, it is an efficient computational technique which requires minimal effort to increase the accuracy. In regards to PDEs, Richardson extrapolation has been used to increase the accuracy of their solutions in [47, 60, 57, 69, 79].

1.2 High-Order Compact (HOC) Difference Schemes

Traditional finite difference schemes, such as the central difference scheme (CDS) which computes approximate solutions with second-order accuracy, require very fine meshes to achieve satisfactory solution accuracy. For large scale simulations and modeling applications in many CSE applications, very fine meshes cause an extremely high computational cost. In order to curtail the computational cost and to get acceptable accuracy, large grid spaces with high-order (higher than two) difference schemes are needed. Using straightforward central differences, higher-order accuracy requires a larger stencil. This, however, may give rise to a problem at the points close to the boundaries, and also increases the bandwidth of the coefficient matrix, which makes fast direct solvers difficult to apply. Therefore, HOC finite difference schemes become noticeable because they are able to offer highly accurate numerical solutions on relatively coarser grids with greater computational efficiency. Here "compact" means that these schemes only use the center node and the adjacent nodes in each dimension. Although HOC difference schemes require more complicated developing procedures for matrix coefficient computation, they usually generate linear systems of much smaller size [1, 33].

For the development methods of HOC difference schemes, they can be classified into two categories. One is known as implicit methods, such as [12, 43], which compute the solution of the dependent variables and their first and second derivatives at the same time. The major shortcoming of the implicit methods lies in high computational cost, especially when the approximations of the first and second derivatives are not needed for some applications. In addition, they are not stable for certain problems, which are the computed solutions that may be oscillatory when a large mesh-size is used [96]. Although using a finer mesh-size may avoid numerical oscillations, it is contrary to the motivation of using high-order schemes. Another category is called explicit methods, which compute the solution of the dependent variables directly

to avoid redundant computation. Literature [66, 67] shows that explicit schemes have better stability property and will suppress nonphysical oscillations. However, compared with implicit compact schemes, high-order explicit compact schemes are more complicated to develop [35, 94].

1.2.1 Fourth-order compact (FOC) difference schemes

In the past three decades, many fourth-order explicit compact difference schemes were developed for 2D equations [32, 44, 65, 66, 93, 97] and 3D equations [2, 35, 36, 67, 88, 91, 94]. These fourth-order schemes not only provide high accuracy approximations with good numerical stability [90], but also work very well with fast iterative solution methods, e.g., multigrid methods [34, 84, 89, 95].

There are mainly two strategies to develop fourth-order compact (FOC) schemes explicitly. One is based on the truncated Taylor series expansions, represented by [2, 32]. Their procedures are based upon giving the approximate value of a function at a mesh point as a linear combination of the analytic solutions of the differential equation. The finite difference schemes are obtained by collocation over a set of mesh points surrounding the given mesh point for which the difference formula is derived. The process of simplification is straightforward but extremely tedious. Another technique to develop FOC schemes considers a particular equation and employs CDS repeatedly. The discretization continues by expanding the leading truncation error term until a desired order of approximation is reached. The representative methods are from Spotz and Carey's work [66, 67].

To illustrate the development process of the HOC scheme, we take a brief introduction to the FOC difference scheme for solving a 1D Poisson equation of the form

$$u_{xx} = f(x), \qquad 0 \le x \le l, \tag{1.2}$$

with suitable boundary conditions. A uniform grid with mesh-size $\Delta = l/n$ is con-

structed, where n is the number of intervals. $f(x)$ is assumed to have the necessary derivatives up to certain orders. We denote $x_j = j\Delta$, $u_j = u(x_j)$, and $f_j = f(x_j)$, where $j = 0, 1, ..., n$. We have the second-order central difference operator as

$$\delta_x^2 u_j = \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta^2}, \qquad j = 1, 2, ..., n-1.$$

By using the Taylor series expansions, the second derivative u_{xx} at a grid point j can be approximated using the central difference operator as

$$u_{xx} = \delta_x^2 u_j - \frac{\Delta^2}{12} u_x^4 + O(\Delta^4). \tag{1.3}$$

To get the fourth-order compact approximation, the term $O(\Delta^4)$ can be ignored. But, we cannot drop the term $\frac{\Delta^2}{12} u_x^4$, unless it can be approximated further to fourth-order accuracy. Since $\frac{\Delta^2}{12} u_x^4$ has an Δ^2 factor, the key issue is to approximate the term u_x^4 to second-order accuracy.

we double differentiate Eq. (1.2) to get

$$u_x^4 = f_{xx}. \tag{1.4}$$

Applying the central difference operator on f_{xx}, we have $f_{xx} = \delta_x^2 f + O(\Delta^2)$. Hence, Eq. (1.4) can be approximated to second-order accuracy as

$$u_x^4 = \delta_x^2 f + O(\Delta^2). \tag{1.5}$$

Substituting (1.5) into (1.3) yields a fourth-order compact approximation for the second derivative

$$u_{xx} = \delta_x^2 u - \frac{\Delta x^2}{12} \delta_x^2 f + O(\Delta^4). \tag{1.6}$$

Hence, the fourth-order compact approximation scheme of the 1D Poisson equation is

$$\delta_x^2 u - \frac{\Delta x^2}{12} \delta_x^2 f = f + O(\Delta^4). \tag{1.7}$$

1.2.2 Sixth-order compact (SOC) difference schemes

Recently, there has been growing interest in developing sixth-order schemes. By using Taylor series expansion, Soptz and Carey [67] developed a compact scheme for the 3D Poisson equation which can achieve sixth-order accuracy only when the derivatives of source term can be determined analytically. Sutmann used Padé approximation discussed by Lele [43] on the Taylor expansion for the discretized Laplace operator to develop sixth-order compact schemes for the 3D Poisson equation [70] and the 3D Helmholtz equation [71]. Although the schemes need less grid points than the straightforward expansion approach, they are not fully compact since other grid points besides center and adjacent points are involved. Chu and Fan [12] proposed a three point combined compact difference (CCD) scheme for solving 2D Stommel Ocean model, which is a special convection-diffusion equation. Their scheme can achieve sixth-order accuracy for the inner grid points and fifth-order accuracy for the boundary grid points, but it is an implicit scheme which asks to compute the dependent variables and their derivatives together, resulting in a triple-tridiagonal system with high computational cost for solution. In addition, the CCD scheme has a stability problem that, for certain problems with a large mesh-size, the computed solution may be oscillatory [96]. There are other sixth-order schemes generated similarly [53, 42, 77], but all of them share common weak points such as: (1) derivatives of the source term appeared in the right-hand side which require analytical forms or approximations for the derivatives with certain order accuracy; (2) non-compact schemes which may cause problems at near-boundary points; (3) resulting complicated linear systems which increase the difficulty of choosing effective iterative solvers. As we know, there is no existing explicit sixth-order compact difference schemes on a single scale grid [78].

In this dissertation, we aim to study using Richardson extrapolation for sixth-order compact approximations. Although assumptions of smoothness and monotone

truncation error convergence in the mesh-size are involved, using Richardson extrapolation to reach sixth-order accuracy is more convenient than developing direct discretizations [60]. We can avoid complicated stencils, wide bandwidth matrices, and special considerations for near-boundary points, etc. In addition, highly efficient solvers for the resulting large sparse linear systems can be easily applied in such sixth-order methods. Sun and Zhang [69] first proposed to use Richardson extrapolation to obtain sixth-order solutions. The basic idea is applying Richardson extrapolation technique to the computed fourth-order solutions on two scale grids to remove fourth-order leading error terms. As for most uniform FOC schemes, their truncation error expressions do not have any fifth-order error term, thus the extrapolated solution can reach the sixth-order accuracy.

For the purpose of illustration, consider the 1D Poisson equation (1.2). The computational domain has uniform grids denoted by Ω. We compute a fourth-order accurate solution u_j^{Δ} at a grid point j by using the FOC scheme (1.7) on Ω_{Δ} with mesh-size Δ. After changing the mesh-size to 2Δ, we compute a similar fourth-order approximate solution $u_j^{2\Delta}$ at a grid point j on $\Omega_{2\Delta}$. Using the general Richardson extrapolation formula (1.1) and setting $p = 4$ and $r = 1/2$, a sixth-order solution $\tilde{u}^{2\Delta}$ can be computed by

$$\tilde{u}_j^{2\Delta} = \frac{(1/2)^4 u_j^{2\Delta} - u_{2j}^{\Delta}}{(1/2)^4 - 1} = \frac{(16 u_{2j}^{\Delta} - u_j^{2\Delta})}{15}. \tag{1.8}$$

Note that the sixth-order solution $\tilde{u}^{2\Delta}$ is computed on the coarse grid $\Omega_{2\Delta}$. Since we are interested in computing a sixth-order solution on the fine grid Ω_{Δ}, we inject $\tilde{u}^{2\Delta}$ from $\Omega_{2\Delta}$ to the corresponding even grid points on Ω_{Δ}. Sixth-order solutions at odd grid points on Ω_{Δ} can be computed by using some appropriate interpolation. Fig. 1.1 illustrates this process.

1.3 Multiscale Multigrid (MSMG) Method

Multigrid methods. For solving the resulting linear systems from discretized PDEs,

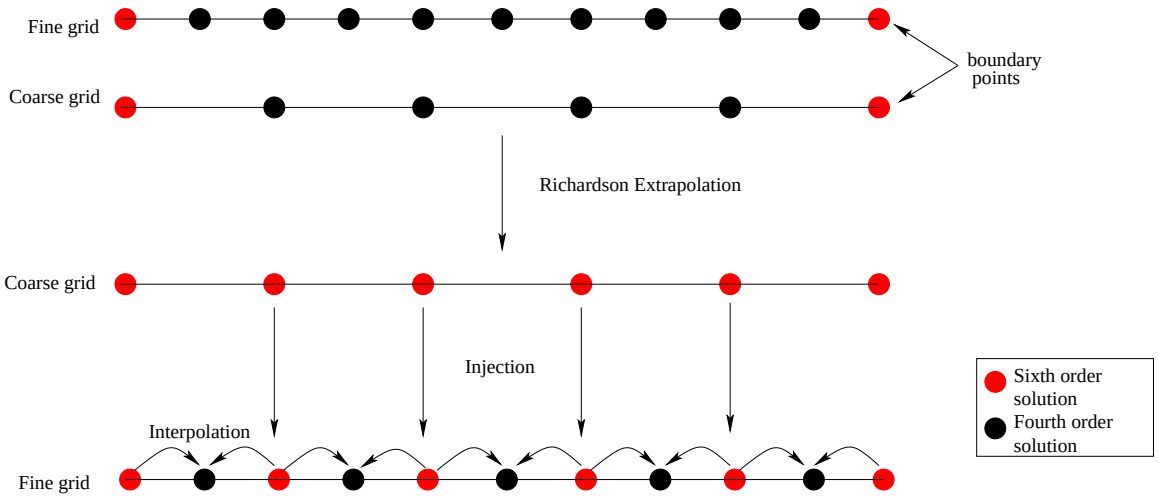


Figure 1.1: Illustration of the Richardson extrapolation and interpolation process in a 1D two-grid computation. Solution values at the boundary points are known.

multigrid methods are considered one of the most efficient contemporary iterative methods. The convergence rate of multigrid methods is independent of the grid size [4, 7]. Meanwhile, this approach often scales linearly with the number of unknowns. In other words, the computational complexity of multigrid methods is $O(n)$, where n is the number of unknowns. In [29, 36, 86, 89, 93, 95], previous scholars discussed various multigrid implementations with HOC schemes to solve 2D/3D Poisson and convection-diffusion equations.

The multigrid methods fully use multiscale grids to overcome the smoothing property of standard relaxation schemes and make the relaxation more effective. Due to the smoothing property, the standard relaxation schemes, such as Jacobi and Gauss-Seidel methods, can only effectively eliminate the oscillatory error components, and begin to stall when the smooth error components become dominant. In order to remove all error components effectively, multiscale grids are used and thus give birth to multigrid methods.

There are two strategies of utilizing multiscale grids to improve relaxation. One is to use coarser grids to generate improved initial guesses for finer grids, which is called nested iteration [7]. Obviously, relaxation on a coarse grid is less expensive

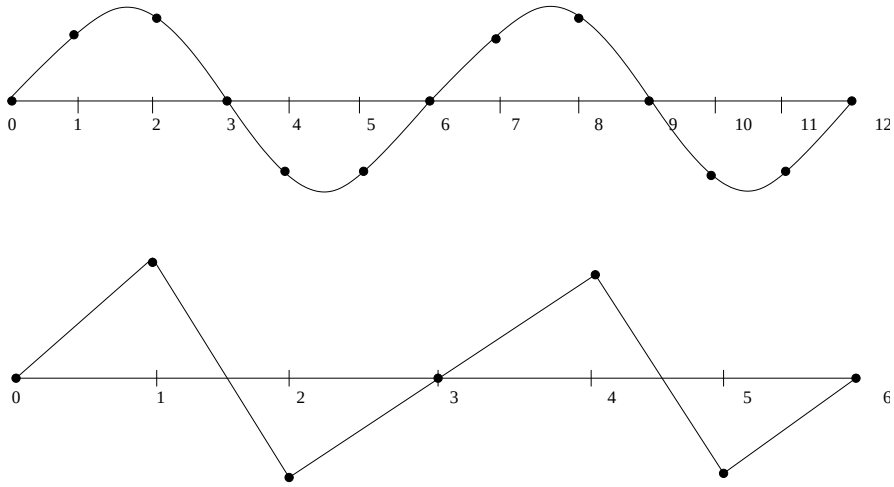


Figure 1.2: Smooth error component on Ω_Δ projected onto $\Omega_{2\Delta}$.

than on a fine grid because there are fewer unknowns to update. Although we obtain some improvement from using the coarse grids in this way, smooth error components still remain and the final iteration will stall. A second strategy focuses on removing smooth error components by using different scale grids. There is an observation that smooth error components look more oscillatory on a coarser grid, as in Fig. 1.2. Based on the idea that the residual equation on the coarse grid has a similar structure as the original problem on the fine grid, the residual is projected to the coarser grid and solved there. This procedure is known as the correction scheme [7], which consists of smoothing the error using a standard relaxation scheme (the smoother), restricting the residual to the coarse grid, solving the residual equation on the coarse grid to obtain an approximation of error correction, interpolating the error correction to the fine grid, and finally adding the error correction into the current approximation. In this process, the relaxation and error correction work together to remove both the oscillatory and smooth error components.

In practice, the coarse grid has twice the mesh-size of the fine grid. In respect to the intergrid transfers in the correction scheme, there are two classes of operations. One is transferring the error approximation from the coarse grid to the fine grid, which is generally called interpolation or prolongation. Interpolation is most effective when

the error is smooth. This process provides a perfect complement to relaxation, which is most effective for the oscillatory error. There are many interpolation methods that could be used. Bilinear (or trilinear) interpolation is mainly used in this dissertation. The other operation involves moving residual vectors from the fine grid to the coarse grid, which is known as restriction. The most obvious restriction operator is injection. In this dissertation, a more accurate operator, called full weighting [7], is used.

When the correction scheme is recursively applied to the residual equation on the coarse grids, a standard multigrid method, known as V-cycle algorithm, is constructed. In a multigrid $V(\nu_1, \nu_2)$-cycle algorithm, we carry out ν_1 relaxation sweeps on a given grid before going to a coarser grid and ν_2 relaxation sweeps after adding the coarse grid correction to the current approximation. ν_1 and ν_2 are called presmoothing sweeps and postsmoothing sweeps, respectively. Algorithm 1 gives the definition of V-cycle scheme. The notation $A^\Delta u^\Delta = f^\Delta$ denotes the linear system to be solved. $I_\Delta^{2\Delta}$ and $I_{2\Delta}^\Delta$ are the restriction and interpolation operators, respectively.

Algorithm 1 $V(\nu_1, \nu_2)$-cycle scheme (Recursive Definition)

1: **procedure** $V^\Delta(v^\Delta, f^\Delta)$
2: Relax ν_1 times on $A^\Delta u^\Delta = f^\Delta$ with a given initial guess v^Δ.
3: **if** Ω_Δ = coarsest grid **then** go to line 10.
4: **else**
5: $f^{2\Delta} \leftarrow I_\Delta^{2\Delta}(f^\Delta - A^\Delta v^\Delta)$,
6: $v^{2\Delta} \leftarrow 0$,
7: $v^{2\Delta} \leftarrow V^{2\Delta}(v^{2\Delta}, f^{2\Delta})$.
8: **end if**
9: Correct $v^\Delta \leftarrow v^\Delta + I_{2\Delta}^\Delta v^{2\Delta}$.
10: Relax ν_2 times on $A^\Delta u^\Delta = f^\Delta$ with initial guess v^Δ.
11: **end procedure**

There are two other kinds of multigrid schemes. One is called W-cycle, which is the multigrid method with two corrections. The other one is called full multigrid (FMG) scheme, in which each V-cycle is preceded by a coarse-grid V-cycle designed to provide the best initial guess possible. The FMG scheme fully uses the multiscale grids and can be viewed as the combination of nested iteration and recursive correction

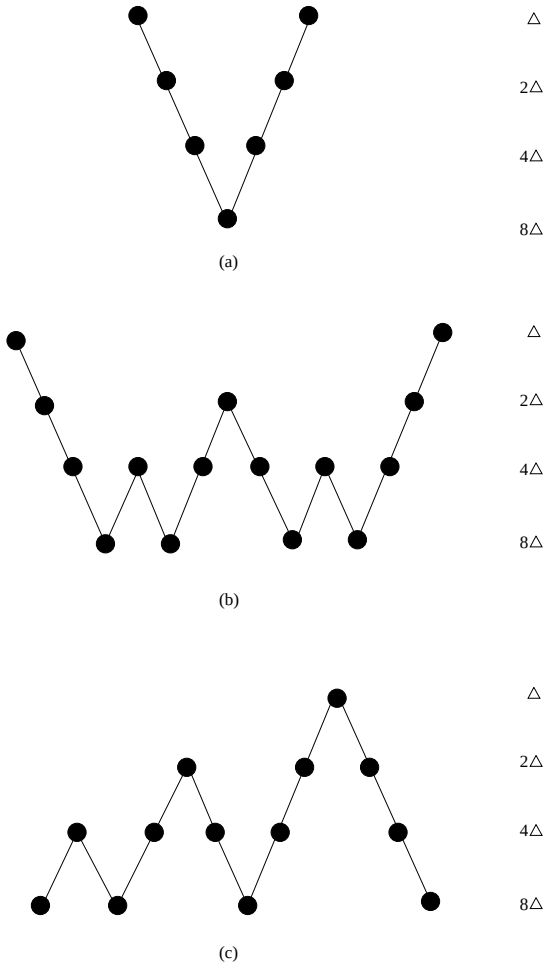


Figure 1.3: Illustration of three different multigrid schemes on four levels: (a) V-cycle (b) W-cycle (c) FMG scheme.

scheme. Fig. 1.3 illustrates the structure of three multigrid schemes.

Multiscale multigrid (MSMG) method. Although we have high-order discretization methods, such as HOC schemes, as well as fast iterative methods, such as multigrid methods, the studies on the discretization process and the linear system solvers are generally carried on by two different groups of people with their own goals in mind. People studying high accuracy discretization schemes may not care about how the resulting linear systems will be solved. While, people developing efficient linear system solvers may pay little attention to the sources of these linear systems. The MSMG method, first proposed in [79], aims to accelerate the linear system compu-

tation by using multigrid methods and to obtain a higher-order accurate solution by using a multiscale strategy to extrapolate on two lower-order solutions computed from different level discretizations. The most important feature of the MSMG method is the seamless integration of multiscale and multigrid computation. As a result, the high accuracy solution and high speed computation are achieved within the same framework. On one hand, the MSMG method offers the convergence rate that is independent of the grid size, a feature that is presented in multigrid methods. On the other hand, it utilizes different scale grids involved in the sixth-order approximation to provide better initial guesses and thus accelerates the convergence rate.

In Section 1.2.2, a class of Richardson extrapolation-based sixth-order methods is described. The MSMG method is particularly designed for this kind of computation. Richardson extrapolation asks for using two different discretized grids. It would not be cost-effective to construct a coarse grid exclusively for this purpose. Fortunately, different scale coarse grids are generated when using multigrid methods to solve the discretized equations. The MSMG method skillfully fuses the Richardson extrapolation procedure and multigrid methods for computing higher-order solutions of PDEs. The salient superiority of this method is to efficiently utilize the multilevel grids to accelerate the iterative process of the linear system and to enhance the order of accuracy of the computed solution simultaneously, not separately.

In recent years, the MSMG method has been implemented and applied to solve various PDEs, which is shown to be very efficient and stable [80, 81]. The MSMG method is structurally similar to the FMG scheme, but the computation does not start from the coarsest grid, as in Fig. 1.4. It first computes on $\Omega_{4\Delta}$, then goes to compute on $\Omega_{2\Delta}$ and Ω_{Δ}. The solutions from the coarser grids are used as the initial guesses for the finer grids. The MSMG method uses the solution $u^{4\Delta}$ as the initial guess to compute the solution $u^{2\Delta}$ on $\Omega_{2\Delta}$, and uses the solution $u^{2\Delta}$ as the initial guess to compute for u^{Δ} on Ω_{Δ}.

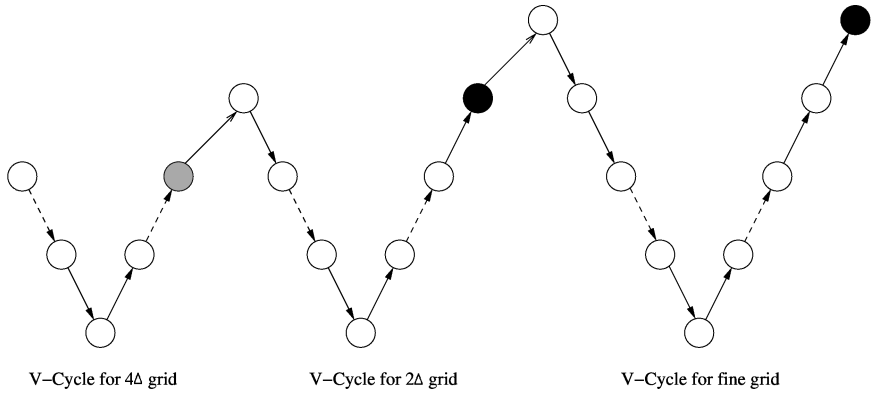


Figure 1.4: Illustration of the standard multiscale multigrid method.

1.4 Multiple Coarse Grid (MCG) Computation

The idea of MCG computation can be traced back to the parallel superconvergent multigrid method [24]. The superconvergent multigrid method uses multiple coarse grids to realize parallelization by solving many coarse scale problems simultaneously and to speed up convergence rates by generating a better correction for the fine grid solution than the correction from a single coarse grid. The appearance of multiple coarse grids is from the observation that for a 1D fine grid there are two kinds of grid points - the *even* fine grid points and the *odd* fine grid points, which construct two coarse grids, as in Fig. 1.5. In Fig. 1.5, the fine grid on Ω^{Δ} is coarsened into two coarse grids on $\Omega^1_{2\Delta}$ and $\Omega^2_{2\Delta}$. The coarse grid 1 is composed by *even* fine grid points and boundary points, while the coarse grid 2 is composed by *odd* fine grid points and boundary points.

In general, for a d dimensional problem, the fine grid can easily be coarsened into 2^d coarse grids. For instance, for a 2D fine grid, there are four kinds of grid points which generate four coarse grids. For a 3D fine grid, eight kinds of fine grid points can lead to eight coarse grids.

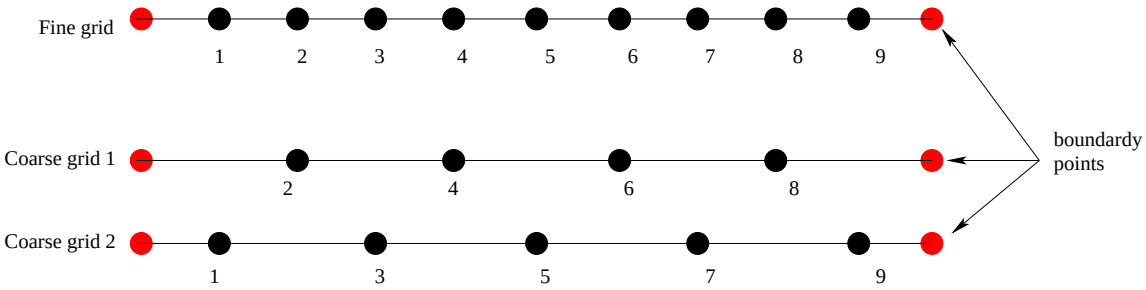


Figure 1.5: Illustration of the multiple coarse grids for 1D problem.

1.5 Alternating Direction Implicit (ADI) Method

Alternating direction implicit methods [17, 18, 55, 72] are popular non-iterative methods for solving 2D/3D parabolic differential equations. The main idea is to reduce multi-dimensional problems to a series of one-dimensional problems and solving a sequence of tridiagonal linear systems. Hence, the overall computation is simple and relatively fast. Among various ADI schemes, we consider the Peaceman-Rachford ADI scheme [55] in this dissertation.

Consider a 2D heat equation on the unit square

$$u_t = u_{xx} + u_{yy} + f(x, y, t), \qquad (x, y) \in (0, 1) \times (0, 1), t \in (0, T], \tag{1.9}$$

with Dirichlet boundary and given initial conditions. In order to solve Eq. (1.9), a uniform grid with mesh-sizes h in both x and y directions is constructed. The temporal domain is discretized by time step size Δt. The approximate solution at (x_i, y_j, t_n) is denoted by $u^n_{i,j}$.

Using the 2D Crank-Nicolson scheme [72] to discretize (1.9) in time gives

$$u^{n+1}_{i,j} = u^n_{i,j} + \frac{\Delta t}{2}[(u_{xx} + u_{yy})^{n+1}_{i,j} + (u_{xx} + u_{yy})^n_{i,j} + f^{n+1}_{i,j} + f^n_{i,j}] + O(\Delta t^3). \tag{1.10}$$

We rewrite (1.10) as

$$u^{n+1}_{i,j} - \frac{\Delta t}{2}(u_{xx} + u_{yy})^{n+1}_{i,j} = u^n_{i,j} + \frac{\Delta t}{2}(u_{xx} + u_{yy})^n_{i,j} + \frac{\Delta t}{2}(f^{n+1}_{i,j} + f^n_{i,j}). \tag{1.11}$$

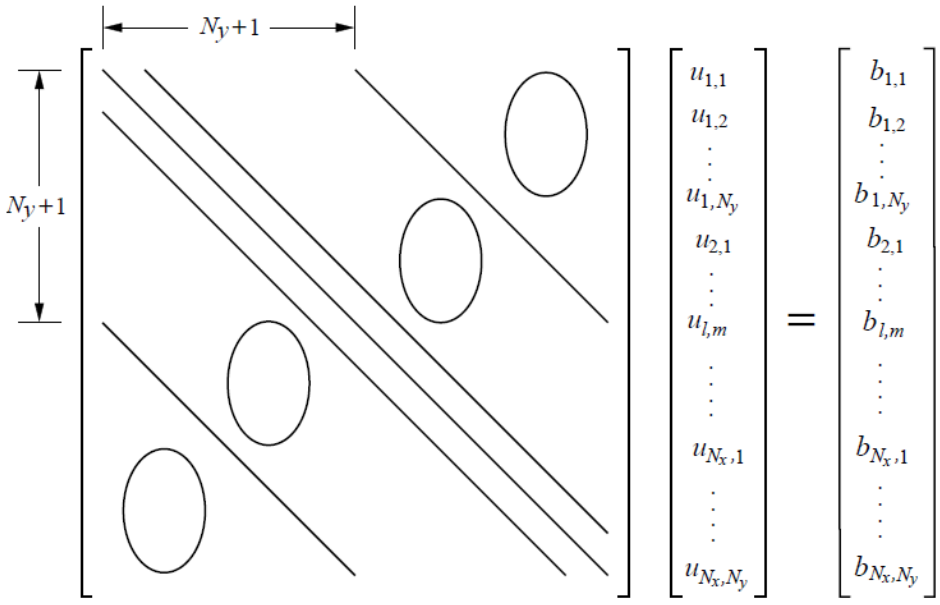


Figure 1.6: Matrix structure for 2D Crank-Nicolson method.

By using the CDS to approximate the spatial derivatives in (1.11), we obtain

$$u_{i,j}^{n+1} - \frac{\Delta t}{2h^2}(u_{i-1,j}^{n+1} + u_{i,j-1}^{n+1} - 4u_{i,j}^{n+1} + u_{i,j+1}^{n+1} + u_{i+1,j}^{n+1}) \\ = u_{i,j}^{n} + \frac{\Delta t}{2h^2}(u_{i-1,j}^{n} + u_{i,j-1}^{n} - 4u_{i,j}^{n} + u_{i,j+1}^{n} + u_{i+1,j}^{n}) + \frac{\Delta t}{2}(f_{i,j}^{n+1} + f_{i,j}^{n}). \quad (1.12)$$

The linear system from (1.12) has the structure displayed in Fig. 1.6, where N_x is the number of grid intervals along the x direction and N_y is the number of grid intervals along the y direction. This coefficient matrix is a large sparse matrix with high degree and consists of five nonzero bands. Neither direct elimination nor sparse LU composition can be applied. However, ADI methods are able to convert the current system to a system of two sets of equations, each of which involves only one spatial direction and requires solution of only tridiagonal systems. Hence, the new system requires only $O(N)$ ($N = N_x \times N_y$) arithmetic operations per time step, in contrast to the $O(N^3)$ required by direct Gaussian elimination applied to the entire system [51].

To derive an ADI scheme, the approximation (1.12) can be rearranged and repre-

sented as

$$[I - \frac{\Delta t}{2}(A_x + A_y)]u^{n+1} = [I + \frac{\Delta t}{2}(A_x + A_y)]u^n + \frac{\Delta t}{2}(f^{n+1} + f^n), \tag{1.13}$$

where A_x and A_y are tridiagonal matrices.

In (1.13), the left-hand side matrix can be approximately factored as

$$(I - \frac{\Delta t}{2}A_x)(I - \frac{\Delta t}{2}A_y) = I - \frac{\Delta t}{2}(A_x + A_y) + \frac{\Delta t^2}{4}A_xA_y, \tag{1.14}$$

and similarly for the right-hand side,

$$(I + \frac{\Delta t}{2}A_x)(I + \frac{\Delta t}{2}A_y) = I + \frac{\Delta t}{2}(A_x + A_y) + \frac{\Delta t^2}{4}A_xA_y. \tag{1.15}$$

Notice that each of the two factored matrices on the left of these expressions is tridiagonal. In addition, their products are within $O(\Delta t^2)$ of the original unfactored matrices.

Substitute Eqs. (1.14) and (1.15) into (1.13) and obtain

$$(I-\frac{\Delta t}{2}A_x)(I-\frac{\Delta t}{2}A_y)u^{n+1} = (I+\frac{\Delta t}{2}A_x)(I+\frac{\Delta t}{2}A_y)u^n+\frac{\Delta t}{2}(f^{n+1}+f^n)+\frac{t^2}{4}(u^{n+1}-u^n). \tag{1.16}$$

If $u(x, y, t)$ is sufficiently smooth, we have

$$u_{i,j}^{n+1} - u_{i,j}^n = O(\Delta t)$$

for any (i, j).

Hence, the factorization

$$(I - \frac{\Delta t}{2}A_x)(I - \frac{\Delta t}{2}A_y)u^{n+1} = (I + \frac{\Delta t}{2}A_x)(I + \frac{\Delta t}{2}A_y)u^n + \frac{\Delta t}{2}(f^{n+1} + f^n) \tag{1.17}$$

is with $O(\Delta t^3)$ of the original 2D Crank-Nicolson scheme (1.10).

Splitting (1.17) into two equations gives

$$(I - \frac{\Delta t}{2}A_x)u^{n+1^*} = (I + \frac{\Delta t}{2}A_y)u^n + \frac{\Delta t}{2}f^n, \tag{1.18}$$

$$(I - \frac{\Delta t}{2}A_y)u^{n+1} = (I + \frac{\Delta t}{2}A_x)u^{n+1^*} + \frac{\Delta t}{2}f^{n+1}. \tag{1.19}$$

Thus, we calculate the advanced time step values in two consecutive steps. The first involves only x derivatives, while the second involves only y derivatives.

1.6 Organization

This dissertation is composed of seven chapters. The remainder is organized as follows:

- In the existing MSMG method, there is a computational efficiency issue caused by an iterative refinement procedure on the fine grid. In Chapter 2, a direct solution based on multiple coarse grids is proposed to replace the iterative refinement procedure for computing fine grid sixth-order solutions. An MSMG method with MCG updating strategy is presented and applied to solve 2D Poisson and convection-diffusion equations. Numerical investigations show that the MCG updating strategy is more efficient and scalable than the iterative refinement procedure for sixth-order accuracy computation.

- In Chapter 3, an improved MSMG method with MCG updating strategy for 3D convection-diffusion equations is presented. The new MCG updating strategy is used to replace the iterative refinement procedure in the existing MSMG method for 3D steady-state equations to obtain higher-order solutions on the fine grid. Since the proposed method needs an FOC scheme with unequal mesh-sizes, a 19-point FOC difference scheme with unequal mesh-size discretization is given for the 3D convection-diffusion equation. Numerical experiments are carried out to compare the computed accuracy and the computational efficiency of the MCG updating strategy against the iterative refinement procedure in computing sixth-order solutions with the MSMG method.

- Another Richardson extrapolation-based sixth-order solution for steady-state PDEs is described in Chapter 4. Completed Richardson extrapolation technique is used to obtain a sixth-order solution on the entire fine grid. Numerical experiments are conducted to test its high accuracy and high efficiency.

- Extrapolated sixth-order coarse grid solutions can be injected into the fine grid and make partial fine grid points obtain sixth-order solutions, but other techniques are needed to compute sixth-order solutions for the remaining fine grid points. There are three different techniques (iterative refinement procedure with operator based interpolation, multiple coarse grid updating strategy, and completed Richardson extrapolation), which lead to three kinds of Richardson extrapolation-based sixth-order methods. Chapter 5 analyzes the truncation errors from these three different methods respectively. Numerical comparisons on several test problems are also provided.

- In Chapter 6, a higher-order ADI method with completed Richardson extrapolation is proposed for solving unsteady 2D convection-diffusion equations. The method is sixth-order accuracy in space and third-order accuracy in time. Completed Richardson extrapolation is used to improve the accuracy of the solution in spatial and temporal domains simultaneously. A stability analysis is given to discuss the effects of Richardson extrapolation on solution stability. Numerical experiments are conducted to test the proposed method and to compare it with Karaa-Zhang's high-order ADI method.

- In Chapter 7, I summarize contributions of this dissertation and outlook for possible future research work.

2 Sixth-Order Solution with Multiscale Multigrid Method and Multiple Coarse Grid Updating Strategy for 2D Steady-State Equations

2.1 Introduction

This chapter discusses using the Richardson extrapolation technique, multiscale multigrid (MSMG) method and multiple coarse grid (MCG) computation for computing sixth-order solutions of 2D Poisson and convection-diffusion equations.

We first consider a 2D Poisson equation of the form

$$u_{xx}(x,y) + u_{yy}(x,y) = f(x,y), \qquad (x,y) \in \Omega, \tag{2.1}$$

where Ω is a rectangular domain, with suitable boundary conditions defined on $\partial\Omega$. The solution $u(x,y)$ and the forcing function $f(x,y)$ are assumed to be sufficiently smooth and have required continuous partial derivatives.

Recently, Zhang *et al.* proposed a series of explicit methods for sixth-order compact approximations by using Richardson extrapolation [69, 79, 80, 81]. Among these methods, the most efficient one is the MSMG method, which incorporates the sixth-order explicit compact computing strategy and multigrid solution idea [79]. In the existing MSMG computational framework. Richardson extrapolation is applied on two fourth-order computed solutions from two different scale uniform grids – Ω_Δ with mesh-size Δ and $\Omega_{2\Delta}$ with mesh-size 2Δ – to obtain a sixth-order solution on the standard coarse grid $\Omega_{2\Delta}$. The extrapolated solution is directly interpolated (injected) from the standard coarse grid to the fine grid, which makes the $(even, even)$ fine grid points obtain sixth-order solutions, as in Fig. 2.1. In Fig. 2.1, the grid points marked in red have sixth-order solutions, while the grid points marked in black have fourth-order solutions. Since the goal is to obtain a sixth-order solution on the fine grid Ω_Δ, Wang [78] used an operator based interpolation scheme to iteratively update the solution of black fine grid points. Fig. 2.2 shows the updating process of one interpolation iteration. However, this process is an iterative refinement procedure,

which is similar to basic iterative methods like Gauss-Seidel [61], and the convergence rate is usually slow. As the iterative refinement procedure is performed on the fine grid, it may take a number of iterations to converge. Thus, the computational cost becomes expensive. In this chapter, we want to reduce the computational cost by using an alternative method to directly calculate sixth-order solutions for all fine grid points. An updating strategy based on multiple coarse grids is developed and used to accelerate the MSMG computation by eliminating the iterative refinement procedure.

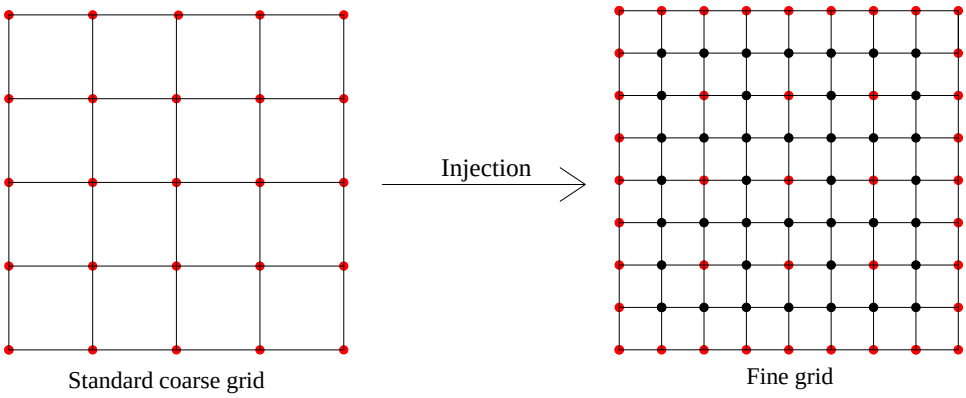


Figure 2.1: Injection from the standard coarse grid to the fine grid.

2.2 FOC Scheme with Unequal Mesh-Size Discretization for the 2D Poisson Equation

The Richardson extrapolation-based sixth-order methods involve fourth-order solutions on two different scale grids. In this section, we first introduce the fourth-order compact (FOC) scheme for the 2D Poisson equation. The basic idea stems from Zhang's previous work [93].

In order to discretize Eq. (2.1), consider a rectangular domain $\Omega = [0, L_x] \times [0, L_y]$ with mesh-sizes $\Delta x = L_x/N_x$ and $\Delta y = L_y/N_y$ in the x and y coordinate directions, respectively. Here N_x and N_y are the number of uniform intervals in the x and y coordinate directions, respectively. The mesh points are (x_i, y_j) with $x_i = i\Delta x$ and $y_j = j\Delta y$, $0 \le i \le N_x$, $0 \le j \le N_y$. In the following, we may also use the index pair (i, j) to represent the grid point (x_i, y_j).

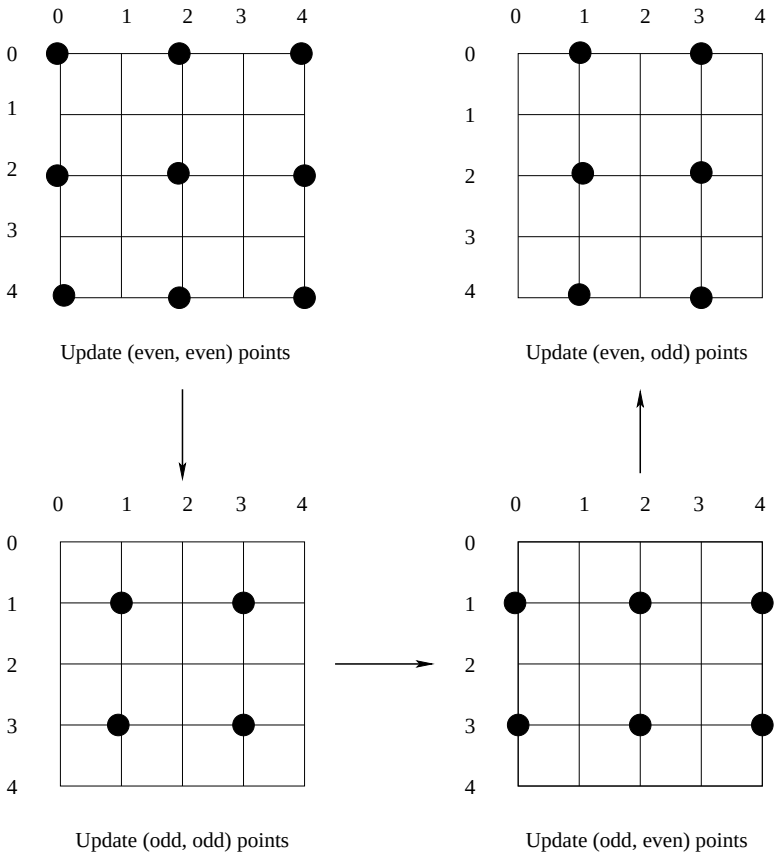


Figure 2.2: Operator based interpolation scheme for a 5×5 fine grid.

The standard second-order central difference operators defined at a grid point (x_i, y_j) can be written as

$$\delta_x^2 u_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}, \qquad \delta_y^2 u_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}.$$

Using Taylor series expansions at the grid point (x_i, y_j), we have

$$\delta_x^2 u_{i,j} = u_{xx} + \frac{\Delta x^2}{12} u_x^4 + \frac{\Delta x^4}{360} u_x^6 + O(\Delta x^6) \tag{2.2}$$

and

$$\delta_y^2 u_{i,j} = u_{yy} + \frac{\Delta y^2}{12} u_y^4 + \frac{\Delta y^4}{360} u_y^6 + O(\Delta y^6). \tag{2.3}$$

Recall the FOC scheme for the 1D Poisson equation introduced in Section 1.2.1, we rewrite (1.7) as

$$\delta_x^2 u = (1 + \frac{\Delta x^2}{12} \delta_x^2) f + O(\Delta^4), \tag{2.4}$$

which can be formulated symbolically as

$$(1 + \frac{\Delta x^2}{12}\delta_x^2)^{-1}\delta_x^2 u = f + O(\Delta^4), \tag{2.5}$$

where the operator $(1 + \frac{\Delta x^2}{12}\delta_x^2)^{-1}$ has symbolic meaning only.

We similarly have the symbolic fourth-order compact approximation operator for the y variable as

$$(1 + \frac{\Delta y^2}{12}\delta_y^2)^{-1}\delta_y^2 u = f + O(\Delta y^4). \tag{2.6}$$

We apply (2.5) and (2.6) to the second derivatives u_{xx} and u_{yy} in Eq. (2.1), respectively. This yields symbolically

$$(1 + \frac{\Delta x^2}{12}\delta_x^2)^{-1}\delta_x^2 u + (1 + \frac{\Delta y^2}{12}\delta_y^2)^{-1}\delta_y^2 u = f + O(\Delta^4), \tag{2.7}$$

where $O(\Delta^4)$ denotes the truncated terms in the order of $O(\Delta x^4 + \Delta y^4)$. Applying the symbolic operators and absorbing the $O(\Delta x^2 \cdot \Delta y^2)$ term into the $O(\Delta^4)$ generates

$$\begin{aligned}(1 + \frac{\Delta y^2}{12}\delta_y^2)\delta_x^2 u + (1 + \frac{\Delta x^2}{12}\delta_x^2)\delta_y^2 u &= (1 + \frac{\Delta x^2}{12}\delta_x^2)(1 + \frac{\Delta y^2}{12}\delta_y^2)f + O(\Delta^4) \\ &= [1 + \frac{1}{12}(\Delta x^2\delta_x^2 + \Delta y^2\delta_y^2)]f + O(\Delta^4).\end{aligned}$$

After some rearrangement and dropping the $O(\Delta^4)$ term, the general FOC scheme for the 2D Poisson equation is given by

$$(\delta_x^2 + \delta_y^2)u + \frac{1}{12}(\Delta x^2 + \Delta y^2)\delta_x^2\delta_y^2 u = f + \frac{1}{12}(\Delta x^2\delta_x^2 + \Delta y^2\delta_y^2)f. \tag{2.8}$$

If we denote the mesh aspect ratio $\gamma = \Delta x/\Delta y$, (2.8) changes into the following form with a 9-point computational stencil [93]

$$\begin{aligned}&au_{i,j} + b(u_{i+1,j} + u_{i-1,j}) + c(u_{i,j+1} + u_{i,j-1}) \\ &+d(u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1}) \\ &= \tfrac{\Delta x^2}{2}(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}),\end{aligned} \tag{2.9}$$

where the coefficients are

$$a = 10(1 + \gamma^2), \qquad b = -5 + \gamma^2, \qquad c = -5\gamma^2 + 1, \qquad d = -(1 + \gamma^2)/2.$$

Eq. (2.9) can be viewed as the general FOC scheme for the 2D Poisson equation on a rectangular domain. In a special case of $\Delta x = \Delta y = \Delta$, Eq. (2.9) can be written as

$$u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} + 4(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - 20u_{i,j} = \frac{\Delta^2}{2}(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}). \tag{2.10}$$

2.3 Richardson Extrapolation-based Sixth-Order Solution with MSMG Method and MCG Updating Strategy for the 2D Poisson Equation

2.3.1 Improving solution accuracy by Richardson extrapolation in 2D

The general Richardson extrapolation can be written as

$$\tilde{u}_{i,j}^{2\Delta} = \frac{(2^p u_{2i,2j}^{\Delta} - u_{i,j}^{2\Delta})}{2^p - 1}, \tag{2.11}$$

where p is the order of accuracy before the extrapolation, and the order of accuracy will be increased to $p + 2$ after the extrapolation [6, 56].

Consider a computational domain Ω discretized by uniform 2D grids. By using the FOC scheme (2.10), we compute fourth-order solutions u^{Δ} on the Ω_{Δ} grid with mesh-size Δ and $u^{2\Delta}$ on the $\Omega_{2\Delta}$ grid with mesh-size 2Δ, respectively. Then, the sixth-order solution on the $\Omega_{2\Delta}$ grid can be calculated by the Richardson extrapolation formula as

$$\tilde{u}_{i,j}^{2\Delta} = \frac{(16u_{2i,2j}^{\Delta} - u_{i,j}^{2\Delta})}{15}. \tag{2.12}$$

2.3.2 MCG updating strategy for 2D problems

Since our ultimate goal is to obtain a sixth-order solution on the fine grid Ω_{Δ}, the sixth-order solution on the coarse grid $\Omega_{2\Delta}$ can be injected into the fine grid and make the $(even, even)$ fine grid points obtain sixth-order solutions. Unlike Wang's method [79] using an operator based interpolation scheme on the fine grid to asymptotically approach sixth-order solutions for the remaining fine grid points, a direct calculation strategy is proposed.

The new strategy is inspired by the MCG computation introduced in Section 1.4. Fig. 2.3 illustrates that four coarse grids can be generated from a 2D fine grid. The (odd, odd) coarse grid consists of (odd, odd) fine grid points (green-colored); the $(odd, even)$ coarse grid consists of $(odd, even)$ fine grid points (black-colored); the $(even, odd)$ coarse grid consists of $(even, odd)$ fine grid points (blue-colored); the $(even, even)$ coarse grid is the standard coarse grid, which consists of $(even, even)$ fine grid points (red-colored). In addition, all boundary points are marked in red.

In order to compute sixth-order solutions for $(odd, even)$ fine grid points, we create an X-odd grid view composed of $(even, even)$ and $(odd, even)$ fine grid points, as in Fig. 2.4(a). It looks similar to a combination of the $(even, even)$ coarse grid and the $(odd, even)$ coarse grid from Fig. 2.3. The reason for calling "grid view" is that all the following computations can be conducted on the fine grid and there is no need to build the X-odd grid physically.

The X-odd grid view is a view of unequal mesh-size grid with mesh-sizes Δ and 2Δ in the x and y coordinate directions, respectively. In Figure 2.4(a), red-colored $(even, even)$ fine grid points and boundary points have sixth-order solutions, while black-colored $(odd, even)$ fine grid points have computed fourth-order solutions. The black points form vertical lines. Therefore, we can perform a tridiagonal solver in the y direction with $x = odd$ to solve all black-colored points line by line and make all $(odd, even)$ fine grid points obtain sixth-order solutions. Next, we will construct the tridiagonal systems from Eq. (2.9).

Assume that the grid points are ordered lexicographically, i.e., first from left to right along the x direction then from bottom to top along the y direction. The coefficient matrix of the FOC difference scheme with this ordering is a block tridiagonal matrix of block order $N_y/2$ [93], (the order of the coefficient matrix A is $N_x \times N_y/2$)

$$A = \text{diag}[A_1, A_0, A_1], \tag{2.13}$$

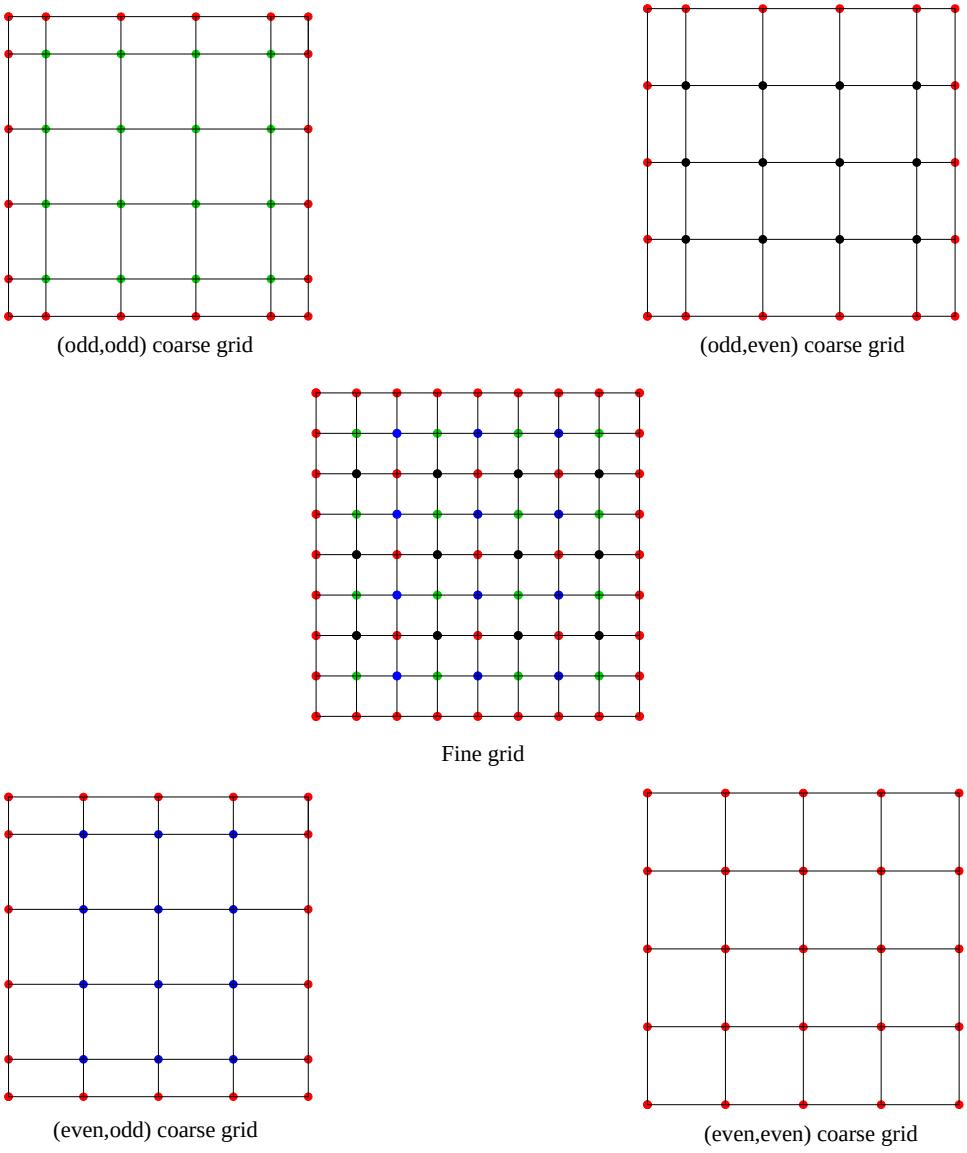
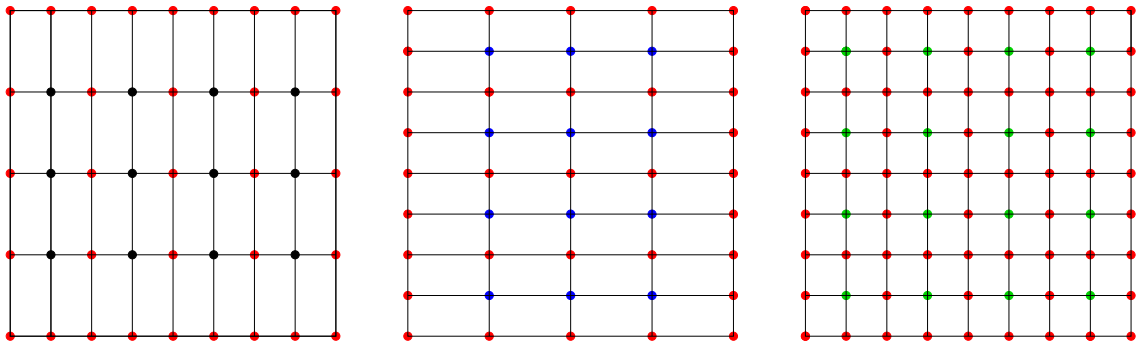


Figure 2.3: Illustration of the multiple coarse grids for 2D problem.



(a) X-odd grid view: (even,even) coarse grid + (odd,even) coarse grid.

(b) Y-odd grid view: (even,even) coarse grid + (even,odd) coarse grid.

(c) The fine grid after X-odd grid view and Y-odd grid view computation.

Figure 2.4: Illustration of the MCG updating strategy in 2D.

where

$$A_0 = \text{diag}[c, a, c], \qquad A_1 = \text{diag}[d, b, d] \tag{2.14}$$

are symmetric tridiagonal submatrices of order N_x. Here a, b, c, d are the coefficients defined in Eq. (2.9). They represent the submatrix of each grid line along the y direction. The grid points on the lines $i = 0, 2, 4, ..., N_x$ in the X-odd grid view have at least sixth-order solutions. Thus, sixth-order solutions at $(odd, even)$ fine grid points can be computed on the vertical lines with $i = 1, 3, 5, ..., N_x - 1$ by solving

$$A_0 u_i = F_i - A_1(u_{i-1} + u_{i+1}), \tag{2.15}$$

where u_i is part of the solution vector representing the grid points on the ith line, and F_i is the corresponding part of the right-hand side vector from Eq. (2.9).

Similarly, we create a Y-odd grid view composed of $(even, even)$ and $(even, odd)$ fine grid points, as in Figure 2.4(b). It looks similar to a combination of the $(even, even)$ coarse grid and the $(even, odd)$ coarse grid from Fig. 2.3. The Y-odd grid view is virtual too, and all calculations are on the fine grid.

The Y-odd grid view is a view of unequal mesh-size grid with mesh-sizes 2Δ and Δ in the x and y coordinate directions, respectively. In Figure 2.4(b), red-colored $(even, even)$ fine grid points and boundary points have sixth-order solutions, while

blue-colored $(even, odd)$ fine grid points have computed fourth-order solutions. The blue points form horizontal lines. Hence, we can similarly perform a tridiagonal solver in the x direction with $y = odd$ to solve all blue-colored points line by line and make them obtain sixth-order solutions. In the following, we discuss how to build the tridiagonal systems from Eq. (2.9).

Let us change the order of the grid points, i.e., first from bottom to top along the y direction then from left to right along the x direction. The coefficient matrix of the FOC difference scheme with this order is as a block tridiagonal matrix of block order $N_x/2$, (the order of the coefficient matrix B is $N_x/2 \times N_y$)

$$B = \text{diag}[B_1, B_0, B_1], \tag{2.16}$$

where

$$B_0 = \text{diag}[b, a, b], \qquad B_1 = \text{diag}[d, c, d] \tag{2.17}$$

are symmetric tridiagonal submatrices of order N_y. Here a, b, c, d are also the coefficients defined in Eq. (2.9). They represent the submatrix of each grid line along the x direction. The grid points on the lines $j = 0, 2, 4, ..., N_y$ in the Y-odd grid view have at least sixth-order solutions. Thus, sixth-order solutions at $(even, odd)$ fine grid points can be computed on the horizontal lines with $j = 1, 3, 5, ..., N_y - 1$ by

$$B_0 u_j = F_j - B_1(u_{j-1} + u_{j+1}), \tag{2.18}$$

where u_j is part of the solution vector representing the grid points on the jth line, and F_j is the corresponding part of the right-hand side vector from Eq. (2.9).

By now, the $(even, even)$, $(odd, even)$ and $(even, odd)$ fine grid points have sixth-order solutions. As for the $(even, even)$ fine grid points, their sixth-order solutions are interpolated from the standard coarse grid directly. As for the $(odd, even)$ and $(even, odd)$ fine grid points, their sixth-order solutions are solved from the X-odd and Y-odd grid views, respectively. Because the involved extrapolated sixth-order

solution is on the coarse grid, the computed sixth-order solution for fine grid points has truncation error $O((2\Delta)^6)$, not $O((\Delta)^6)$.

In Figure 2.4(c), the color red is used to denote the grid points with sixth-order solution. Only the (odd, odd) fine grid points in green still have fourth-order solutions. Since every green-colored point is immediately surrounded by red-colored points, some suitable interpolation can be used to compute sixth-order solutions for (odd, odd) fine grid points. We simply apply a one step operator based interpolation to update the solution of every (odd, odd) fine grid point by

$$\begin{aligned}\tilde{u}_{i,j} = -\frac{1}{a}[F_{i,j} - b(u_{i+1,j} + u_{i-1,j}) - c(u_{i,j+1} + u_{i,j-1}) \\ - d(u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1})],\end{aligned} \tag{2.19}$$

where a, b, c and d have the same definition as in the FOC scheme (2.9) and $F_{i,j}$ represents the right-hand side part of Eq. (2.9). Since all grid points in the right-hand side have approximate solutions with sixth-order accuracy, $\tilde{u}_{i,j}$ now has a sixth-order solution.

2.3.3 MSMG method with Richardson extrapolation and MCG updating strategy for the 2D Poisson equation

We use the MCG updating strategy to replace the iterative refinement procedure to accelerate the MSMG computation. Algorithm 2 gives the MSMG method with the MCG updating strategy and Richardson extrapolation technique for computing sixth-order solutions of the 2D Poisson equation. It is easy to use the following algorithm to compute sixth-order solutions for other 2D steady-state equations. The changes need to make are using an FOC scheme for the specific equation and generating appropriate tridiagonal systems and operator based interpolation from the FOC schemes with unequal mesh-size discretization.

The notation $u^{m\Delta}$ represents the computed fourth-order solution on the uniform grid with mesh-size $m\Delta$. $\tilde{u}^{m\Delta,n\Delta}$ denotes the computed sixth-order solution on the

grid with unequal mesh-sizes $m\Delta$ and $n\Delta$ in the x and y directions, respectively. $\tilde{u}^{m\Delta}$ denotes the computed sixth-order solution on the uniform grid with mesh-size $m\Delta$. $\Omega_{m\Delta}$ represents the computational space discretized by a uniform grid with mesh-size $m\Delta$.

Algorithm 2 Sixth-order solution computation for the 2D Poisson equation using MSMG method with MCG updating strategy and Richardson extrapolation

1. Use the uniform FOC scheme (2.10) and MSMG method to compute fourth-order solutions $u^{2\Delta} \in \Omega_{2\Delta}$ and $u^{\Delta} \in \Omega_{\Delta}$.
2. Compute sixth-order solutions for $(even, even)$ fine grid points on Ω_{Δ}.
For every inner grid point on $\Omega_{2\Delta}$, from $u^{2\Delta}_{i,j} \in \Omega_{2\Delta}$ and $u^{\Delta}_{2i,2j} \in \Omega_{\Delta}$, apply Richardson extrapolation using (2.12) to calculate $\tilde{u}^{2\Delta}_{i,j} \in \Omega_{2\Delta}$, then use direct interpolation to obtain $\tilde{u}^{\Delta}_{2i,2j} \in \Omega_{\Delta}$.
3. Compute sixth-order solutions for $(odd, even)$ fine grid points on Ω_{Δ}.
Use all $(even, even)$ fine grid points with $\tilde{u}^{\Delta}_{2i,2j} \in \Omega_{\Delta}$ and all $(odd, even)$ fine grid points with $u^{\Delta}_{2i+1,2j} \in \Omega_{\Delta}$ to form an X-odd grid view.
From the X-odd grid view, solve $N_x/2$ y direction tridiagonal systems using (2.15) on Ω_{Δ} to calculate $\tilde{u}^{\Delta,2\Delta}_{2i+1,2j} \in \Omega_{\Delta}$.
4. Compute sixth-order solutions for $(even, odd)$ fine grid points on Ω_{Δ}.
Use all $(even, even)$ fine grid points with $\tilde{u}^{\Delta}_{2i,2j} \in \Omega_{\Delta}$ and all $(even, odd)$ fine grid points with $u^{\Delta}_{2i,2j+1} \in \Omega_{\Delta}$ to form a Y-odd grid view.
From the Y-odd grid view, solve $N_y/2$ x direction tridiagonal systems using (2.18) on Ω_{Δ} to calculate $\tilde{u}^{2\Delta,\Delta}_{2i,2j+1} \in \Omega_{\Delta}$.
5. Compute sixth-order solutions for (odd, odd) fine grid points on Ω_{Δ}.
For every (odd, odd) fine grid point with $u^{\Delta}_{2i+1,2j+1} \in \Omega_{\Delta}$, perform one step operator based interpolation on Ω_{Δ} using Eq. (2.19) to obtain $\tilde{u}^{\Delta}_{2i+1,2j+1} \in \Omega_{\Delta}$.

Comparing the improved MSMG (MSMG-MCG) method with the existing MSMG (MSMG-Iter) method [79], the difference lies in the fine grid updating process, which aims to improve the solution accuracy of $(odd, even)$, $(even, odd)$ and (odd, odd) fine grid points. In the MSMG-MCG method, tridiagonal systems corresponding to X-odd and Y-odd grid views need to be generated and solved to get sixth-order solutions for $(odd, even)$ and $(even, odd)$ fine grid points. Then, a one step interpolation is applied to compute a sixth-order solution for every (odd, odd) fine grid point. However, in the MSMG-Iter method, the operator based interpolation is used to update three groups of grid points iteratively. In the following part, we analyze and compare the

computational cost of these two updating processes, respectively.

Assume $N_x = N_y = n$ on the finest grid and the total number of fine grid points $N = n * n$. In both methods, N/4 fine grid points achieve sixth-order solutions by injecting from the extrapolated coarse grid points. We only need to compare the cost of updating the remaining 3N/4 fine grid points by using the MCG updating strategy or the iterative refinement procedure. One arithmetic operation (i.e., addition, subtraction, multiplication, and division) between two floating points is counted as one unit of work. Our source code shows: generating a tridiagonal coefficient matrix needs 18 units of work; solving a tridiagonal system with n unknowns needs $10n - 21$ units of work; conducting one operator based interpolation needs 17 units of work. Tables 2.1 and 2.2 list the main costs of updating $(odd, even)$, $(even, odd)$ and (odd, odd) fine grid points in the MCG updating strategy and the iterative refinement procedure, respectively.

In Table 2.1, the total cost of the fine grid updating process in the MCG updating strategy is $\frac{37}{4}n^2 - 3n$. In Table 2.2, the total cost of the fine grid updating process in the iterative refinement procedure is $k \times (\frac{51}{4}n^2 - 17n)$, where k is the number of iterative refinement steps. If set $n > 3$, then $\frac{37}{4}n^2 - 3n < \frac{51}{4}n^2 - 17n$. This inequality shows that the computational cost of the MCG updating strategy is less than that of one iterative refinement step in the iterative refinement procedure when the number of intervals on the fine grid is larger than 3.

2.4 Extension to the 2D Convection-Diffusion Equation

When the MSMG-MCG method is applied to solve a 2D convection-diffusion equation, a 9-point FOC scheme with unequal mesh-size discretization for the 2D convection-diffusion equation is needed. We consider the 2D convection-diffusion equation of the form

$$u_{xx}(x,y) + u_{yy}(x,y) + p(x,y)u_x(x,y) + q(x,y)u_y(x,y) = f(x,y), \qquad (x,y) \in \Omega, \quad (2.20)$$

Table 2.1: Computational cost of the fine grid updating process with the MCG updating strategy.

Operation	**Cost**
compute (odd,even) fine grid points for line $i = 1, 3, ..., N_x - 1$, generate and solve the tridiagonal system	$\frac{n}{2} \times (18 + (10\frac{n}{2} - 21))$ $= \frac{5}{2}n^2 - \frac{3}{2}n$
compute (even,odd) fine grid points for line $j = 1, 3, ..., N_y - 1$, generate and solve the tridiagonal system	$\frac{n}{2} \times (18 + (10\frac{n}{2} - 21))$ $= \frac{5}{2}n^2 - \frac{3}{2}n$
compute (odd,odd) fine grid points for $u_{i,j}$ with $i = 1, 3, ..., N_x - 1$ and $j = 1, 3, ..., N_y - 1$, conduct one step operator based interpolation	$\frac{n}{2} \times \frac{n}{2} \times 17$ $=\frac{17}{4}n^2$
Total cost:	$\frac{37}{4}n^2 - 3n$

Table 2.2: Computational cost of the fine grid updating process with the iterative refinement procedure.

Operation	**Cost**
for $k = 1, 2, 3, ...$	
update (odd,odd) fine grid points for $u_{i,j}$ with $i = 1, 3, ..., N_x - 1$ and $j = 1, 3, ..., N_y - 1$, conduct operator based interpolation	$\frac{n}{2} \times \frac{n}{2} \times 17$ $=\frac{17}{4}n^2$
update (odd,even) fine grid points for $u_{i,j}$ with $i = 1, 3, ..., N_x - 1$ and $j = 2, 4, ..., N_y - 2$, conduct operator based interpolation	$\frac{n}{2} \times (\frac{n}{2} - 1) \times 17$ $=\frac{17}{4}n^2 - \frac{17}{2}n$
update (even,odd) fine grid points for $u_{i,j}$ with $i = 2, 4, ..., N_x - 2$ and $j = 1, 3, ..., N_y - 1$, conduct operator based interpolation	$(\frac{n}{2} - 1) \times \frac{n}{2} \times 17$ $=\frac{17}{4}n^2 - \frac{17}{2}n$
check the L^2-norm. If converged, exit the iteration and stop.	
Total cost: *(k is the number of iterative refinements)*	$k \times (\frac{51}{4}n^2 - 17n)$

where Ω is a rectangular domain with appropriate boundary conditions defined on $\partial\Omega$. We assume that the coefficients $p(x, y)$ and $q(x, y)$ are sufficiently smooth on Ω.

u_0 is used to denote the approximate value of $u(x, y)$ at a grid point (x, y). The approximate values of the eight immediate neighboring points are denoted by u_i, $i = 1, 2, ..., 8$. The 9-point compact grid points are labeled as

$$\begin{pmatrix} u_6 & u_2 & u_5 \\ u_3 & u_0 & u_1 \\ u_7 & u_4 & u_8 \end{pmatrix}.$$

We use p_i, q_i and f_i $(i = 0, 1, .., 4)$ to denote the function values at the corresponding grid points.

By using the symbolic computation package from Maple, the 9-point general FOC scheme with unequal mesh-sizes Δx and Δy for Eq. (2.20) at the mesh point (x, y) is obtained as [95]

$$\sum_{j=0}^{8} \alpha_j u_j = F. \tag{2.21}$$

In a special case of $\Delta x = \Delta y$, the scheme is the same as Gupta's 9-point FOC scheme [32]. In general case, if we denote the mesh aspect ratio $\lambda = \Delta y/\Delta x$, the coefficients α_j and the right-hand side F are given in Appendix A.

To compute a sixth-order solution for Eq. (2.20), we first compute fourth-order solutions on Ω_Δ and $\Omega_{2\Delta}$ by using Eq. (2.21). Next, Richardson extrapolation and direct interpolation are used to obtain sixth-order solutions of $(even, even)$ fine grid points. Then, by setting different λs, Eq. (2.21) can be used to build tridiagonal systems from X-odd and Y-odd grid views for computing sixth-order solutions for $(odd, even)$ and $(even, odd)$ fine grid points. Finally, a one step operator based interpolation is generated from Eq. (2.21) for updating (odd, odd) fine grid points.

2.5 Numerical Results

In this section, we compare the MSMG-MCG method with the MSMG-Iter method. The codes were written in Fortran 77 programming language and run on one login

node of Lipscomb HPC Cluster at the University of Kentucky. The node has Dual Intel E5-2670 8 Core (totally 16 cores) with 2.6GHz and 128 GB RAM.

We used the standard V(1,1)-Cycle in the MSMG computation. The initial guess for the V-Cycle on $\Omega_{4\Delta}$ was the zero vector. The multigrid V-Cycles on $\Omega_{2\Delta}$ and Ω_{Δ} stopped when the L^2-norm of the difference of the successive solutions was reduced by a factor of 10^{10}. The iterative refinement procedure in the MSMG-Iter method was terminated when the L^2-norm of the correction vector of the approximate solution was less than 10^{-10}. The upper limit for the number of iterations in the MSMG-Iter method was set as 10000. The errors reported were the maximum absolute errors over Ω_{Δ}.

We also computed an estimated order of accuracy for every computing strategy with different mesh-sizes. Consider two mesh-sizes Δ^H and Δ^h on Ω^H and Ω^h, respectively. The maximum absolute errors of these two grids are denoted as $Error^H$ and $Error^h$. If we set the order of accuracy as m, then we have the following form

$$\frac{(\Delta^H)^m}{(\Delta^h)^m} = \frac{Error^H}{Error^h}.$$

So, the order of accuracy m can be computed as

$$m = \frac{\log \frac{Error^H}{Error^h}}{\log \frac{\Delta^H}{\Delta^h}}. \tag{2.22}$$

The order of accuracy is formally defined when the mesh-size approaches zero. Therefore, when the mesh-size is relatively large, the discretization scheme may not achieve its formal order of accuracy.

2.5.1 Test problem 1

We considered a 2D Poisson equation as follows

$$u_{xx}(x,y) + u_{yy}(x,y) = -2\pi^2 \sin(\pi x)\cos(\pi y), \qquad (x,y) \in \Omega = [0,4] \times [0,4], \tag{2.23}$$

which has the Dirichlet boundary condition. The analytical solution of Eq. (2.23) is

$$u(x,y) = \sin(\pi x)\cos(\pi y).$$

In the following, we use $N_x = N_y = n$. Table 2.3 gives the comparison between the two strategies. The number of iterations contains three parts. The first two parts enclosed with parentheses are the number of V-Cycles on $\Omega_{2\Delta}$ and the number of V-Cycles on Ω_{Δ}. For the MSMG-Iter method, the third part is the number of iterations in the iterative refinement procedure. We recorded two different kinds of CPU time. "Total CPU" is the total CPU time for solving the problem; "Accuracy-Improve CPU" is the CPU time for obtaining a fine grid sixth-order solution from an extrapolated coarse grid sixth-order solution by using iterative refinement procedure (MSMG-Iter) or MCG updating strategy (MSMG-MCG). We find that the MSMG-MCG method and the MSMG-Iter method were able to compute comparable sixth-order solutions, but the MSMG-MCG method took less CPU time than the MSMG-iter method. This is because the proposed MCG updating strategy eliminates the iterative refinement procedure. The column of "Accuracy-Improve CPU" shows the CPU cost that the MCG updating strategy can save compared to the iterative refinement procedure. From the "error" column, we find that the MSMG-iter method is slightly more accurate than the MSMG-MCG method. One possible reason for this is that the MSMG-iter method uses the operator based interpolation iteratively, which is generating from the FOC scheme for uniform grids with mesh-size Δ to obtain sixth-order solutions for the remaining fine grid points, while the MSMG-MCG method uses the FOC scheme with unequal mesh-sizes Δ and 2Δ to compute sixth-order solutions for those fine grid points. The former FOC scheme theoretically has smaller truncation error than the latter one.

In Figure 2.5, we note that when the mesh became finer, the CPU time for the iterative refinement procedure in the MSMG-Iter method increased very fast. However, for the MCG updating strategy in the MSMG-MCG method, its CPU time showed slow growth. The numerical comparison on CPU costs between two fine grid updating strategies were consistent with the analysis in Tables 2.1 and 2.2. It is visible that

Table 2.3: Test Problem 1: Comparison of the CPU costs and solution accuracy with different mesh-sizes.

n	Strategy	# it	Total CPU(s)	Accuracy -Improve CPU(s)	error	order
64	MSMG-Iter	(8,8),24	0.023	0.004	2.060e-7	-
	MSMG-MCG	(8,8),-	0.019	0.000	2.836e-7	-
128	MSMG-Iter	(8,9),23	0.096	0.006	3.356e-9	5.94
	MSMG-MCG	(8,9),-	0.092	0.000	4.416e-9	6.00
256	MSMG-Iter	(9,9),21	0.423	0.022	4.642e-11	6.18
	MSMG-MCG	(9,9),-	0.401	0.004	8.262e-11	5.74
512	MSMG-Iter	(9,9),19	1.283	0.042	8.284e-13	5.81
	MSMG-MCG	(9,9),-	1.143	0.006	1.089e-12	6.24
1024	MSMG-Iter	(9,9),18	4.037	0.199	3.211e-14	4.69
	MSMG-MCG	(9,9),0	3.839	0.026	3.577e-14	4.93

the MCG updating strategy has better scalability and computational efficiency than the iterative updating strategy.

2.5.2 Test problem 2

We chose a 2D convection-diffusion equation with variable coefficients as follows

$$u_{xx}(x,y) + u_{yy}(x,y) + p(x,y)u_x(x,y) + q(x,y)u_y(x,y) = f(x,y),$$
$$(x,y) \in \Omega = [0,1] \times [0,1], \tag{2.24}$$

where

$$\begin{cases} u(x,y) = x^2y^2(1-x)(1-y) \\ p(x,y) = Px(1-y) \\ q(x,y) = Py(1-x) \end{cases}.$$

For this test problem, a larger value of P means a larger Reynolds number (Re). The cell Reynolds number is defined as the ratio of the convection to diffusion in the form of

$$Re = \max(\sup_{(x,y)\in\Omega} |p(x,y)|, \sup_{(x,y)\in\Omega} |q(x,y)|)h/2. \tag{2.25}$$

Convection-diffusion equations like Eq. (2.24) become increasingly difficult to solve by iterative methods as Re increases [86]. Since the exact order of solution accuracy from the FOC scheme is related to the Reynolds number [81], we computed

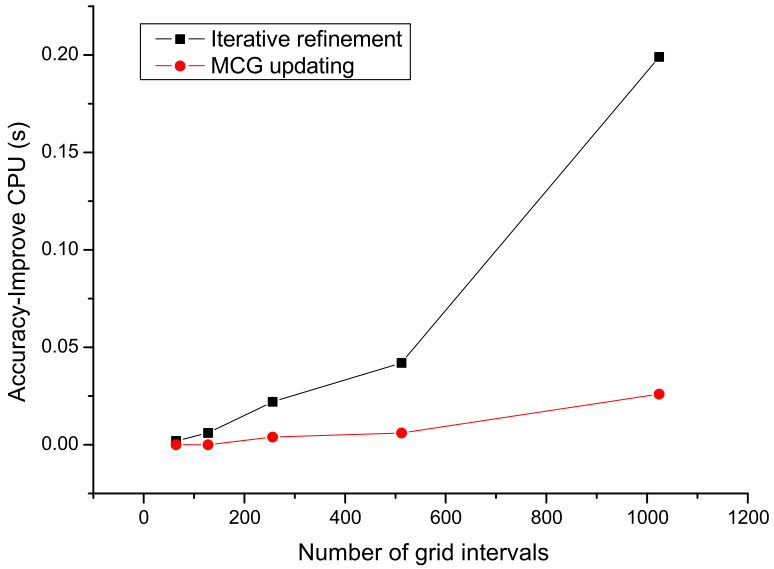


Figure 2.5: Comparison of the Accuracy-Improve CPU time and the number of grid intervals between the iterative refinement strategy and the MCG updating strategy for solving Problem 1. Each symbol with increasing CPU cost corresponds to an increasing fine grid: 64, 128, 256, 512 and 1024 intervals.

the exact order of accuracy and used the general Richardson extrapolation formula (2.11).

We tested for a difficult case with $P = 10^5$ and used the residual scaling technique [85] to accelerate V-Cycles. Set $N_x = N_y = n$ in the following tables and figures.

Residual scaling techniques have been proposed and applied by Zhang in [85, 87] and is a kind of acceleration technique which is designed to accelerate standard multigrid methods in different situations. There are two categories of acceleration techniques. The first one contains pre-acceleration techniques, which accelerate the multigrid process before the coarse grid procedure. The other one consists of the post-acceleration techniques, which accelerate the multigrid process after the coarse grid procedure. One type of pre-acceleration techniques is the pre-scaling technique which scales the residual vector by a pre-determined *residual scaling factor* before it is projected to the coarse grid. Meanwhile, one type of post-acceleration techniques is

Table 2.4: Test Problem 2: Comparison of the CPU costs and solution accuracy with different mesh-sizes for $P = 10^5$.

n	Strategy	# it	Total CPU(s)	Accuracy -Improve CPU(s)	error	order
64	MSMG-Iter	(167,443),1745	0.839	0.041	1.931e-4	-
	MSMG-MCG	(167,443),-	0.724	0.001	2.014e-4	-
128	MSMG-Iter	(443,588),2744	2.492	0.263	6.034e-5	1.68
	MSMG-MCG	(443,588),-	2.249	0.002	5.962e-5	1.76
256	MSMG-Iter	(588,367),2176	7.665	0.829	3.663e-6	4.04
	MSMG-MCG	(588,367),-	6.772	0.011	3.877e-6	3.94
512	MSMG-Iter	(367,253),1046	25.833	2.318	1.219e-7	4.91
	MSMG-MCG	(367,253),-	23.316	0.043	1.568e-7	4.63
1024	MSMG-Iter	(253,236),522	95.073	6.029	3.155e-9	5.27
	MSMG-MCG	(253,236),-	89.982	0.179	5.103e-9	4.94

the post-scaling technique which scales the correction term by a scaling factor chosen to minimize the error in energy norm. Zhang unified the pre-scaling technique and the post-scaling technique as the residual scaling techniques. He proved that the pre-scaling and post-scaling techniques are mathematically equivalent if and only if their scaling factors are equal. He also showed that the pre-scaling technique is cheaper and has a wider application than the post-scaling technique. Therefore, we chose the pre-scaling technique to apply in the present algorithm. The details of how to choose an optimal *residual scaling factor* with high Reynolds number can be found in [92]. In our experiments, we tested several scaling factors and only listed the best numerical results.

The numerical data with comparison are shown in Table 2.4. Similarly, we had three parts in the number of iterations and recorded two types of CPU cost. From the third part of the number of iterations, we can see that the iterative refinement procedure took hundreds, even thousands of iterations to update grid points on the finest grid for improving the solution accuracy. The "Accuracy-Improve CPU" column illustrates that the MCG updating strategy has higher computational efficiency than the iterative refinement procedure. At the same time, the "Total CPU" column shows that the proposed method indeed accelerates the MSMG computation. In addition,

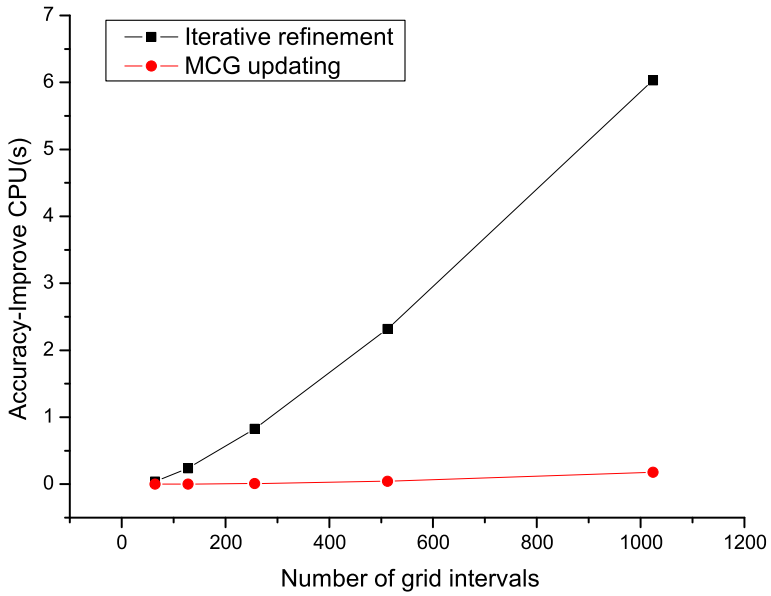


Figure 2.6: Comparison of the Accuracy-Improve CPU time and the number of grid intervals between the iterative refinement strategy and the MCG updating strategy for solving Problem 2 ($P = 10^5$). Each symbol with increasing CPU cost corresponds to an increasing fine grid: 64, 128, 256, 512 and 1024 intervals.

the MSMG-MCG method and the MSMG-Iter method were able to obtain solutions of comparable accuracy, but the order of solutions was reduced and did not reach six. The reason for this is that the solutions from the FOC scheme cannot reach the fourth-order accuracy when Re is very large [81].

Figure 2.6 compares the scalability of the MCG updating strategy and the iterative refinement procedure. When the number of grid intervals increased, the CPU cost on the iterative refinement procedure increased quickly, while the CPU cost on the MCG updating strategy increased much more slowly. When $n = 1024$, the iterative refinement procedure required 6.029 seconds to update fine grid points for improving their solutions, while the MCG updating strategy only took 0.179 seconds on the improvement.

Table 2.5 contains the numerical results with various P values on a grid with a fixed mesh-size $h = 1/256$. Using Eq. (2.25), we computed the corresponding Re

Table 2.5: Test Problem 2: Comparison of the CPU costs and solution accuracy with different P values for $n = 256$.

		$n = 256$				
P	Re	Strategy	# it	Total CPU(s)	Accuracy -Improve CPU(s)	error
1	1.953125e-3	MSMG-Iter	(9,9),8	0.432	0.003	2.181e-13
		MSMG-MCG	(9,9),-	0.467	0.026	1.706e-13
10	1.953125e-2	MSMG-Iter	(9,9),12	0.447	0.012	3.266e-13
		MSMG-MCG	(9,9),-	0.464	0.028	6.720e-13
10^2	1.953125e-1	MSMG-Iter	(9,8),15	0.415	0.014	1.090e-11
		MSMG-MCG	(9,8),-	0.432	0.027	4.600e-11
10^3	1.953125e0	MSMG-Iter	(15,15),18	0.676	0.017	9.779e-10
		MSMG-MCG	(15,15),-	0.616	0.010	4.293e-9
10^4	1.953125e1	MSMG-Iter	(62,52),147	1.389	0.056	8.778e-7
		MSMG-MCG	(58,50),-	1.321	0.009	2.169e-7
10^5	1.953125e2	MSMG-Iter	(588,367),2176	7.601	0.821	3.663e-6
		MSMG-MCG	(588,367),-	6.671	0.010	2.877e-6
10^6	1.953125e3	MSMG-Iter	(131,219),7210	6.029	2.725	1.651e-5
		MSMG-MCG	(131,219),-	3.464	0.010	1.638e-5
10^7	1.953125e4	MSMG-Iter	(132,239),8235	6.975	3.110	1.139e-5
		MSMG-MCG	(132,239),0	3.548	0.011	1.135e-5
10^8	1.953125e5	MSMG-Iter	(132,239),8317	6.860	3.142	1.146e-5
		MSMG-MCG	(132,239),0	3.692	0.010	1.145e-5
10^9	1.953125e6	MSMG-Iter	(132,239),8325	6.815	3.146	1.149e-5
		MSMG-MCG	(132,239),0	3.572	0.010	1.141e-5
10^{10}	1.953125e7	MSMG-Iter	(132,239),8325	6.858	3.142	1.150e-5
		MSMG-MCG	(132,239),0	3.577	0.010	1.149e-5

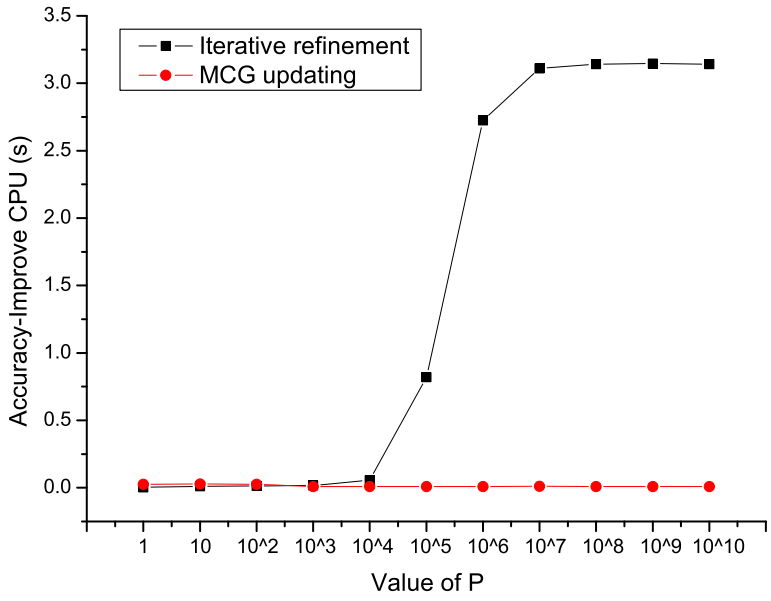


Figure 2.7: Comparison of the Accuracy-Improve CPU time between the iterative refinement strategy and the MCG updating strategy for solving Problem 2 ($n = 256$) with different P values.

values under different P values. We note that the magnitude of the Reynolds number affected the convergence rate of V-cycles and the solution accuracy simultaneously. The accuracy improvement of both methods was degraded when Re increased. For two fine grid updating strategies, when Re was small ($P < 10^3$), the number of iterations in the iterative refinement procedure was small and its CPU cost was almost less than the CPU cost of the MCG updating strategy. When Re increased, the number of iterations in the iterative refinement procedure increased very quickly, and its CPU cost was far more than that of the MCG updating strategy.

Figure 2.7 compares the CPU cost of two fine grid updating strategies for different values of P. We see that there was no evident change of the CPU cost of the MCG updating strategy when P increased. However, for the iterative updating strategy, there is a substantial change in the CPU cost at $10^4 \leq P \leq 10^6$ and the CPU cost remains high for $P > 10^6$. Compared to the iterative refinement procedure, the MCG

updating strategy is less sensitive to the change of $Re(P)$ and exhibits better stability.

2.6 Concluding Remarks

We developed a new MCG updating strategy to compute sixth-order solutions for fine grid points. The proposed strategy can replace the iterative refinement procedure in the existing MSMG method and thus accelerates the MSMG computation in computing high accuracy solutions for the 2D steady-state equations. Numerical results show that the MCG updating strategy is more efficient, scalable and stable than the iterative refinement procedure. The idea of using MCG updating strategy to directly compute more accurate fine grid solutions from the extrapolated coarse grid solutions can be extended to solve higher dimensional PDEs. We will discuss the details of its extension to 3D problems in the next chapter.

3 Sixth-Order Solution with Multiscale Multigrid Method and Multiple Coarse Grid Updating Strategy for 3D Steady-State Equations

3.1 Introduction

In this chapter, we extend our Richardson extrapolation-based sixth-order method with multiscale multigrid (MSMG) method and multiple coarse grid (MCG) updating strategy to compute for three dimensional (3D) convection-diffusion equation as

$$u_{xx} + u_{yy} + u_{zz} + p(x,y,z)u_x + q(x,y,z)u_y + r(x,y,z)u_z = f(x,y,z), \quad (x,y,z) \in \Omega, \tag{3.1}$$

where Ω is a continuous domain in a 3D space composed by a union of rectangular solids with suitable boundary conditions prescribed on $\partial\Omega$. Here the unknown function u, variable coefficient functions $p(x,y,z)$, $q(x,y,z)$, $r(x,y,z)$ and the forcing function $f(x,y,z)$ are assumed to be continuously differentiable and have required partial derivatives on Ω.

The numerical computing of Eq. (3.1) is very important in simulations and modeling applications, such as fluid dynamics, heat transfer and ocean modeling. Compared with lower dimensional problems, 3D problems face more serious computational challenges due to the requirements on the memory and CPU time to obtain solutions with desirable accuracy. In order to get accurate solutions with limited computational resources, high-order compact (HOC) difference methods have been proposed by many researchers and used to solve 3D PDEs [2, 35, 67, 88, 94]. These methods have been demonstrated to achieve high accuracy, numerical stability, and computational efficiency. In addition, they can handle boundary conditions effectively. Besides using HOC schemes, another way to save CPU time is to use parallel computing. Gupta and Zhang realized parallelization and vectorization by using four colors for 19-point scheme [36] and using two colors for 15-point scheme [91]. For other parallel computations of the 3D convection-diffusion equation, readers are referred to [49, 99, 100].

Up to now, fourth-order compact (FOC) schemes for solving the 3D convection-diffusion equation have been studied extensively in the literature. Recently, sixth-order compact schemes are found to be more computationally efficient than the FOC schemes. To obtain a computed solution of given accuracy, the sixth-order scheme uses less computational cost than the FOC scheme does. Ma and Ge [50] extended the explicit sixth-order method with Richardson extrapolation proposed by Sun and Zhang [69] to solve 3D convection-diffusion equations. The MSMG method for 2D problems proposed by Wang and Zhang [79, 81] has been extended to solve 3D convection-diffusion equations in [80]. However, in the current MSMG computation, updating fine grid points uses an iterative refinement procedure, which converges slowly and consumes a large amount of CPU time. For 2D problems, there are 1/4 fine grid points with solutions of sixth-order accuracy after applying Richardson extrapolation. Then, an iterative refinement procedure is used to upgrade the solution accuracy for the remaining 3/4 fine grid points. For 3D problems, only 1/8 fine grid points could reach sixth-order solutions directly from the Richardson extrapolation procedure. Next, the iterative procedure is executed to improve solution accuracy for the other 7/8 fine grid points. In addition, the number of grid points for 3D problems is much more than that for 2D problems. It is evident that the computational cost of the iterative refinement procedure is considerable for high dimensional problems. In Chapter 2, we presented an MCG updating strategy to compute sixth-order solutions for all fine grid points in 2D. In this Chapter, we extend the MCG updating strategy to 3D and thus to improve the MSMG computation for the 3D convection-diffusion equation. Analogously, the MCG updating strategy eliminates the iterative refinement procedure on the fine grid as well as decreases the coupling among grid points in the updating process. In other words, the new strategy not only reduces the CPU cost but also provides a convenient way for parallelization. Therefore, the proposed MSMG method with the MCG updating strategy is able to reach a higher computational efficiency

compared to the current MSMG method with the iterative refinement procedure.

3.2 FOC Scheme With Unequal Mesh-Size Discretization for the 3D Convection-Diffusion Equation

The Richardson extrapolation-based sixth-order method is based on fourth-order discretization schemes. There are two situations that FOC schemes are needed. The first one is to provide fourth-order solutions on two scale uniform grids for Richardson extrapolation. We could use the FOC scheme from Zhang's paper [88] like Wang did in [80]. The second one is during the implementation process of our new fine grid updating strategy. Zhang's FOC scheme is not appropriate for this situation since the grids involved in the proposed updating strategy are with unequal mesh-sizes in each coordinate direction. Therefore, we need an FOC scheme with unequal mesh-size discretization for solving the 3D convection-diffusion equation in this section.

Assume that the discretization is carried out on a 3D grid with mesh-sizes Δx, Δy and Δz in the x, y and z coordinate directions, respectively. We use u_0 to denote the value of $u(x, y, z)$ at an internal mesh point (i, j, k). The approximate values of its immediate 18 neighboring points are denoted by u_l, $l = 1, 2, ..., 18$, as in Fig. 3.1. The 8 white colored corner points in Fig. 3.1 are not used in the finite difference scheme. The discrete values of p_l, q_l, r_l and f_l for $l = 0, 1, ..., 6$ are defined similarly.

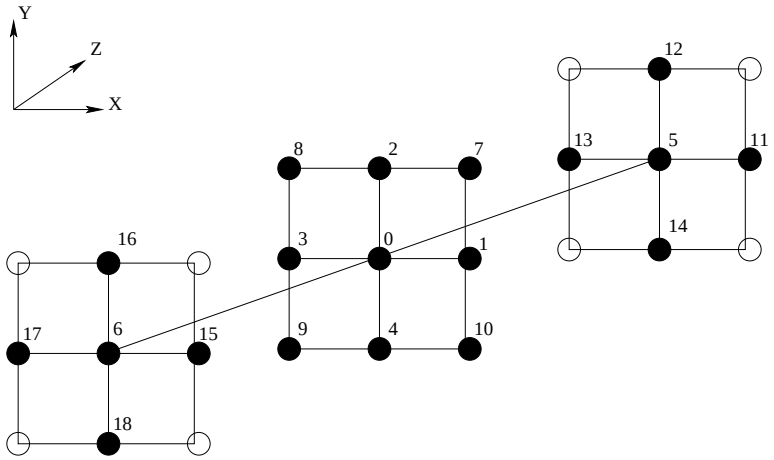


Figure 3.1: Labeling of the 3D grid points in a cuboid.

The approach we take here was advocated by Spotz and Carey [66, 65, 67] and used by Zhang and Ge [94] in developing an HOC scheme for the 3D convection-diffusion equation on uniform grids with a symbolic computation procedure. Since the derivation process is a complex substitution and term collection process, we choose to use the symbolic computation package from Maple.

Suppose the solution u has continuous partial derivatives of sufficient orders. By using Taylor series, its first and second-order partial derivatives with respect to x can be approximated by

$$\frac{\partial u}{\partial x} = \delta_x u - \frac{\Delta x^2}{6}\frac{\partial^3 u}{\partial x^3} - \frac{\Delta x^4}{120}\frac{\partial^5 u}{\partial x^5} + O(\Delta x^6), \tag{3.2}$$

$$\frac{\partial^2 u}{\partial x^2} = \delta_x^2 u - \frac{\Delta x^2}{12}\frac{\partial^4 u}{\partial x^4} - \frac{\Delta x^4}{360}\frac{\partial^6 u}{\partial x^6} + O(\Delta x^6), \tag{3.3}$$

where δ_x and δ_x^2 are the first and second-order central difference operators with respect to x. The first and second-order partial derivatives of u with respect to y and z can be approximated to $O(\Delta y^6)$ and $O(\Delta z^6)$ order analogously.

In order to derive a compact scheme up to the fourth-order accuracy, we use the approximation formulas (3.2) and (3.3) for the first and second-order partial derivatives of u with respect to x and their counterparts with respect to y and z to substitute the first and second-order partial derivatives of u in Eq. (3.1) and drop the $O(\Delta^4)$ and higher-order items. The substitution yields

$$\begin{aligned}&(\delta_x^2 u + \delta_y^2 u + \delta_z^2 u + p\delta_x u + q\delta_y u + r\delta_z u - f)\\ &- \frac{\Delta x^2}{6}\left(\frac{\partial^4 u}{2\partial x^4} + p\frac{\partial^3 u}{\partial x^3}\right) - \frac{\Delta y^2}{6}\left(\frac{\partial^4 u}{2\partial y^4} + q\frac{\partial^3 u}{\partial y^3}\right) - \frac{\Delta z^2}{6}\left(\frac{\partial^4 u}{2\partial z^4} + r\frac{\partial^3 u}{\partial z^3}\right) + O(\Delta^4) = 0,\end{aligned} \tag{3.4}$$

where p, q, r and f are short for $p(x,y,z)$, $q(x,y,z)$, $r(x,y,z)$ and $f(x,y,z)$, and $O(\Delta^4)$ denotes the truncated terms in the order of $O(\Delta x^4 + \Delta y^4 + \Delta z^4)$. Since the $O(\Delta^2)$ terms include Δx^2, Δy^2 and Δz^2, if the third and fourth-order partial derivatives of u with respect to x, y and z can be approximated to reach the second-

order accuracy, we can have the fourth-order accuracy for the whole approximation scheme.

We differentiate the original partial differential equation to obtain the approximations for the third and fourth-order partial derivatives of u with respect to x, y and z. To illustrate the idea, we differentiate Eq. (3.1) with respect to x to obtain

$$\frac{\partial^3 u}{\partial x^3} = \frac{\partial f}{\partial x} - \frac{\partial^3 u}{\partial x \partial y^2} - \frac{\partial^3 u}{\partial x \partial z^2} - \frac{\partial p}{\partial x}\frac{\partial u}{\partial x} - p\frac{\partial^2 u}{\partial x^2} - \frac{\partial q}{\partial x}\frac{\partial u}{\partial y} - q\frac{\partial^2 u}{\partial x \partial y} - \frac{\partial r}{\partial x}\frac{\partial u}{\partial z} - r\frac{\partial^2 u}{\partial x \partial z} \quad (3.5)$$

for the third order partial derivative of u with respect to x, and repeat the process to obtain

$$\begin{aligned}\frac{\partial^4 u}{\partial x^4} = {} & \frac{\partial^2 f}{\partial x^2} - \frac{\partial^4 u}{\partial x^2 \partial y^2} - \frac{\partial^4 u}{\partial x^2 \partial z^2} - \frac{\partial^2 p}{\partial x^2}\frac{\partial u}{\partial x} - 2\frac{\partial p}{\partial x}\frac{\partial^2 u}{\partial x^2} - p\frac{\partial^3 u}{\partial x^3} \\ & - \frac{\partial^2 q}{\partial x^2}\frac{\partial u}{\partial y} - 2\frac{\partial q}{\partial x}\frac{\partial^2 u}{\partial x \partial y} - q\frac{\partial^3 u}{\partial x^2 \partial y} - \frac{\partial^2 r}{\partial x^2}\frac{\partial u}{\partial z} - 2\frac{\partial r}{\partial x}\frac{\partial u}{\partial x \partial z} - r\frac{\partial^3 u}{\partial x^2 \partial z} \end{aligned} \quad (3.6)$$

for the fourth-order partial derivative of u with respect to x. The third and fourth-order partial derivatives of u with respect to y and z can be obtained analogously.

Then, we examine whether or not the approximations of the third and fourth-order partial derivatives of u can reach the second-order accuracy. After using the approximation of $\frac{\partial^3 u}{\partial x^3}$ in Eq. (3.5) to substitute the $\frac{\partial^3 u}{\partial x^3}$ term in the approximation of $\frac{\partial^4 u}{\partial x^4}$ in Eq. (3.6) (similar substitutions in the approximations of $\frac{\partial^4 u}{\partial y^4}$ and $\frac{\partial^4 u}{\partial z^4}$), all terms in the approximations of the third and fourth-order partial derivatives of u are no higher than second-order derivatives of u with respect to any variables. For the first and second-order partial derivatives of the unknown function u and other known functions p, q, r, and f, they can be approximated to the second-order accuracy by applying the first and second-order central difference operators, which uses 7 minimum grid points centered at 0, i.e., the grid points 0,1,2,3,4,5 and 6. For the cross derivatives, we can use the grid points outside the 7 minimum grid points

to approximate. For example, the approximations of $\frac{\partial^3 u}{\partial x \partial y^2}$ and $\frac{\partial^4 u}{\partial x^2 \partial y^2}$ as

$$\begin{aligned}
\frac{\partial^3 u}{\partial x \partial y^2} \approx \delta_x \delta_y^2 u &= \frac{1}{\Delta y^2}(\delta_x u_2 - 2\delta_x u_0 + \delta_x u_4) \\
&= \frac{1}{2\Delta x \Delta y^2}[u_7 - u_8 - 2(u_1 - u_3) + u_{10} - u_9] \\
\frac{\partial^4 u}{\partial x^2 \partial y^2} \approx \delta_x^2 \delta_y^2 u &= \frac{1}{\Delta y^2}(\delta_x^2 u_2 - 2\delta_x^2 u_0 + \delta_x^2 u_4) \\
&= \frac{1}{\Delta x^2 \Delta y^2}[u_7 - 2u_2 + u_8 - 2(u_1 - 2u_0 + u_3) + u_9 - 2u_4 + u_{10}]
\end{aligned}$$

have $O(\Delta^2)$ accuracy. Here $O(\Delta^2)$ absorbs $O(\Delta x^2)$, $O(\Delta y^2)$ and $O(\Delta x \cdot \Delta y)$. All other cross derivatives can be approximated analogously, which results in a 19-point FOC difference scheme as

$$\sum_{l=0}^{18} \alpha_l u_l = F. \tag{3.7}$$

In a special case with $\Delta x = \Delta y = \Delta z$, the scheme is the same as Zhang's explicit FOC scheme [88]. In general cases, if we denote the mesh aspect ratio $\lambda_1 = \Delta y/\Delta x$ and $\lambda_2 = \Delta z/\Delta x$, the coefficients α_l and the right hand side F are given in Appendix B.

3.3 Richardson Extrapolation-based Sixth-Order Solution with MSMG Method and MCG Updating Strategy for the 3D Convection-Diffusion Equation

3.3.1 Improving solution accuracy by Richardson extrapolation in 3D

With Eq. (3.7), the fourth-order accurate solutions $u_{i,j,k}^{\Delta}$ and $u_{i,j,k}^{2\Delta}$ can be computed by the MSMG method on Ω_Δ and $\Omega_{2\Delta}$, respectively. Then we apply the Richardson extrapolation technique to compute a higher-order solution $\tilde{u}_{i,j,k}^{2\Delta}$ on $\Omega_{2\Delta}$. The Richardson extrapolation has the formula as [6]

$$\tilde{u}_{i,j,k}^{2\Delta} = \frac{(2^p u_{2i,2j,2k}^{\Delta} - u_{i,j,k}^{2\Delta})}{2^p - 1}, \tag{3.8}$$

where p is the order of accuracy before the extrapolation, and the order of accuracy will be upgraded to $p + 2$ after the extrapolation. When the convection-diffusion equation is diffusion-dominated, we could assume $p = 4$ if the solution values for

the extrapolation are computed by the FOC scheme. Therefore, we can get a sixth-order solution on the coarse grid $\Omega_{2\Delta}$ by using Eq. (3.8). By using the direct interpolation, the sixth-order coarse grid solution $\tilde{u}^{2\Delta}_{i,j,k}$ are injected into corresponding $(even, even, even)$ fine grid points.

3.3.2 MCG updating strategy for 3D problems

The extrapolated sixth-order solution is on the coarse grid, not on the fine grid. The coarse grid sixth-order solution can be directly interpolated into the fine grid, which makes partial fine grid points reach sixth-order solutions. The problem is how to improve the solution of the remaining fine grid points to reach the sixth-order accuracy. Since the updating strategy deals with the fine grid points by groups, the fine grid points are divided into eight different groups by their *odd* or *even* index in the x, y and z coordinate directions, as in Fig. 3.2.

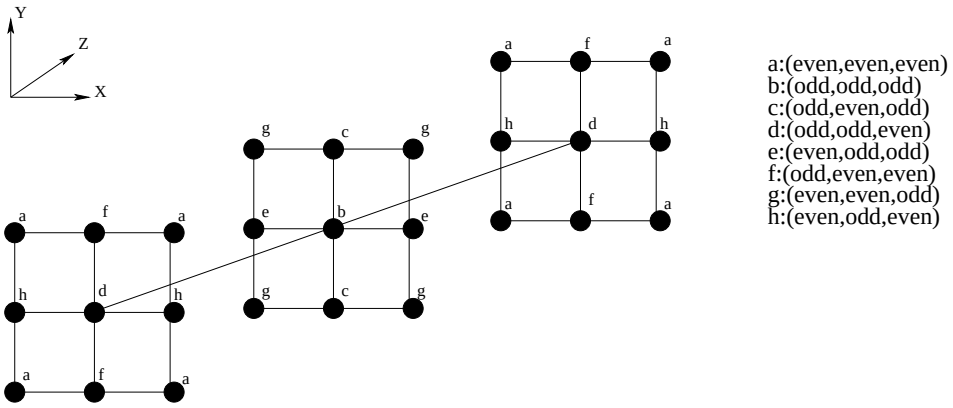


Figure 3.2: Group information of 3D grid points in a cuboid.

From Section 1.4, we have known that for 3D problems, a fine grid can be coarsened into eight coarse grids. Each coarse grid is composed by one group of fine grid points in Fig. 3.2. For instance, the $(even, even, even)$ coarse grid (also called the standard coarse grid) is built up by grid points in group a, which are the $(even, even, even)$ fine grid points. Unlike using an iterative refinement procedure to update the solution of the remaining fine grid points, the main idea of the MCG updating strategy is to

directly compute sixth-order solutions for fine grid points group by group by using different coarse grid views from various combinations of the standard coarse grid and other coarse grids.

Before we start to illustrate our ideas in details, there are following three points to be underlined. First, six coarse grid views will be constructed for computing sixth-order solutions for six groups' fine grid points, respectively. These groups exclude groups a and b for that the former one has reached the sixth-order solution after the direct interpolation from the extrapolated coarse grid solutions and the latter one will get the sixth-order solution by using an operator based interpolation. Second, the coarse grid views are virtual. In other words, the grid views to be described look like the corresponding coarse grids, but they do not physically exist. All computations are conducted on the fine grid. Third, the updating process has three stages. At the beginning, fine grid points in groups f, g and h are updated by some 2D solver. Then, fine grid points in groups c, d and e are updated by some 1D solver. At last, fine grid points in group b are updated by a one step operator based interpolation.

Update fine grid points of groups f, g and h

In order to update the fine grid points of group f, we create an X-odd grid view composed of fine grid points from groups a and f, which looks like a combination of the $(even, even, even)$ coarse grid and the $(odd, even, even)$ coarse grid, as in Fig. 3.3.

Notice that the X-odd grid view is a view of unequal mesh-size grid with mesh-sizes Δ, 2Δ and 2Δ in the x, y and z coordinate directions, respectively. In Fig. 3.3, the red-colored grid points, which are from group a, have sixth-order solutions, while the black-colored grid points from group f have fourth-order solutions. If we visit the (y, z)-plane through the x direction for $x = odd$, we can compute sixth-order solutions of all black-colored grid points plane by plane. There are $N_x/2$ 2D sub-

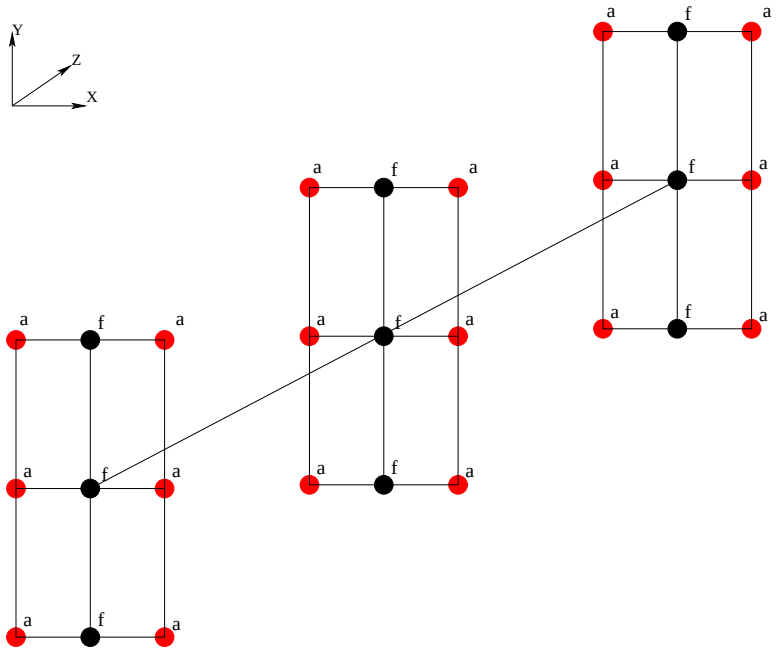


Figure 3.3: X-odd grid view: fine grid points from groups a and f.

problems, where N_x is the number of grid intervals along the x direction. The 9-point computational stencil of the (y, z)-plane 2D sub-problem is generated from Eq. (3.7) as

$$A_{f0}\tilde{u}_{f0} + A_{f1}\tilde{u}_{f1} + A_{f2}\tilde{u}_{f2} + A_{f3}\tilde{u}_{f3} + A_{f4}\tilde{u}_{f4} + A_{f5}\tilde{u}_{f5} + A_{f6}\tilde{u}_{f6} + A_{f7}\tilde{u}_{f7} + A_{f8}\tilde{u}_{f8} \tag{3.9}$$

$$= F - \alpha_1 u_1 - \alpha_3 u_3 - \alpha_7 u_7 - \alpha_8 u_8 - \alpha_9 u_9 - \alpha_{10} u_{10} - \alpha_{11} u_{11}$$

$$- \alpha_{13} u_{13} - \alpha_{15} u_{15} - \alpha_{17} u_{17},$$

where the coefficients A_{fl} and the 2D solutions $\tilde{u}_{fl}(l = 0, 1, ..., 8)$ of grid points from group f are set in the group f part of Table 3.1.

It is essentially a 9-point compact scheme with 2Δ mesh-size. Since the coefficients (A_{fl} and α_{fl}) and F in Eq. (3.9) are from the unequal mesh-size coarse grid view X-odd, they are calculated by the FOC scheme with unequal mesh-size we developed in Section 3.2. For the u values in the right hand side of Eq. (3.9), we use the computed solutions of grid points in group a which have reached the sixth-order accuracy.

Next, we construct a Z-odd grid view composed of fine grid points from groups

a and g, which looks like a combination of the $(even, even, even)$ coarse grid and the $(even, even, odd)$ coarse grid, as in Fig. 3.4. This view is used to compute a sixth-order solution for the fine grid points of group g.

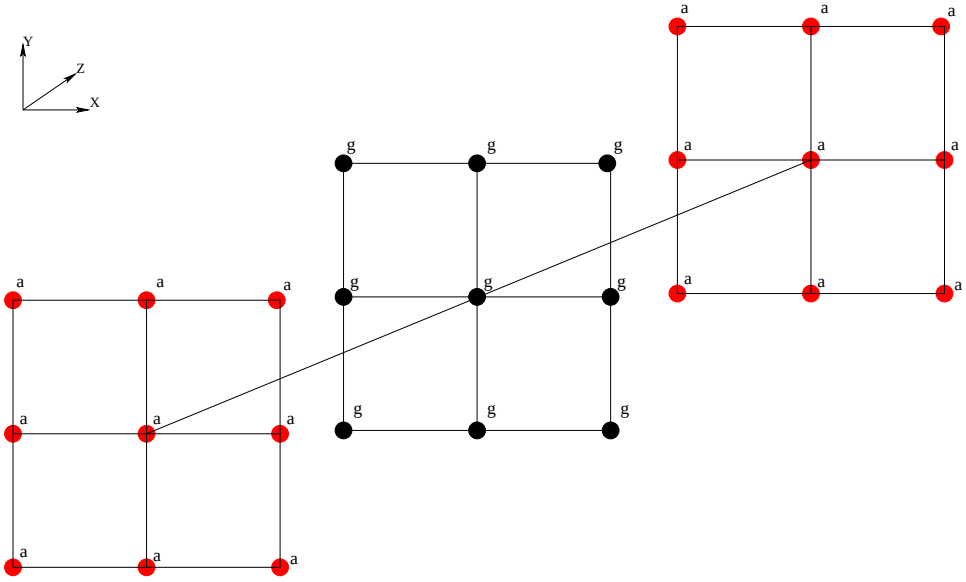


Figure 3.4: Z-odd grid view: fine grid points from groups a and g.

The Z-odd grid view is a view of unequal mesh-size grid with mesh-sizes 2Δ, 2Δ and Δ in the x, y and z coordinate directions, respectively. In Fig. 3.4, we mark the grid points from group a as red, which have sixth-order solutions. At the same time, we mark the grid points from group g as black, which have fourth-order solutions. If we visit the (x, y)-plane through the z direction for $z = odd$, all black-colored grid points can be solved plane by plane. There are totally $N_z/2$ 2D sub-problems, where N_z is the number of grid intervals along the z direction. For the (x, y)-plane 2D sub-problem, its 9-point computational stencil is generated from Eq. (3.7) as

$$A_{g0}\tilde{u}_{g0} + A_{g1}\tilde{u}_{g1} + A_{g2}\tilde{u}_{g2} + A_{g3}\tilde{u}_{g3} + A_{g4}\tilde{u}_{g4} + A_{g5}\tilde{u}_{g5} + A_{g6}\tilde{u}_{g6} + A_{g7}\tilde{u}_{g7} + A_{g8}\tilde{u}_{g8} \tag{3.10}$$
$$= F - \alpha_5 u_5 - \alpha_6 u_6 - \alpha_{11} u_{11} - \alpha_{12} u_{12} - \alpha_{13} u_{13} - \alpha_{14} u_{14} - \alpha_{15} u_{15}$$
$$- \alpha_{16} u_{16} - \alpha_{17} u_{17} - \alpha_{18} u_{18},$$

where the coefficients A_{gl} and the 2D solutions $\tilde{u}_{gl} (l = 0, 1, ..., 8)$ of grid points from

group g are set in the group g part of Table 3.1.

Similarly, the coefficients (A_{gl} and α_{gl}) and F in Eq. (3.10) are from the unequal mesh-size coarse grid view Z-odd and need to be computed by the FOC scheme with unequal mesh-size. Because the u values in the right hand side of Eq. (3.10) have the computed solutions with sixth-order accuracy, we can compute a sixth-order solution for the grid points in the left hand side which are from group g.

We continue updating the fine grid points in group h. To this end, we build a Y-odd grid view formed by fine grid points from groups a and h, which looks like a combination of the $(even, even, even)$ coarse grid and the $(even, odd, even)$ coarse grid, as in Fig. 3.5.

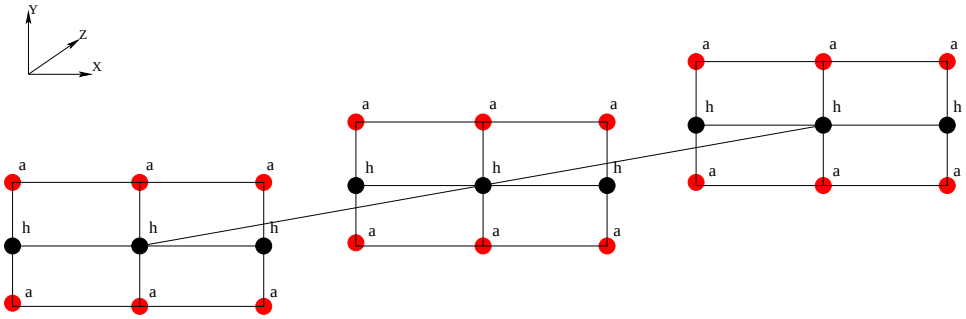


Figure 3.5: Y-odd grid view: fine grid points from groups a and h.

The Y-odd grid view is a view of unequal mesh-size grid with mesh-sizes 2Δ, Δ and 2Δ in the x, y and z coordinate directions, respectively. In Fig. 3.5, the red-colored points are the grid points from group a with sixth-order solutions, while the black-colored points are the grid points from group h with fourth-order solutions. When we visit the (x, z)-plane through the y direction for $y = odd$, all black-colored grid points could be solved plane by plane. There are $N_y/2$ 2D sub-problems, where N_y is the number of grid intervals along the y direction. For the (x, z)-plane 2D

sub-problem, its 9-point computational stencil is generated from Eq. (3.7) as

$$A_{h0}\tilde{u}_{h0} + A_{h1}\tilde{u}_{h1} + A_{h2}\tilde{u}_{h2} + A_{h3}\tilde{u}_{h3} + A_{h4}\tilde{u}_{h4} + A_{h5}\tilde{u}_{h5} + A_{h6}\tilde{u}_{h6} + A_{h7}\tilde{u}_{h7} + A_{h8}\tilde{u}_{h8} \tag{3.11}$$
$$= F - \alpha_2 u_2 - \alpha_4 u_4 - \alpha_7 u_7 - \alpha_8 u_8 - \alpha_9 u_9 - \alpha_{10} u_{10} - \alpha_{12} u_{12}$$
$$- \alpha_{14} u_{14} - \alpha_{16} u_{16} - \alpha_{18} u_{18},$$

where the coefficients A_{hl} and the 2D solutions $\tilde{u}_{hl}(l = 0, 1, ..., 8)$ of grid points from group h are set in the group h part of Table 3.1.

The coefficients (A_{hl} and α_{hl}) and F in Eq. (3.11) are from the unequal mesh-size coarse grid view Y-odd and we should use the FOC scheme with unequal mesh-size to compute them. Since the u values in the right hand side of Eq. (3.11) are from the grid points of group a with the sixth-order solution, we expect to get sixth-order solutions for the grid points in the left hand side.

Table 3.1: Settings of the coefficients $A_{\sharp l}$ and the solutions $\tilde{u}_{\sharp l}$ in 2D sub-problems ($\sharp$ denotes the group name).

group f				group g				group h			
A_{fl}	α_l	$\tilde{u}_{fl}$	u_l	A_{gl}	α_l	$\tilde{u}_{gl}$	u_l	A_{hl}	α_l	$\tilde{u}_{hl}$	u_l
A_{f0}	α_0	$\tilde{u}_{f0}$	u_0	A_{g0}	α_0	$\tilde{u}_{g0}$	u_0	A_{h0}	α_0	$\tilde{u}_{h0}$	u_0
A_{f1}	α_5	$\tilde{u}_{f1}$	u_5	A_{g1}	α_1	$\tilde{u}_{g1}$	u_1	A_{h1}	α_1	$\tilde{u}_{h1}$	u_1
A_{f2}	α_2	$\tilde{u}_{f2}$	u_2	A_{g2}	α_2	$\tilde{u}_{g2}$	u_2	A_{h2}	α_5	$\tilde{u}_{h2}$	u_5
A_{f3}	α_6	$\tilde{u}_{f3}$	u_6	A_{g3}	α_3	$\tilde{u}_{g3}$	u_3	A_{h3}	α_3	$\tilde{u}_{h3}$	u_3
A_{f4}	α_4	$\tilde{u}_{f4}$	u_4	A_{g4}	α_4	$\tilde{u}_{g4}$	u_4	A_{h4}	α_6	$\tilde{u}_{h4}$	u_6
A_{f5}	α_{12}	$\tilde{u}_{f5}$	u_{12}	A_{g5}	α_7	$\tilde{u}_{g5}$	u_7	A_{h5}	α_{11}	$\tilde{u}_{h5}$	u_{11}
A_{f6}	α_{16}	$\tilde{u}_{f6}$	u_{16}	A_{g6}	α_8	$\tilde{u}_{g6}$	u_8	A_{h6}	α_{13}	$\tilde{u}_{h6}$	u_{13}
A_{f7}	α_{18}	$\tilde{u}_{f7}$	u_{18}	A_{g7}	α_9	$\tilde{u}_{g7}$	u_9	A_{h7}	α_{17}	$\tilde{u}_{h7}$	u_{17}
A_{f8}	α_{14}	$\tilde{u}_{f8}$	u_{14}	A_{g8}	α_{10}	$\tilde{u}_{g8}$	u_{10}	A_{h8}	α_{15}	$\tilde{u}_{h8}$	u_{15}

Update fine grid points of groups c, d and e

Until now, we have four groups of fine grid points which have reached sixth-order solutions. We are going to update other three groups' fine grid points using the updated fine grid points. First, we create a Y-even grid view which is built up

with fine grid points from groups a, c, f and g. It looks like a combination of the $(even, even, even)$ coarse grid, the $(odd, even, odd)$ coarse grid, the $(odd, even, even)$ coarse grid and the $(even, even, odd)$ coarse grid, as in Fig. 3.6. We use this view to compute sixth-order solutions for the fine grid points of group c.

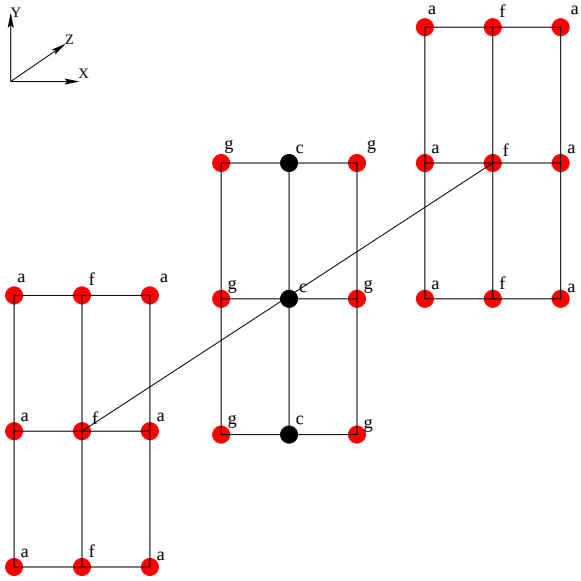


Figure 3.6: Y-even grid view: fine grid points from groups a, c, f and g.

The Y-even grid view has mesh-sizes Δ, 2Δ and Δ in the x, y and z coordinate directions, respectively. In Fig. 3.6, the red-colored grid points, which are from groups a, f and g, have computed solutions with sixth-order accuracy, while the black-colored grid points from group c have solutions with fourth-order accuracy. Note that the black points form vertical lines and we can use a tridiagonal solver in the y direction with $x = odd$ and $z = odd$ to solve all black-colored points line by line. Here we have $N_x/2 \times N_z/2$ Y-line 1D sub-problems, where N_x is the number of grid intervals along the x direction and N_z is the number of grid intervals along the z direction. The 3-point computational stencil of the Y-line 1D sub-problem is obtained from Eq.

(3.7) as

$$A_{c-1}\tilde{u}_{c-1} + A_{c0}\tilde{u}_{c0} + A_{c1}\tilde{u}_{c1} \tag{3.12}$$
$$= F - \alpha_1 u_1 - \alpha_3 u_3 - \alpha_5 u_5 - \alpha_6 u_6 - \alpha_7 u_7 - \alpha_8 u_8 - \alpha_9 u_9 - \alpha_{10} u_{10} - \alpha_{11} u_{11}$$
$$- \alpha_{12} u_{12} - \alpha_{13} u_{13} - \alpha_{14} u_{14} - \alpha_{15} u_{15} - \alpha_{16} u_{16} - \alpha_{17} u_{17} - \alpha_{18} u_{18},$$

where the coefficients A_{cl} and the 1D solutions $\tilde{u}_{cl}(l = -1, 0, 1)$ of grid points from group c are set in the group c part of Table 3.2.

At the same time, we are aware that the coefficients (A_{cl} and α_{cl}) and F in Eq. (3.12) are from the unequal mesh-size coarse grid view Y-even and need to be calculated by the FOC scheme with unequal mesh-size. For the u values in the right hand side of Eq. (3.12), the computed solutions with sixth-order accuracy from grid points of groups a, f and g are involved.

Next, we create another grid view to compute sixth-order solutions for the fine grid points of group d. The grid view called Z-even contains fine grid points from groups a, d, f and h, which can be viewed as a combination of the $(even, even, even)$ coarse grid, the $(odd, odd, even)$ coarse grid, the $(odd, even, even)$ coarse grid and the $(even, odd, even)$ coarse grid, as in Fig. 3.7.

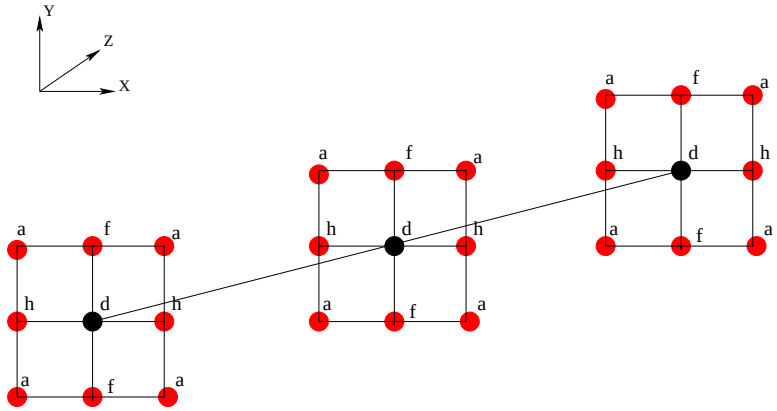


Figure 3.7: Z-even grid view: fine grid points from groups a, d, f and h.

The Z-even grid view has mesh-sizes Δ, Δ and 2Δ in the x, y and z coordinate directions, respectively. In Fig. 3.7, we use red to mark the grid points from groups a,

f and g with sixth-order solutions, and use black to mark the grid points from group d with fourth-order solutions. We notice that the black points form lines along the z direction and a tridiagonal solver can be applied to solve all black-colored points line by line for $x = odd$ and $y = odd$. There are $N_x/2 \times N_y/2$ Z-line 1D sub-problems, where N_x is the number of grid intervals along the x direction and N_y is the number of grid intervals along the y direction. For the Z-line 1D sub-problem, its 3-point computational stencil is obtained from Eq. (3.7) as

$$\begin{aligned} & A_{d-1}\tilde{u}_{d-1} + A_{d0}\tilde{u}_{d0} + A_{d1}\tilde{u}_{d1} \qquad (3.13) \\ & = F - \alpha_1 u_1 - \alpha_2 u_2 - \alpha_3 u_3 - \alpha_4 u_4 - \alpha_7 u_7 - \alpha_8 u_8 - \alpha_9 u_9 - \alpha_{10} u_{10} - \alpha_{11} u_{11} \\ & \quad - \alpha_{12} u_{12} - \alpha_{13} u_{13} - \alpha_{14} u_{14} - \alpha_{15} u_{15} - \alpha_{16} u_{16} - \alpha_{17} u_{17} - \alpha_{18} u_{18}, \end{aligned}$$

where the coefficients A_{dl} and the 1D solutions $\tilde{u}_{dl}(l = -1, 0, 1)$ of grid points from group d are set in the group d part of Table 3.2.

And we also note that the coefficients (A_{dl} and α_{dl}) and F in Eq. (3.13) are related to the unequal mesh-size coarse grid view Z-even and thus are calculated by the FOC scheme with unequal mesh-size. For the u values in the right hand side of Eq. (3.13), the computed solutions of grid points from groups a, f and h are used.

Fine grid points in group e are updated analogously. We combine the $(even, even, even)$ coarse grid, the $(even, odd, odd)$ coarse grid, the $(even, even, odd)$ coarse grid, and the $(even, odd, even)$ coarse grid to get an X-even grid view, which contains fine grid points from groups a, e, g and h, as in Fig. 3.8. We use this view to compute sixth-order solutions for the grid points of group e.

The X-even grid view has mesh-sizes 2Δ, Δ and Δ in the x, y and z coordinate directions, respectively. In Fig. 3.8, the red-colored points have computed solutions with sixth-order accuracy, which are from groups a, f and g. The black-colored points have computed solutions with fourth-order accuracy, which are from group e. Note that the black points form horizontal lines and we can use a tridiagonal solver in

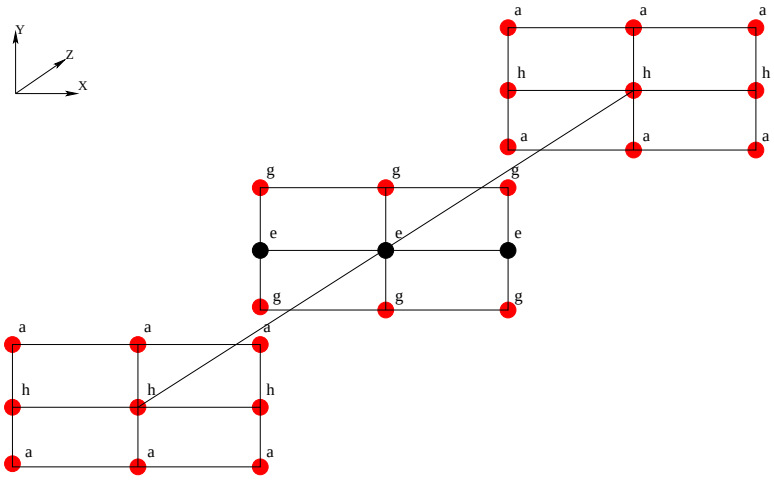


Figure 3.8: X-even grid view: fine grid points from groups a, e, g and h.

the x direction with $y = odd$ and $z = odd$ to solve all black-colored points line by line. There are totally $N_y/2 \times N_z/2$ X-line 1D sub-problems, where N_y is the number of grid intervals along the y direction and N_z is the number of grid intervals along the z direction. The 3-point computational stencil of the X-line 1D sub-problem is obtained from Eq. (3.7) as

$$\begin{aligned} & A_{e-1}\tilde{u}_{e-1} + A_{e0}\tilde{u}_{e0} + A_{e1}\tilde{u}_{e1} \qquad (3.14) \\ & = F - \alpha_2 u_2 - \alpha_4 u_4 - \alpha_5 u_5 - \alpha_6 u_6 - \alpha_7 u_7 - \alpha_8 u_8 - \alpha_9 u_9 - \alpha_{10} u_{10} - \alpha_{11} u_{11} \\ & \quad - \alpha_{12} u_{12} - \alpha_{13} u_{13} - \alpha_{14} u_{14} - \alpha_{15} u_{15} - \alpha_{16} u_{16} - \alpha_{17} u_{17} - \alpha_{18} u_{18}, \end{aligned}$$

where the coefficients A_{el} and the 1D solutions $\tilde{u}_{el}(l = -1, 0, 1)$ of grid points from group e are set in the group e part of Table 3.2.

And we notice that the coefficients (A_{el} and α_{el}) and F in Eq. (3.14) are from the unequal mesh-size coarse grid view X-even and could be calculated by the FOC scheme with unequal mesh-size. For the u values in the right hand side of Eq. (3.14), the computed sixth-order solutions of grid points from groups a, g and h are used.

Update fine grid points of group b

So far, we have updated six groups' fine grid points and computed sixth-order solutions for them. The last group of fine grid points with fourth-order solutions is group

Table 3.2: Settings of the coefficients $A_{\sharp l}$ and the solutions $\tilde{u}_{\sharp l}$ in 1D sub-problems ($\sharp$ denotes the group name).

group c				group d				group e			
A_{cl}	α_l	$\tilde{u}_{cl}$	u_l	A_{dl}	α_l	$\tilde{u}_{dl}$	u_l	A_{el}	α_l	$\tilde{u}_{el}$	u_l
A_{c-1}	α_4	$\tilde{u}_{c-1}$	u_4	A_{d-1}	α_6	$\tilde{u}_{d-1}$	u_6	A_{e-1}	α_3	$\tilde{u}_{e-1}$	u_3
A_{c0}	α_0	$\tilde{u}_{c0}$	u_0	A_{d0}	α_0	$\tilde{u}_{d0}$	u_0	A_{e0}	α_0	$\tilde{u}_{e0}$	u_0
A_{c1}	α_2	$\tilde{u}_{c1}$	u_2	A_{d1}	α_5	$\tilde{u}_{d1}$	u_5	A_{e1}	α_1	$\tilde{u}_{e1}$	u_1

b, which contains (odd, odd, odd) fine grid points. Since every (odd, odd, odd) grid point is immediately surrounded by grid points with sixth-order solutions, some suitable interpolation can be used to compute a sixth-order solution. We choose to use a one step operator based interpolation [80] to update the solution for every (odd, odd, odd) fine grid point by the following equation

$$\begin{aligned}\tilde{u}_{i,j,k} = [&F_{i,j,k} - \alpha_1 u_{i+1,j,k} - \alpha_2 u_{i,j+1,k} - \alpha_3 u_{i-1,j,k} - \alpha_4 u_{i,j-1,k} \\ &- \alpha_5 u_{i,j,k+1} - \alpha_6 u_{i,j,k-1} - \alpha_7 u_{i+1,j+1,k} - \alpha_8 u_{i-1,j+1,k} \\ &- \alpha_9 u_{i-1,j-1,k} - \alpha_{10} u_{i+1,j-1,k} - \alpha_{11} u_{i+1,j,k+1} - \alpha_{12} u_{i,j+1,k+1} \\ &- \alpha_{13} u_{i-1,j,k+1} - \alpha_{14} u_{i,j-1,k+1} - \alpha_{15} u_{i+1,j,k-1} - \alpha_{16} u_{i,j+1,k-1} \\ &- \alpha_{17} u_{i-1,j,k-1} - \alpha_{18} u_{i,j-1,k-1}]/\alpha_0.\end{aligned} \tag{3.15}$$

Here, α_l are the coefficients in Eq. (3.7) and $F_{i,j,k}$ represents the right hand side part of Eq. (3.7). Since all grid points in the right hand side have computed sixth-order solutions, $\tilde{u}_{i,j,k}$ can get the sixth-order solution.

In the above updating processes, we used multiple coarse grids to generate a series of direct solutions for fine grid points, which can replace the iterative refinement procedure so as to improve the computational efficiency. Another potential benefit of using multiple coarse grids is the concurrency. We notice that the fine grid points in groups f, g, and h can be updated in parallel since the computational processes only depend on the grid points from group a with sixth-order solutions. In addition, within each groups, all 2D sub-problems can be computed simultaneously. For the

fine grid points in groups c, d and e, they can also be updated in parallel since the computational processes only need updated grid points with sixth-order solutions from groups a, c, d and e. Therefore, the proposed MCG updating strategy has potential to be suitable for current generation supercomputers with large numbers of processors.

3.3.3 MSMG Method with Richardson extrapolation and MCG updating strategy for the 3D convection-diffusion equation

By now, we have described all the strategies needed to compute a high accuracy solution for the 3D convection-diffusion equation with high efficiency. Algorithm 3 gives out the complete description.

3.4 Numerical Results

We tested our sixth-order method (MCG-update-six) and compared the results with Wang-Zhang's sixth-order method (Iter-update-six) [80]. The codes were written in Fortran 77 programming language and all computations were run on one node of the Lipscomb HPC Cluster at the University of Kentucky. The node has 12 cores with 2.66GHz and 36 GB RAM.

The domain Ω for the following two test cases was the unit cube $(0,1)^3$. For both cases, we tested for various Reynolds numbers, respectively. For the diffusion-dominant equations with small Reynolds number, we could get almost sixth-order solutions since the computed solutions from the FOC scheme have the order of four in accuracy and p in Richardson extrapolation Eq. (3.8) is set as 4 so that the extrapolated solution could be upgraded to the order of six. When the Reynolds number increases, the order of accuracy of computed solutions using the FOC scheme varies from four to two and the accuracy improvement from the extrapolation is degraded [81].

We used the standard V(1,1)-cycle in the MSMG method. The initial guess for the

Algorithm 3 Sixth order compact approximation for the 3D convection-diffusion equation using the MSMG method with the MCG updating strategy and Richardson extrapolation

1. Use the MSMG method to compute the fourth-order solutions $u^{2\Delta} \in \Omega^4_{2\Delta}$ and $u^{\Delta} \in \Omega^4_{\Delta}$.
2. Divide fine grid points on Ω_{Δ} into eight groups, see Fig. 3.2.
3. *Update fine grid points of group a.*
From $u^{2\Delta} \in \Omega^4_{2\Delta}$ and $u^{\Delta} \in \Omega^4_{\Delta}$, compute $\tilde{u}^{2\Delta} \in \Omega^6_{2\Delta}$ by Richardson extrapolation using Eq. (3.8); Directly interpolate the sixth-order coarse grid solutions $\tilde{u}^{2\Delta}_{i,j,k}$ to the corresponding fine grid points in group a to get $\tilde{u}^{\Delta}_{2i,2j,2k} \in \Omega^6_{\Delta}$.
4. *Update fine grid points of groups f, g and h.*
Use fine grid points from groups a and f to create an X-odd grid view, see Fig. 3.3; Solve $N_x/2$ 2D sub-problems using Eq. (3.9) to get $\tilde{u}^{\Delta}_{i,2j,2k} \in \Omega^6_{\Delta}$ for fine grid points of group f.
Use fine grid points from groups a and g to create a Z-odd grid view, see Fig. 3.4; Solve $N_z/2$ 2D sub-problems using Eq. (3.10) to get $\tilde{u}^{\Delta}_{2i,2j,k} \in \Omega^6_{\Delta}$ for fine grid points of group g.
Use fine grid points from groups a and h to create a Y-odd grid view, see Fig. 3.5; Solve $N_y/2$ 2D sub-problems using Eq. (3.11) to get $\tilde{u}^{\Delta}_{2i,j,2k} \in \Omega^6_{\Delta}$ for fine grid points of group h.
5. *Update fine grid points of groups c, d and e.*
Use fine grid points from groups a, f, g and c to create a Y-even grid view, see Fig. 3.6; Solve $N_x/2 \times N_z/2$ 1D sub-problems using Eq. (3.12) to get $\tilde{u}^{\Delta}_{i,2j,k} \in \Omega^6_{\Delta}$ for fine grid points of group c.
Use fine grid points from groups a, f, h and d to create a Z-even grid view, see Fig. 3.7; Solve $N_x/2 \times N_y/2$ 1D sub-problems using Eq. (3.13) to get $\tilde{u}^{\Delta}_{i,j,2k} \in \Omega^6_{\Delta}$ for fine grid points of group d.
Use fine grid points from groups a, g, h and e to create an X-even grid view, see Fig. 3.8; Solve $N_y/2 \times N_z/2$ 1D sub-problems using Eq. (3.14) to get $\tilde{u}^{\Delta}_{2i,j,k} \in \Omega^6_{\Delta}$ for fine grid points of group e.
6. *Update fine grid points of group b.*
For every fine grid point of group b, do a one step operator based interpolation on Ω_{Δ} using
Eq. (3.15) to get $\tilde{u}^{\Delta}_{2i+1,2j+1,2k+1} \in \Omega^6_{\Delta}$.

V-Cycle on $\Omega_{4\Delta}$ was the zero vector. The multigrid V-cycle on $\Omega_{2\Delta}$ and Ω_{Δ} stopped when the L^2-norm of the difference of the successive solutions was reduced by a factor of 10^{10}. In the Iter-update-six method, the refinement iterative procedure was terminated when the L^2-norm of the correction vector of the approximate solution was less than 10^{-10}. In the MCG-update-six method, there are two selections for 2D sub-problem solver. One is traditional iterative methods, such as Gauss-Seidel method. The other is multigrid methods. We compared them in numerical experiments and set the stopping criteria for both as 10^{-10}. Though literatures [48, 73] show that exact solutions of 2D sub-problems are not necessary in solving 3D problems and that one multigrid cycle or Gauss-Seidel relaxation is sufficient, we computed full converged solutions here. By setting the same stopping criteria in the MCG updating process as in the iterative refinement procedure, we could compare the two sixth-order methods under the conditions as close as possible. The errors reported were the maximum absolute errors over the finest grid.

3.4.1 Test problem 1

The first test problem is

$$\begin{cases} u(x,y,z) = \cos(4x+6y+8z) \\ p(x,y,z) = Re\sin y\sin z\cos x \\ q(x,y,z) = Re\sin x\sin z\cos y \\ r(x,y,z) = Re\sin x\sin y\cos z \end{cases}.$$

This problem has variable coefficients and the constant Re represents the magnitude of the convection coefficients. The Dirichlet boundary conditions and the forcing term f are set to satisfy the exact solution. Assume $N_x = N_y = N_z = n$. We used the point Gauss-Seidel relaxation as the smoother for both sixth-order methods in the MSMG computation.

We first set $Re = 0$, which reduces the problem to a 3D Poisson equation. Table 3.3 contains the numerical results, which compare the maximum absolute errors, the CPU time in seconds, the number of iterations and the order of accuracy for the

computed solutions. The number of iterations has three parts. They are the number of V-cycles for $\Omega_{2\Delta}$, the number of V-cycles for Ω_{Δ}, and the number of iterations for the iterative operator based interpolation in the Iter-update-six method. There are two kinds of recorded CPU time. "Total CPU" is the elapsed time for the whole solving process of the problem. "Updating CPU" is the CPU time for fine grid updating to reach the sixth-order solution from the fourth-order solutions on the fine and coarse grids. For the Iter-update-six method, this part is the iterative refinement procedure with Richardson extrapolation. For the MCG-update-six, it is the MCG fine grid updating process with Richardson extrapolation.

Table 3.3: Comparison of the number of iterations, the CPU time in seconds, the maximum errors and the order of accuracy between the Iter-update-six method and the MCG-update-six methods with different 2D sub-problem solvers for solving Problem 1 with $Re = 0$.

n	Method	# iteration	Total CPU(s)	Updating CPU(s)	Error	Order
8	Iter-update-six	(8,11), 33	0.005	0.002	1.55e-3	-
	MCG-update-six(2D-line)	(8,11), -	0.005	0.001	5.64e-3	-
	MCG-update-six(2D-MG)	(8,11) -	0.006	0.002	5.64e-3	-
16	Iter-update-six	(11,12), 43	0.050	0.018	4.90e-5	4.98
	MCG-update-six(2D-line)	(11,12), -	0.042	0.010	1.19e-4	5.57
	MCG-update-six(2D-MG)	(11,12) -	0.043	0.011	1.19e-4	5.57
32	Iter-update-six	(12,11), 44	0.447	0.171	1.15e-6	5.41
	MCG-update-six(2D-line)	(12,11), -	0.360	0.084	2.05e-6	5.86
	MCG-update-six(2D-MG)	(12,11), -	0.363	0.087	2.05e-6	5.86
64	Iter-update-six	(11,11), 43	4.847	2.557	2.14e-8	5.75
	MCG-update-six(2D-line)	(11,11), -	1.990	0.672	3.31e-8	5.95
	MCG-update-six(2D-MG)	(11,11), -	2.999	0.680	3.31e-8	5.95
128	Iter-update-six	(11,11), 39	42.148	22.862	3.81e-10	5.81
	MCG-update-six(2D-line)	(11,11), -	24.952	5.548	5.56e-10	5.90
	MCG-update-six(2D-MG)	(11,11), -	25.005	5.601	5.56e-10	5.90

Table 3.3 verifies that the new MCG updating strategy is more efficient than the iterative updating strategy. When the mesh became finer, the CPU time for the Iter-update-six method increased very quickly and was mainly from the iterative refinement procedure, which is demonstrated by the "Updating CPU" column. For the MCG-update-six method, since we used a series of direct solutions to update fine

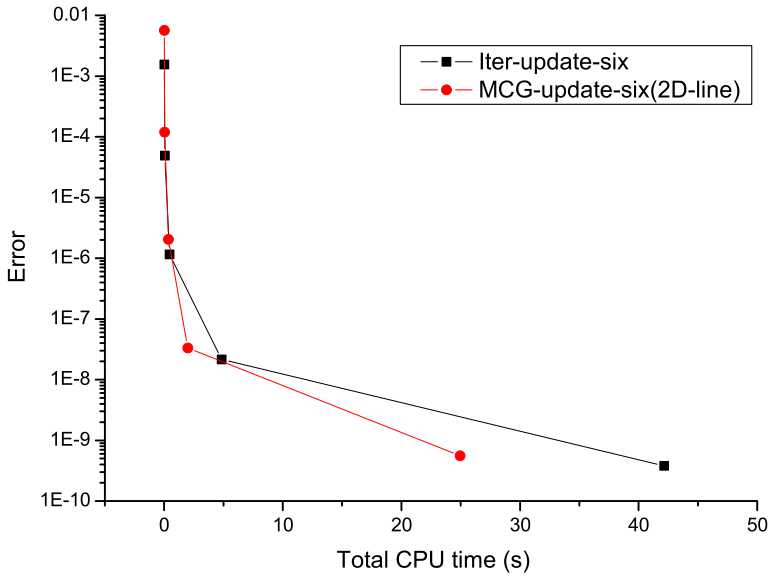


Figure 3.9: Comparison of the maximum errors and the total CPU time between the Iter-update-six method and the MCG-update-six(2D-line) method for solving Problem 1(Re=0). Each symbol with increasing CPU time corresponds to an increasing fine grid: 8, 16, 32, 64, and 128 intervals.

grid points and thus eliminated the iterative refinement procedure on the 3D fine grid, the CPU time for the updating process was reduced effectively. For instance, when $n = 128$, the Iter-update-six method spent 42.148 seconds to solve the problem and 22.862 seconds on the iterative updating procedure on the finest grid; while the MCG-update-six with Gauss-Seidel line solver for the 2D sub-problems took 24.952 seconds for solving the whole problem and 5.548 seconds for computing sixth-order solutions for the finest grid points. As to the solution accuracy, though the Iter-update-six method showed a little bit more accurate than the MCG-update-six method, their maximum absolute errors for different discretized grids were in the same order of magnitude. The order of accuracy for the computed solutions from both methods were close to six as we expected.

Table 3.3 also compares two kinds of 2D sub-problem solver. "MCG-update-six(2D-line)" used the line Gauss-Seidel iterative method to solve the 2D sub-problems

and "MCG-update-six(2D-MG)" used the multigrid method with Red Black Gauss-Seidel relaxation to solve the 2D sub-problems. By using either 2D solver, the MCG-update-six computed the same accurate solution, which means both methods had converged when solving the 2D sub-problems. The experimental results show that the line Gauss-Seidel solver ran a little bit faster than the multigrid method.

Fig. 3.9 describes the relationship between the maximum errors and the total CPU cost for the two sixth-order methods. It illustrates that the MCG-update-six(2D-line) method spent less time than the Iter-update-six method to compute a certain accurate solution. When a solution with high accuracy is required, such as the maximum error is no more than 10^{-8}, the superiority of the MCG-update-six method on computational time is apparent.

Similar conclusions are summarized in Table 3.4 when $Re = 10$. First, the MCG-update-six method ran faster than the Iter-update-six method and the efficiency benefits were mainly from the new updating strategy. Second, the solution accuracy obtained from both methods was comparable. Third, both 2D solvers converged in 2D sub-problem solutions and the line Gauss-Seidel solver did slightly faster.

For better understanding, we plotted the updating CPU time in two sixth-order methods for solving the Problem 1 with $Re = 10$ on different scale grids in Fig. 3.10. It is obvious that the MCG updating strategy has higher computational efficiency and better scalability than the iterative refinement procedure.

Then we tested for large Reynolds number cases ($Re = 10^3$ and $Re = 10^4$) and reported numerical results in Table 3.5. We chose Gauss-Seidel line solver for 2D sub-problems in the MCG-update-six method. From Table 3.5 we find that the MCG-update-six method has obvious advantages in computational efficiency for difficult problems with large Reynolds number. For instance, when $Re = 10^4$ and $n = 128$, the iterative refinement procedure needed 240.833 seconds to update solutions on the finest grid, while the MCG strategy only took 12.589 seconds. For solving the whole

Table 3.4: Comparison of the number of iterations, the CPU time in seconds, the maximum errors and the order of accuracy between the Iter-update-six method and the MCG-update-six methods with different 2D sub-problem solvers for solving Problem 1 with $Re = 10$.

n	Method	# iteration	Total CPU(s)	Updating CPU(s)	Error	Order
8	Iter-update-six	(9,12), 34	0.005	0.002	1.95e-3	–
	MCG-update-six(2D-line)	(9,12), -	0.006	0.003	8.76e-3	–
	MCG-update-six(2D-MG)	(9,12), -	0.006	0.003	8.76e-3	–
16	Iter-update-six	(12,13), 46	0.052	0.019	6.13e-5	4.99
	MCG-update-six(2D-line)	(12,13), -	0.051	0.018	2.06e-4	5.41
	MCG-update-six(2D-MG)	(12,13) -	0.051	0.018	2.06e-4	5.41
32	Iter-update-six	(13,12), 47	0.460	0.182	1.40e-6	5.45
	MCG-update-six(2D-line)	(13,12), -	0.416	0.138	3.69e-6	5.80
	MCG-update-six(2D-MG)	(13,12), -	0.427	0.149	3.69e-6	5.80
64	Iter-update-six	(12,12), 44	5.089	2.655	2.56e-8	5.76
	MCG-update-six(2D-line)	(12,12), -	3.558	1.121	6.03e-8	5.93
	MCG-update-six(2D-MG)	(12,12), -	3.643	1.213	6.03e-8	5.93
128	Iter-update-six	(12,11), 40	40.069	20.807	9.70e-10	4.72
	MCG-update-six(2D-line)	(12,11), -	28.387	9.133	1.40e-9	5.43
	MCG-update-six(2D-MG)	(12,11), -	29.219	9.943	1.40e-9	5.43

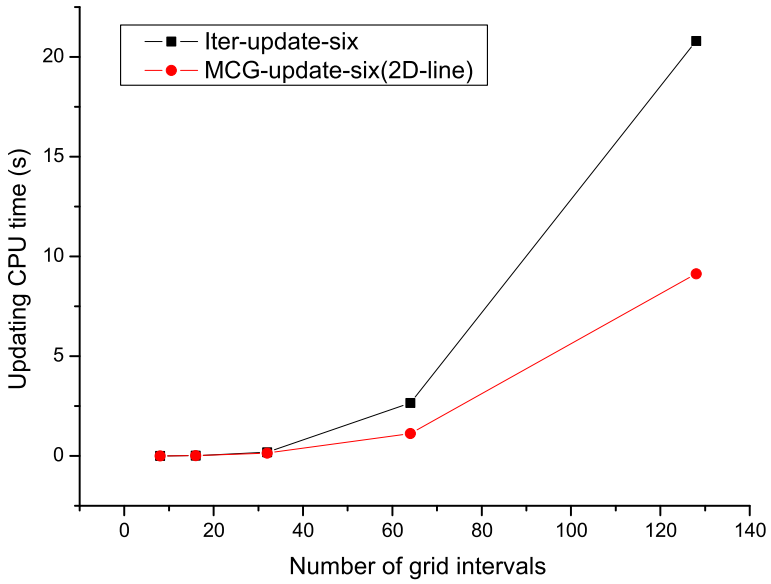


Figure 3.10: Comparison of the updating CPU time and the number of grid intervals between the Iter-update-six method and the MCG-update-six(2D-line) method for solving Problem 1 ($Re = 10$). Each symbol with increasing CPU time corresponds to an increasing fine grid: 8, 16, 32, 64, and 128 intervals.

Table 3.5: Comparison of the number of iterations, the CPU time in seconds, the maximum errors and the order of accuracy between the Iter-update-six method and the MCG-update-six methods for solving Problem 1 with $Re = 10^3$ and $Re = 10^4$.

Re	n	Method	# iteration	Total CPU(s)	Updating CPU(s)	Error	Order
10^3	16	Iter-update-six	(47,67), 91	0.155	0.039	1.07e-2	2.84
		MCG-update-six	(47,67), -	0.135	0.020	4.21e-2	2.64
	32	Iter-update-six	(67,87), 96	1.683	0.378	1.05e-3	3.36
		MCG-update-six	(67,87), -	1.452	0.155	3.77e-3	3.48
	64	Iter-update-six	(87,124), 67	19.950	4,146	3.41e-5	4.94
		MCG-update-six	(87,124), -	16.788	1.187	1.41e-4	4.74
	128	Iter-update-six	(124,181), 59	226.062	40.959	7.17e-7	5.57
		MCG-update-six	(124,181), -	195.657	9.477	2.86e-6	5.62
10^4	16	Iter-update-six	(54,150), 126	0.288	0.053	1.77e-2	2.20
		MCG-update-six	(54,150), -	0.253	0.020	5.15e-2	2.35
	32	Iter-update-six	(150,369), 230	5.898	0.889	3.39e-3	2.38
		MCG-update-six	(150,369), -	5.150	0.181	8.46e-3	2.61
	64	Iter-update-six	(369,382), 348	68.695	20.755	4.90e-4	2.79
		MCG-update-six	(369,382), -	48.987	1.679	1.01e-3	3.07
	128	Iter-update-six	(382,360), 422	671.852	240.833	3.55e-5	3.79
		MCG-update-six	(382,360), -	389.847	12.589	7.24e-5	3.80

problem, the Iter-update-six method spent 671.852 seconds while the MCG-update-six method took 389.847 seconds, which denotes that, compared to the Iter-update-six method, the proposed sixth-order method is able to save 40% computing time for solving this difficult problem. And we also note that the magnitude of Reynolds number affected the computed solution accuracy inversely. The reason lies in the effects of high Reynolds number on the fourth-order and sixth-order truncation error terms from the FOC scheme. More details about the analysis of such affects can be referred to [81].

3.4.2 Test problem 2

The second test case is from a test problem of Gupta and Zhang's high accuracy multigrid solution of the 3D convection-diffusion equation [36], which can be written

Table 3.6: Comparison of the number of iterations, the CPU time in seconds, the maximum errors and the order of accuracy between the Iter-update-six method and the MCG-update-six methods with different 2D sub-problem solvers for solving Problem 2 with $Re = 10$.

n	Method	# iteration	Total CPU(s)	Updating CPU(s)	Error	Order
8	Iter-update-six	(9,12), 28	0.005	0.001	2.17e-4	-
	MCG-update-six(2D-line)	(9,12), -	0.007	0.003	5.96e-4	-
	MCG-update-six(2D-MG)	(9,12) -	0.007	0.003	5.96e-4	-
16	Iter-update-six	(12,12), 36	0.046	0.014	5.86e-6	5.21
	MCG-update-six(2D-line)	(12,12), -	0.056	0.025	1.46e-5	5.35
	MCG-update-six(2D-MG)	(12,12) -	0.057	0.026	1.46e-5	5.35
32	Iter-update-six	(12,12), 35	0.411	0.136	1.24e-7	5.56
	MCG-update-six(2D-line)	(12,12), -	0.459	0.187	2.85e-7	5.68
	MCG-update-six(2D-MG)	(12,12), -	0.469	0.197	2.85e-7	5.68
64	Iter-update-six	(12,11), 30	4.715	2.382	2.44e-9	5.68
	MCG-update-six(2D-line)	(12,11), -	3.821	1.524	5.11e-9	5.80
	MCG-update-six(2D-MG)	(12,11), -	3.858	1.567	5.11e-9	5.80
128	Iter-update-six	(11,11), 25	34.960	15.959	2.41e-10	3.34
	MCG-update-six(2D-line)	(11,11), -	30.967	11.940	2.75e-10	4.23
	MCG-update-six(2D-MG)	(11,11), -	32.010	12.668	2.75e-10	4.23

as

$$\begin{cases} u(x,y,z) = xyz(1-x)(1-y)(1-z)exp(x+y+z) \\ p(x,y,z) = Re\sin y\sin z\cos x \\ q(x,y,z) = Re\sin x\sin z\cos y \\ r(x,y,z) = Re\sin x\sin y\cos z \end{cases}.$$

First, we used the point relaxation smoothers in the MSMG computation and tested a diffusion-dominant equation with a small Reynolds number (Re=10). The numerical comparison on two strategies were listed in Table 3.6. When the number of intervals was relatively small ($n \leqslant 32$), the Iter-update-six method yielded better performance. When n became large, the MCG-update-six method ran faster and kept comparable solution accuracy. Therefore, the proposed method is a scalable computational strategy. As for the 2D sub-problem solvers in the MCG-update-six method, both methods converged and the line Gauss-Seidel solver consistently showed a little bit superiority in computational efficiency.

Then we compared two methods in solving some convection-dominated equations. We chose the line Gauss-Seidel solver for 2D sub-problems in the MCG-update-six

Table 3.7: Comparison of the number of iterations, the CPU time in seconds, the maximum errors and the order of accuracy between the Iter-update-six method and the MCG-update-six methods for solving Problem 2 with $Re = 10^3$ and $Re = 10^4$.

Re	n	Method	# iteration	Total CPU(s)	Updating CPU(s)	Error	Order
10^3	16	Iter-update-six	(52,67), 72	0.147	0.030	7.93e-3	2.55
		MCG-update-six	(52,67), -	0.144	0.027	1.32e-2	2.62
	32	Iter-update-six	(67,83), 72	1.588	0.286	2.88e-4	4.78
		MCG-update-six	(67,83), -	1.450	0.201	6.98e-4	4.24
	64	Iter-update-six	(83,117), 51	18.975	3.904	4.72e-6	5.93
		MCG-update-six	(83,117), -	16.381	1.547	1.95e-5	5.16
	128	Iter-update-six	(117,170), 45	204.996	27.525	1.05e-7	5.49
		MCG-update-six	(117,170), -	189.660	12.343	4.18e-7	5.54
10^4	16	Iter-update-six	(63,189), 97	0.335	0.041	1.19e-2	1.81
		MCG-update-six	(63,189), -	0.320	0.027	1.72e-2	2.18
	32	Iter-update-six	(189,401), 166	6.082	0.644	3.16e-3	1.91
		MCG-update-six	(189,401), -	5.658	0.224	3.51e-3	2.29
	64	Iter-update-six	(401,358), 235	64.733	18.654	3.06e-4	3.37
		MCG-update-six	(401,358), -	47.213	1.912	3.28e-4	3.42
	128	Iter-update-six	(358,342), 277	525.251	161.614	6.12e-6	5.64
		MCG-update-six	(358,342), -	373.307	14.621	8.45e-6	5.28

method. Numerical results for $Re = 10^3$ and $Re = 10^4$ are described in Table 3.7 and Fig. 3.11. It is visible that the proposed method performed better on scalability and efficiency when we increased the number of grid intervals, especially for the problem with large Reynolds numbers. For instance, when $Re = 10^4$ and $n = 128$, the MCG-update-six method took nearly 30% less total CPU time than the Iter-update-six method to compute a solution with the same magnitude error. The improvement is from the new MCG updating strategy, which can be used to eliminate the iterative refinement procedure for high accuracy solution computation on the finest grid. The CPU time saved by the MCG updating strategy is displayed in the column of "Updating CPU" in Table 3.7. Again, the experimental results show that for large magnitude of Reynolds number, the convergence and the computed accuracy were severely degraded.

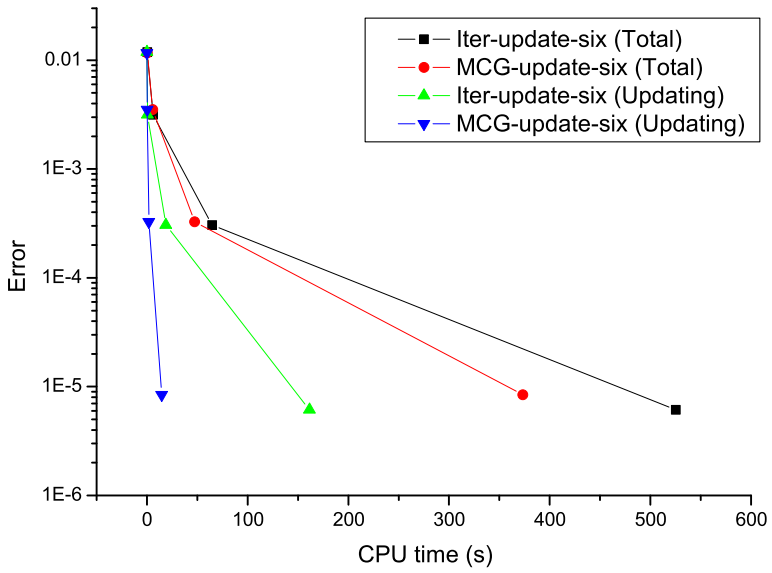


Figure 3.11: Comparison of the maximum absolute errors and the CPU time for solving Problem 2 ($Re = 10^4$). Each symbol with increasing CPU time corresponds to an increasing fine grid: 16, 32, 64, and 128 intervals.

3.5 Concluding Remarks

We improved the sixth-order compact computation for the 3D convection-diffusion equation. A new fine grid updating strategy based on the MCG computation is proposed, which can replace the iterative refinement procedure on the finest grid in the current MSMG method for the sixth-order compact approximation. We also derived a 19-point FOC scheme with unequal mesh-size for the 3D convection-diffusion equation. An algorithm is given to describe our sixth-order compact computation for the 3D convection-diffusion equation by using the MSMG method with the MCG updating strategy and the Richardson extrapolation technique.

The numerical results show that the MSMG method with the MCG updating strategy is more cost-effective than the MSMG method with the iterative refinement procedure to compute solutions with comparable accuracy. The proposed method also demonstrates a better scalability for problems with a large number of unknowns.

In addition, the MCG updating strategy supports concurrency and has good potential for parallelization.

4 Sixth-Order Solution with Completed Richardson Extrapolation for Steady-State Equations

4.1 Introduction

In this chapter, we consider another type of Richardson extrapolation-based sixth-order method, which uses the completed Richardson extrapolation technique to produce sixth-order solutions at all fine grid points. The completed Richardson extrapolation was first developed by Roache and Knupp [60] to produce a fourth-order solution on the fine grid. They did not use the extrapolated fourth-order solution but rather the correction between the second-order solution and the fourth-order solution in the interpolation process. Here, we borrow the idea from the completed Richardson extrapolation and similarly use the correction between the fourth-order solution and the extrapolated sixth-order solution rather than the extrapolated sixth-order solution to obtain a sixth-order solution on the entire fine grid. Since the completed Richardson extrapolation procedure neither requires special treatment for near-boundary points, nor involves significant computational cost, we can expect to reach high efficiency at the same time.

Consider a 1D uniform fine grid $j = 0, 1, 2, ...$ with mesh-size h on which a fourth-order solution is computed by some FOC scheme. A separate fourth-order solution on the subgrid (coarse grid) with mesh-size $2h$ of even points $j = 0, 2, 4...$ can also be computed. By applying Richardson extrapolation, a sixth-order solution on the subgrid of even-numbered grid points $j = 0, 2, 4, ...$ is obtained. We want to obtain a sixth-order solution on the fine grid points which were skipped in the Richardson extrapolation process, i.e., the odd-numbered grid points $j = 1, 3, 5, ...$ Instead of seeking some appropriate interpolation on the extrapolated sixth-order solution [69], the difference between the fourth-order solution and the extrapolated sixth-order solution can be utilized.

Let u_j^* be the exact solution at grid point j, u_j^h be the fine grid fourth-order

solution, and u_j^{2h} be the subgrid fourth-order solution. The extrapolated sixth-order solution $\tilde{u}_j$ is obtained on the odd-numbered grid points by Richardson extrapolation as

$$\tilde{u}_j = \frac{16}{15}u_j^h - \frac{1}{15}u_j^{2h}. \quad j = even$$

This extrapolation can be conveniently expressed in terms of c_j, the correction from the fourth-order solution to the sixth-order solution, as

$$\tilde{u}_j = u_j^h + c_j, \quad j = even \tag{4.1}$$

where

$$c_j = \frac{1}{15}(u_j^h - u_j^{2h}). \quad j = even \tag{4.2}$$

This c_j can be considered a fourth-order accurate error estimator.

The solution accuracy has the definition of

$$u_j^* = u_j^h + A_j h^4 + O(h^{5+m}), \quad j = even \tag{4.3}$$

$$u_{j+1}^* = u_{j+1}^h + A_{j+1} h^4 + O(h^{5+m}), \tag{4.4}$$

$$u_{j+2}^* = u_{j+2}^h + A_{j+2} h^4 + O(h^{5+m}), \tag{4.5}$$

where the As are the coefficients of the leading error terms, which vary spatially and is independent of h. By using simple two-point linear interpolation on smooth solutions, we have

$$A_{j+1} = 1/2(A_j + A_{j+2}) + O(h^2). \quad j + 1 = odd \tag{4.6}$$

Increasing the order of this interpolation will not improve the order of the overall method since the accuracy is limited by the error terms of $O(h^{5+m})$.

Evaluating A_j for even-numbered points from Eq. (4.3) gives

$$A_j = \frac{1}{h^4}[u_j^* - u_j^h + O(h^{5+m})]. \quad j = even \tag{4.7}$$

The sixth-order solution is defined as

$$u_j^* = \tilde{u}_j + O(h^6). \quad j = even \tag{4.8}$$

Substituting (4.8) into (4.7) obtains

$$A_j = \frac{1}{h^4}[\tilde{u}_j - u_j^h + O(h^{5+m})]. \quad j = even \tag{4.9}$$

Similarly,

$$A_{j+2} = \frac{1}{h^4}[\tilde{u}_{j+2} - u_{j+2}^h + O(h^{5+m})]. \quad j = even \tag{4.10}$$

Using (4.9) and (4.10) in (4.6) gives

$$A_{j+1} = \frac{1}{2h^4}[\tilde{u}_j - u_j^h + \tilde{u}_{j+2} - u_{j+2}^h + O(h^{5+m})]. \quad j+1 = odd \tag{4.11}$$

This defines the completed Richardson extrapolation method.

For clarity we can write the correction c_j of (4.1) from the fourth-order solution to the $(5+m)$th order solution as

$$c_j = \tilde{u}_j - u_j^h. \quad j = even \tag{4.12}$$

Eq. (4.12) is the correction of the original Richardson extrapolation. Then at the odd-numbered fine grid points, not covered by the the original Richardson extrapolation, the correction from the fourth-order solution to the $(5+m)$th order solution is approximated by

$$\tilde{u}_{j+1} = u_{j+1}^h + c_{j+1}, \quad j+1 = odd$$

where

$$c_{j+1} = \frac{1}{2}(c_j + c_{j+2}). \quad j+1 = odd$$

The second error term of the fourth-order solution, $O(h^{5+m})$ in Eq. (4.3), will limit the accuracy of the completed Richardson extrapolation. When using the FOC scheme Eq.(1.7), there is no fifth-order error term and thus the completed Richardson extrapolation can obtain the sixth-order accurate solution.

4.2 Sixth-Order Solution with Completed Richardson Extrapolation for 2D Problems

For 2D problems, let $u_{i,j}^*$ be the exact solution at fine grid point (i,j), $u_{i,j}^h$ be the fourth-order solution at fine grid point (i,j) with mesh-size h, and $u_{i,j}^{2h}$ be the fourth-

order solution at coarse grid point (i, j) with mesh-size $2h$. Then the Richardson extrapolation formula

$$\tilde{u}_{i,j}^{2h} = \frac{(16u_{2i,2j}^{h} - u_{i,j}^{2h})}{15} \tag{4.13}$$

is used to compute the sixth-order solution $\tilde{u}_{i,j}^{2h}$ on the coarse grid.

By using direct interpolation, the sixth-order solution $\tilde{u}_{2i,2j}^{h}$ at fine grid point $(2i, 2j)$ is obtained from the extrapolated sixth-order solution $\tilde{u}_{i,j}^{2h}$ at coarse grid point (i, j). Then we can write

$$\tilde{u}_{2i,2j}^{h} = \frac{16}{15}u_{2i,2j}^{h} - \frac{1}{15}u_{i,j}^{2h}. \tag{4.14}$$

We rewrite the extrapolation in terms of $c_{2i,2j}^{h}$, the correction from the fourth-order solution to the sixth-order solution for $(even, even)$ fine grid points, as

$$\tilde{u}_{2i,2j}^{h} = u_{2i,2j}^{h} + c_{2i,2j}^{h}, \tag{4.15}$$

where

$$c_{2i,2j}^{h} = \frac{1}{15}(u_{2i,2j}^{h} - u_{i,j}^{2h}).$$

Then, we consider (odd, odd) fine grid points. If we use Eq. (2.10) to compute fourth-order accurate solutions, we have

$$u_{i,j}^{*} = u_{i,j}^{h} + A_{i,j}h^4 + O(h^6), \quad i = odd, j = odd \tag{4.16}$$

$$u_{i+1,j+1}^{*} = u_{i+1,j+1}^{h} + A_{i+1,j+1}h^4 + O(h^6), \tag{4.17}$$

$$u_{i+1,j-1}^{*} = u_{i+1,j-1}^{h} + A_{i+1,j-1}h^4 + O(h^6), \tag{4.18}$$

$$u_{i-1,j+1}^{*} = u_{i-1,j+1}^{h} + A_{i-1,j+1}h^4 + O(h^6), \tag{4.19}$$

$$u_{i-1,j-1}^{*} = u_{i-1,j-1}^{h} + A_{i-1,j-1}h^4 + O(h^6), \tag{4.20}$$

where As are the coefficients of the leading error terms, which vary spatially and is independent of h.

By using rotated grid interpolation on smooth solutions, as Fig. 4.1(a), we have

$$A_{i,j} = \frac{1}{4}(A_{i+1,j+1} + A_{i+1,j-1} + A_{i-1,j+1} + A_{i-1,j-1}) + O(h^2) \tag{4.21}$$

for (odd, odd) fine grid points. Since there is an h^4 term with the A in Eq. (4.16), a second-order interpolation of A is enough to achieve the order of six accuracy.

Evaluating $A_{i+1,j+1}$ from Eq. (4.17) gives

$$A_{i+1,j+1} = \frac{1}{h^4}[u^*_{i+1,j+1} - u^h_{i+1,j+1} + O(h^6)]. \tag{4.22}$$

Since the fine grid points $(i+1, j+1)$ $(i = odd, j = odd)$ are $(even, even)$ fine grid points with sixth-order solutions, we have

$$u^*_{i+1,j+1} = \tilde{u}^h_{i+1,j+1} + O(h^6). \tag{4.23}$$

Substituting Eq. (4.23) into Eq. (4.22), we obtain

$$A_{i+1,j+1} = \frac{1}{h^4}[\tilde{u}^h_{i+1,j+1} - u^h_{i+1,j+1} + O(h^6)]. \tag{4.24}$$

Similarly, for $A_{i+1,j-1}$, $A_{i-1,j+1}$, and $A_{i-1,j-1}$ from Eqs. (4.18), (4.19) and (4.20) we obtain

$$A_{i+1,j-1} = \frac{1}{h^4}[\tilde{u}^h_{i+1,j-1} - u^h_{i+1,j-1} + O(h^6)], \tag{4.25}$$

$$A_{i-1,j+1} = \frac{1}{h^4}[\tilde{u}^h_{i-1,j+1} - u^h_{i-1,j+1} + O(h^6)], \tag{4.26}$$

$$A_{i-1,j-1} = \frac{1}{h^4}[\tilde{u}^h_{i-1,j-1} - u^h_{i-1,j-1} + O(h^6)]. \tag{4.27}$$

Using Eqs. (4.24) $\sim$ (4.27) in Eq. (4.21) gives

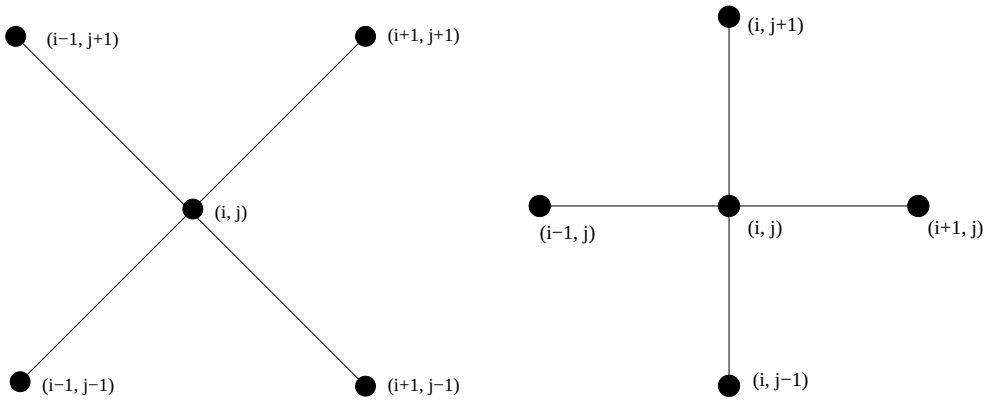
$$\begin{aligned} A_{i,j} = \frac{1}{4h^4}[&(\tilde{u}^h_{i+1,j+1} - u^h_{i+1,j+1}) + (\tilde{u}^h_{i+1,j-1} - u^h_{i+1,j-1}) \\ &+ (\tilde{u}^h_{i-1,j+1} - u^h_{i-1,j+1}) + (\tilde{u}^h_{i-1,j-1} - u^h_{i-1,j-1}) + O(h^6)]. \end{aligned} \tag{4.28}$$

Substituting Eq. (4.28) into Eq. (4.16) gives

$$\begin{aligned} u^*_{i,j} = u^h_{i,j} + \frac{1}{4}[&(\tilde{u}^h_{i+1,j+1} - u^h_{i+1,j+1}) + \tilde{u}^h_{i+1,j-1} - u^h_{i+1,j-1}) \\ &+ (\tilde{u}^h_{i-1,j+1} - u^h_{i-1,j+1}) + (\tilde{u}^h_{i-1,j-1} - u^h_{i-1,j-1})] + O(h^6). \end{aligned} \tag{4.29}$$

Since we have Eq. (4.15) for $(even, even)$ fine grid points, we can compute sixth-order solutions for (odd, odd) fine grid points by Eq. (4.29) as

$$\tilde{u}^h_{i,j} = u^h_{i,j} + c^h_{i,j}, \quad i = odd, j = odd \tag{4.30}$$



(a) Rotated grid interpolation scheme. (b) Standard grid interpolation scheme.

Figure 4.1: Illustration of the interpolation strategy in 2D.

where

$$c_{i,j}^h = \frac{1}{4}(c_{i+1,j+1}^h + c_{i+1,j-1}^h + c_{i-1,j+1}^h + c_{i-1,j-1}^h).$$

By now, we have obtained corrections from the fourth-order solution to the sixth-order solution for $(even, even)$ and (odd, odd) fine grid points, which can be viewed as fourth-order accurate error estimators. We use them to generate fourth-order accurate error estimators for $(even, odd)$ and $(odd, even)$ fine grid points.

By using standard grid interpolation on smooth solutions, as Fig. 4.1(b), we have

$$A_{i,j} = \frac{1}{4}(A_{i+1,j} + A_{i-1,j} + A_{i,j+1} + A_{i,j-1}) + O(h^2) \tag{4.31}$$

for $(odd, even)$ and $(even, odd)$ fine grid points.

Analogously, we can generate correction terms to obtain the computation formula for sixth-order solutions as

$$\begin{aligned} \tilde{u}_{i,j}^h = u_{i,j}^h + c_{i,j}^h, \quad & i = odd, j = even \\ & i = even, j = odd \end{aligned} \tag{4.32}$$

where

$$c_{i,j}^h = \frac{1}{4}(c_{i+1,j}^h + c_{i-1,j}^h + c_{i,j+1}^h + c_{i,j-1}^h).$$

Since another two grid interpolations are involved, i.e., Eqs. (4.21) and (4.31), the error on the (odd, odd), $(odd, even)$ and $(even, odd)$ fine grid points will be larger than that on the $(even, even)$ fine grid points, though still be $O(h^6)$.

4.3 Extension to 3D Problems

For 3D problems, we have the Richardson extrapolation formula

$$\tilde{u}^{2h}_{i,j,k} = \frac{(16u^h_{2i,2j,2k} - u^{2h}_{i,j,k})}{15} \tag{4.33}$$

to compute the sixth-order solutions at $(even, even, even)$ fine grid points. We can analogously use the correction between the fourth-order solution and the extrapolated sixth-order solution at $(even, even, even)$ fine grid points to construct the fourth-order error estimators for other groups of fine grid points. Thus, we have the following formulas for the computation of the sixth-order solution as

$$\tilde{u}^h_{i,j,k} = u^h_{i,j,k} + c^h_{i,j,k} \tag{4.34}$$

where

$$\begin{aligned}
c^h_{i,j,k} &= \frac{1}{15}(u^h_{2i,2j,2k} - u^{2h}_{i,j,k}), && i = even, j = even, k = even \\
c^h_{i,j,k} &= \frac{1}{2}(c^h_{i,j,k-1} + c_{i,j,k+1}), && i = even, j = even, k = odd \\
c^h_{i,j,k} &= \frac{1}{2}(c^h_{i,j-1,k} + c_{i,j+1,k}), && i = even, j = odd, k = even \\
c^h_{i,j,k} &= \frac{1}{2}(c^h_{i-1,j,k} + c_{i+1,j,k}), && i = odd, j = even, k = even \\
c^h_{i,j,k} &= \frac{1}{4}(c^h_{i,j-1,k-1} + c^h_{i,j-1,k+1} + c^h_{i,j+1,k-1} + c^h_{i,j+1,k+1}), && i = even, j = odd, k = odd \\
c^h_{i,j,k} &= \frac{1}{4}(c^h_{i-1,j,k-1} + c^h_{i-1,j,k+1} + c^h_{i+1,j,k-1} + c^h_{i+1,j,k+1}), && i = odd, j = even, k = odd \\
c^h_{i,j,k} &= \frac{1}{4}(c^h_{i-1,j-1,k} + c^h_{i-1,j+1,k} + c^h_{i+1,j-1,k} + c^h_{i+1,j+1,k}), && i = odd, j = odd, k = even \\
c^h_{i,j,k} &= \frac{1}{8}(c^h_{i-1,j-1,k-1} + c^h_{i-1,j-1,k+1} + c^h_{i-1,j+1,k-1} \\
&\quad + c^h_{i-1,j+1,k+1} + c^h_{i+1,j-1,k-1} + c^h_{i+1,j-1,k+1} \\
&\quad + c^h_{i+1,j+1,k-1} + c^h_{i+1,j+1,k+1}). && i = odd, j = odd, k = odd
\end{aligned}$$

4.4 Numerical Results

In this section, we tested the sixth-order method with completed Richardson extrapolation and compared it with Wang-Zhang's sixth-order method [79, 80] in accuracy and efficiency experimentally. The codes were written in Fortran 77 programming language and run on one login grid point of Lipscomb HPC Cluster at the University of Kentucky. The grid point has Dual Intel E5-2670 8 Core (totally 16 cores) with 2.6GHz and 128GB RAM.

Since the proposed sixth-order method is based on Richardson extrapolation, it can also be integrated into the MSMG computation for the purpose of high computational efficiency. We used standard V(1,1)-cycle algorithm in the MSMG computation. The initial guess for the V-cycle on Ω_{4h} was the zero vector. The V-cycles on Ω_{2h} and Ω_h stopped when the L^2-norm of the difference of the successive solutions was reduced by a factor of 10^{10}. The iterative operator based interpolation procedure stopped when the L^2-norm of the correction vector of the approximate solution was less than 10^{-10}. All of the errors reported were the maximum absolute errors over the finest grid.

4.4.1 Test problems

We chose a 2D Poisson equation and a 3D convection-diffusion equation with small Reynolds number (Re) as test problems. For the 2D test problem, the 9-point FOC scheme (2.9) was used to compute fourth-order solutions on different scaled grids. For the 3D test problem, the 19-point finite difference scheme (3.7) was used to compute different discretized fourth-order solutions.

Problem 1.

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = 2\pi^2 \sin(\pi x)\cos(\pi y), \quad (x, y) \in \Omega = [0, 4] \times [0, 1], \tag{4.35}$$

which has the Dirichlet boundary condition.

The analytical solution is

$$u(x, y) = \sin(\pi x)\cos(\pi y).$$

Problem 2.

$$u_{xx} + u_{yy} + u_{zz} + p(x, y, z)u_x + q(x, y, z)u_y + r(x, y, z)u_z = f(x, y, z),$$
$$(x, y, z) \in \Omega = [0, 1] \times [0, 1] \times [0, 1], \quad (4.36)$$

where the coefficients of Eq. (4.36) are set as

$$p(x, y, z) = q(x, y, z) = r(x, y, z) = Re.$$

The analytical solution is

$$u(x, y, z) = \cos(4x + 6y + 8z).$$

4.4.2 Accuracy and efficiency

In order to test accuracy and efficiency of the proposed method, we refined the grid from $N = 32$ to $N = 256$ for Problem 1 and from $N = 16$ to $N = 128$ for Problem 2, where N is the number of intervals in one coordinate direction. For convenience, in the comparison between the two Richardson extrapolation-based sixth-order methods, we use the following abbreviations: "Op-Six" is short for Wang-Zhang's sixth-order method with Richardson extrapolation and iterative operator based interpolation [79, 80]; "CR-Six" denotes the present sixth-order method with completed Richardson extrapolation.

Maximum errors, the computed accuracy order and CPU time in seconds are listed in Table 4.1. The solutions from both methods can achieve sixth-order accuracy. The column error shows that, for both test problems, the computed solutions from the CR-Six method are slightly more accurate than that from the Op-Six method. Table 4.1 also shows that the CR-Six method requires less CPU time than the Op-Six method to

Table 4.1: Numerical comparison between the sixth-order method with Richardson extrapolation and iterative operator based interpolation and the sixth-order method with completed Richardson extrapolation

Test Problem 1	N	Error	Order	CPU(s)
Op-Six	32	2.498e-6	-	0.007
	64	4.582e-8	5.77	0.031
	128	7.662e-10	5.90	0.124
	256	1.234e-11	5.96	0.491
CR-Six	32	8.927e-7	-	0.004
	64	1.362e-8	6.03	0.022
	128	2.105e-10	6.02	0.090
	256	3.270e-12	6.01	0.393
Test Problem 2 (Re=10)	N	Error	Order	CPU(s)
Op-Six	16	3.547e-4	-	0.021
	32	9.234e-6	5.26	0.159
	64	1.764e-7	5.71	0.907
	128	3.062e-9	5.85	5.576
CR-Six	16	1.595e-4	-	0.013
	32	2.517e-6	5.99	0.108
	64	3.842e-8	6.03	0.732
	128	9.324e-10	5.36	3.872

compute solutions with comparable accuracy. The higher computational efficiency of the CR-Six method is due to the avoidance of using the iterative refinement procedure appeared in the Op-Six method, which has a low convergence rate and thus takes a certain amount of CPU time.

4.5 Concluding Remarks

We presented the sixth-order compact approximation with completed Richardson extrapolation and compared it with an existing Richardson extrapolation-based sixth-order method with iterative operator based interpolation. With respect to accuracy, the proposed method is able to obtain sixth-order solutions with smaller errors. As expected, the computational efficiency of the proposed method is higher by eliminating the iterative refinement procedure on the finest grid.

5 Analysis and Comparison of Richardson Extrapolation-based Sixth-Order Methods

5.1 Introduction

Until now, we have learned that Richardson extrapolation can improve the solution accuracy of PDEs by using approximate solutions from two different scale grids. To explicitly obtain a sixth-order solution using Richardson extrapolation, we need to have two computed fourth-order solutions on the coarse and fine grids, respectively. For this purpose, fourth-order compact schemes and multigrid methods are typically used. Then, the Richardson extrapolation technique can be applied to compute a sixth-order solution on the coarse grid. Other techniques are needed to obtain a sixth-order solution on the fine grid. In this dissertation, we have discussed three techniques for computing fine grid sixth-order solutions (operator based interpolation, multiple coarse grid (MCG) updating strategy, and completed Richardson extrapolation), which lead to three kinds of Richardson extrapolation-based sixth-order methods. In this chapter, we will analyze the truncation error terms of these three methods for solving a 2D Poisson equation, and thus compare their accuracy theoretically. Numerical experiments of several test problems are also conducted.

5.2 Truncation Error Analysis

Consider the 2D Poisson equation of the form

$$u_{xx}(x,y) + u_{yy}(x,y) = f(x,y), \qquad (x,y) \in \Omega, \tag{5.1}$$

where Ω is a rectangular domain, with suitable boundary conditions defined on $\partial\Omega$. The solution $u(x,y)$ and the forcing function $f(x,y)$ are assumed to be sufficiently smooth and have required continuous partial derivatives.

All the Richardson extrapolation-based sixth-order methods use the FOC scheme (2.8) to compute fourth-order solutions of Eq. (5.1) on two level discretized grids.

For illustration, we first derive the truncation error of the FOC scheme with unequal mesh-size discretization for solving Eq. (5.1).

Denote Δx and Δy to be the mesh-sizes in the x and y coordinate directions, respectively. The standard second-order central difference operators are

$$\delta_x^2 u_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}, \quad \delta_y^2 u_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}.$$

By using Taylor series, we have

$$\delta_x^2 u_{i,j} = u_{xx} + \frac{\Delta x^2}{12} u_{x^4} + \frac{\Delta x^4}{360} u_{x^6} + \frac{\Delta x^6}{20160} u_{x^8} + O(\Delta x^8), \tag{5.2}$$

and

$$\delta_y^2 u_{i,j} = u_{yy} + \frac{\Delta y^2}{12} u_{y^4} + \frac{\Delta y^4}{360} u_{y^6} + \frac{\Delta y^6}{20160} u_{y^8} + O(\Delta y^8). \tag{5.3}$$

From Eqs. (5.2) and (5.3) we can discretize Eq. (5.1) at the grid point $x_{i,j}$ as

$$\begin{aligned} \delta_x^2 u_{i,j} + \delta_y^2 u_{i,j} = f_{i,j} &+ \frac{1}{12}(\Delta x^2 u_{x^4} + \Delta y^2 u_{y^4}) \\ &+ \frac{1}{360}(\Delta x^4 u_{x^6} + \Delta y^4 u_{y^6}) + \frac{1}{20160}(\Delta x^6 u_{x^8} + \Delta y^6 u_{y^8}) + O(\Delta^8). \end{aligned} \tag{5.4}$$

By taking two times partial derivatives of x and y on both sides of Eq. (5.1), respectively, we have

$$u_{x^4} = f_{xx} - u_{yyxx}, \tag{5.5}$$

and

$$u_{y^4} = f_{xx} - u_{xxyy}. \tag{5.6}$$

Using central difference operators and Taylor series in Eqs. (5.5) and (5.6) gives

$$\begin{aligned} (u_{x^4})_{i,j} = \delta_x^2 f_{i,j} &- \frac{1}{\Delta y^2}(\delta_x^2 u_{i,j+1} - 2\delta_x^2 u_{i,j} + \delta_x^2 u_{i,j-1}) \\ &- \frac{\Delta x^2}{12} f_{x^4} - \frac{\Delta x^4}{360} f_{x^6} - \frac{1}{\Delta y^2}(-\frac{\Delta x^2}{12}(\Delta y^2 u_{x^4y^2} + \frac{\Delta y^4}{12} u_{x^4y^4}) - \frac{\Delta x^4}{360}\Delta y^2 u_{x^6y^2}) \\ &+ \frac{\Delta y^2}{12} u_{x^2y^4} + \frac{\Delta y^4}{360} u_{x^2y^6} + O(\Delta^6), \end{aligned} \tag{5.7}$$

and

$$(u_{y^4})_{i,j} = \delta_y^2 f_{i,j} - \frac{1}{\Delta x^2}(\delta_y^2 u_{i+1,j} - 2\delta_y^2 u_{i,j} + \delta_y^2 u_{i-1,j}) \\ - \frac{\Delta y^2}{12} f_{y^4} - \frac{\Delta y^4}{360} f_{y^6} - \frac{1}{\Delta x^2}(-\frac{\Delta y^2}{12}(\Delta x^2 u_{x^2y^4} + \frac{\Delta x^4}{12} u_{x^4y^4}) - \frac{\Delta y^4}{360}\Delta x^2 u_{x^2y^6}) \\ + \frac{\Delta x^2}{12} u_{x^4y^2} + \frac{\Delta x^4}{360} u_{x^6y^2} + O(\Delta^6). \tag{5.8}$$

By continuously taking partial derivatives of x on both sides of Eq. (5.5), we have

$$f_{x^4} = u_{x^6} + u_{x^4y^2}, \tag{5.9}$$

$$f_{x^6} = u_{x^8} + u_{x^6y^2}. \tag{5.10}$$

Similarly, by continuously taking partial derivatives of y on both sides of Eq. (5.6), we have

$$f_{y^4} = u_{y^6} + u_{x^2y^4}, \tag{5.11}$$

$$f_{y^6} = u_{y^8} + u_{x^2y^6}. \tag{5.12}$$

Substituting Eqs. (5.9) and (5.10) in Eq. (5.7) gives

$$(u_{x^4})_{i,j} = \delta_x^2 f_{i,j} - \frac{1}{\Delta y^2}(\delta_x^2 u_{i,j+1} - 2\delta_x^2 u_{i,j} + \delta_x^2 u_{i,j-1}) \\ - \frac{\Delta x^2}{12} u_{x^6} - \frac{\Delta y^2}{12} u_{x^2y^4} + \frac{\Delta x^2 \Delta y^2}{144} u_{x^4y^4} - \frac{\Delta x^4}{360} u_{x^8} - \frac{\Delta y^4}{360} u_{x^2y^6} + O(\Delta^6). \tag{5.13}$$

And, substituting Eqs. (5.11) and (5.12) in Eq. (5.8) gives

$$(u_{y^4})_{i,j} = \delta_y^2 f_{i,j} - \frac{1}{\Delta x^2}(\delta_y^2 u_{i+1,j} - 2\delta_y^2 u_{i,j} + \delta_y^2 u_{i-1,j}) \\ - \frac{\Delta y^2}{12} u_{y^6} - \frac{\Delta x^2}{12} u_{x^4y^2} + \frac{\Delta x^2 \Delta y^2}{144} u_{x^4y^4} - \frac{\Delta y^4}{360} u_{y^8} - \frac{\Delta x^4}{360} u_{x^6y^2} + O(\Delta^6). \tag{5.14}$$

Then, using Eqs. (5.13) and (5.14) to replace the u_{x^4} and u_{y^4} terms in Eq. (5.4) gives

$$\delta_x^2 u_{i,j} + \delta_y^2 u_{i,j} = f_{i,j} + \frac{1}{12}(\Delta x^2 \delta_x^2 f_{i,j} + \Delta y^2 \delta_y^2 f_{i,j}) \\ - \frac{1}{12}(\frac{\Delta x^2}{\Delta y^2}(\delta_x^2 u_{i,j+1} - 2\delta_x^2 u_{i,j} + \delta_x^2 u_{i,j-1}) + \frac{\Delta y^2}{\Delta x^2}(\delta_y^2 u_{i+1,j} - 2\delta_y^2 u_{i,j} \\ + \delta_y^2 u_{i-1,j})) + (\tau_4)_{i,j} + (\tau_6)_{i,j} + O(\Delta^8), \tag{5.15}$$

where

$$(\tau_4)_{i,j} = \frac{1}{144}(u_{x^4y^2} + u_{x^2y^4})\Delta x^2\Delta y^2 - \frac{1}{240}(u_{x^6}\Delta x^4 + u_{y^6}\Delta y^4),$$

$$\begin{aligned}(\tau_6)_{i,j} = &\frac{1}{1728}(\Delta x^4\Delta y^2 + \Delta x^2\Delta y^4)u_{x^4y^4} + \frac{1}{4320}(\Delta x^2\Delta y^4 u_{x^2y^6} + \Delta x^4\Delta y^2 u_{x^6y^2}) \\ &- \frac{11}{60480}(\Delta x^6 u_{x^8} + \Delta y^6 u_{y^8}).\end{aligned}$$

Let us use the second-order central difference operators in Eq. (5.15) and multiply $6\Delta x^2$ on both sides, and denote the mesh aspect ratio $\lambda = \frac{\Delta x}{\Delta y}$, we obtain a general FOC scheme like the one presented in [93] as

$$\begin{aligned}&m_1(u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1}) + m_2(u_{i,j+1} + u_{i,j-1}) \\ &+ m_3(u_{i+1,j} + u_{i-1,j}) - m_4 u_{i,j} \\ &= \frac{\Delta x^2}{2}(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}),\end{aligned} \tag{5.16}$$

where the coefficients are

$$m_1 = (1+\lambda^2)/2, \quad m_2 = 5\lambda^2 - 1, \quad m_3 = 5 - \lambda^2, \quad m_4 = 10(1+\lambda^2).$$

The fourth-order truncation error of the FOC scheme (5.16) is

$$\tilde{\tau_4} = \{\frac{1}{24\lambda^2}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + \frac{u_{y^6}}{\lambda^4})\}\Delta x^4. \tag{5.17}$$

And, the sixth-order truncation error of the FOC scheme (5.16) is

$$\begin{aligned}\tilde{\tau_6} = \{&\frac{1}{288}(\frac{1}{\lambda^2} + \frac{1}{\lambda^4})u_{x^4y^4} + \frac{1}{720}(\frac{u_{x^2y^6}}{\lambda^4} + \frac{u_{x^6y^2}}{\lambda^2}) \\ &- \frac{11}{10080}(u_{x^8} + \frac{u_{y^8}}{\lambda^6})\}\Delta x^6.\end{aligned} \tag{5.18}$$

Consider a special case with $\Delta x = \Delta y = h$, the FOC scheme has the form as

$$\begin{aligned}&u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} + 4(u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j}) - 20u_{i,j} \\ &= \frac{h^2}{2}(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}).\end{aligned} \tag{5.19}$$

The fourth-order and sixth-order truncation errors of the FOC scheme (5.19) are

$$\tau_{FOC4} = \{\frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + u_{y^6})\}h^4, \tag{5.20}$$

$$\tau_{FOC6} = \{\frac{1}{144}u_{x^4y^4} + \frac{1}{720}(u_{x^2y^6} + u_{x^6y^2}) - \frac{11}{10080}(u_{x^8} + u_{y^8})\}h^6. \tag{5.21}$$

Now we can take a look at the truncation error after applying Richardson extrapolation. From the definition of the fourth-order solutions on the fine and coarse grids, we have

$$u_h^* = u_h^4 + \tau_{FOC4} + \tau_{FOC6}, \tag{5.22}$$

$$u_{2h}^* = u_{2h}^4 + 16\tau_{FOC4} + 64\tau_{FOC6}. \tag{5.23}$$

Using the Richardson extrapolation formula (4.13) gives

$$u_{2h}^* = u_{2h}^6 - \frac{16}{5}\tau_{FOC6}. \tag{5.24}$$

Thus, the sixth-order truncation error after applying Richardson extrapolation has the form as

$$\tau_{Extrapo} = -\frac{16}{5}\tau_{FOC6}. \tag{5.25}$$

For all Richardson extrapolation-based sixth-order compact approximations, Richardson extrapolation is always used to obtain the sixth-order solution on the standard coarse grid and the extrapolated solution is directly interpolated to the corresponding $(even, even)$ fine grid points. Therefore, the truncation error of $(even, even)$ fine grid points is $\tau_{Extrapo}$. For (odd, odd), $(even, odd)$ and $(odd, even)$ fine grid points, three computational strategies (iterative operator based interpolation, MCG updaitng strategy, and completed Richardson extrapolation) are used to obtain sixth-order solutions. In the following part, truncation error analysis for these three strategies are given.

5.2.1 Truncation error of iterative operator based interpolation

In order to obtain sixth-order solutions for the remaining fine grid points, Wang and Zhang [79, 80] proposed an operator based interpolation scheme to iteratively update

the fine grid point solutions in a specific sequence until some convergence condition is satisfied. The operator based interpolation for the 2D Poisson equation (5.1) can be obtained from Eq. (5.19) as

$$\tilde{u}^h_{i,j} = -\frac{1}{20}[F_{i,j} - 4(u^h_{i+1,j} + u^h_{i-1,j} + u^h_{i,j+1} + u^h_{i,j-1}) - (u^h_{i+1,j+1} + u^h_{i+1,j-1} + u^h_{i-1,j+1} + u^h_{i-1,j-1})], \tag{5.26}$$

where $F_{i,j} = 8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}$.

The leading truncation error of Eq. (5.26) comes from τ_{FOC4} and has the form as

$$\tau_{op} = \tau_{FOC4}\frac{h^2}{20} = \frac{1}{20}\{\frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + u_{y^6})\}h^6. \tag{5.27}$$

The operator based interpolation Eq. (5.26) can be written as

$$\begin{aligned} u^*_{i,j} &= \tilde{u}_{i,j} + \tau_{Op} \\ &= -\frac{1}{20}[F_{i,j} - 4(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) \\ &\quad - (u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1})] + \tau_{op}. \end{aligned} \tag{5.28}$$

In order to find the truncation error of other three groups of fine grid points after the iterative refinement procedure with the operator based interpolation scheme (5.26), we assume the truncation error of (odd, odd), $(odd, even)$ and $(even, odd)$ fine grid points as α_{op}, β_{op} and γ_{op}, respectively. A system on the errors of different groups of fine grid points is generated through Eq. (5.28) as

$$\begin{cases} 20\alpha_{op} - 4(\gamma_{op} + \gamma_{op} + \beta_{op} + \beta_{op}) - 4\tau_{Extrapo} = 20\tau_{op}, & \text{i=odd, j=odd} \\ 20\beta_{op} - 4(\tau_{Extrapo} + \tau_{Extrapo} + \alpha_{op} + \alpha_{op}) - 4\gamma_{op} = 20\tau_{op}, & \text{i=odd, j=even} \\ 20\gamma_{op} - 4(\alpha_{op} + \alpha_{op} + \tau_{Extrapo} + \tau_{Extrapo}) - 4\beta_{op} = 20\tau_{op}. & \text{i=even, j=odd} \end{cases} \tag{5.29}$$

From Eq. (5.29), we get

$$\begin{cases} \alpha_{op} = \tau_{Extrapo} + \frac{10}{3}\tau_{op}, & \text{i=odd, j=odd} \\ \beta_{op} = \tau_{Extrapo} + \frac{35}{12}\tau_{op}, & \text{i=odd, j=even} \\ \gamma_{op} = \tau_{Extrapo} + \frac{35}{12}\tau_{op}. & \text{i=even, j=odd} \end{cases} \tag{5.30}$$

5.2.2 Truncation error of MCG updating strategy

In the MCG updating strategy for solving the 2D Poisson equation [16], the X-odd and Y-odd grid views are constructed to compute sixth-order solutions for $(odd, even)$ and $(even, odd)$ fine grid points, respectively. The X-odd grid view, composed by $(even, even)$ and $(odd, even)$ fine grid points, is a view of unequal mesh-size grid with mesh-sizes h and $2h$ in the x and y coordinate directions, respectively. The Y-odd grid view, composed by $(even, even)$ and $(even, odd)$ fine grid points, is a view of unequal mesh-size grid with mesh-sizes $2h$ and h in the x and y coordinate directions, respectively. The sixth-order computations on the X-odd grid view and the Y-odd grid view by solving tridiagonal systems lead to sixth-order truncation errors τ_{x-odd} and τ_{y-odd}, respectively. By using the general fourth-order truncation error expressed by Eq. (5.17) and setting corresponding mesh aspect ratio λ, we have an explicit form of τ_{x-odd} and τ_{y-odd} as

$$\begin{cases} \tau_{x-odd} = \{4 \times \frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + 16 \times \ u_{y^6})\}h^6, & \lambda_{x-odd} = \frac{1}{2} \\ \tau_{y-odd} = \frac{1}{16} \times \{4 \times \frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(16 \times u_{x^6} + u_{y^6})\}(2h)^6. & \lambda_{y-odd} = 2 \end{cases} \tag{5.31}$$

For the computation of $(odd, even)$ fine grid points on the X-odd grid view, the mesh aspect ratio $\lambda_{x-odd} = \frac{1}{2}$ and the coefficients in Eq. (5.16) are set as

$$m_1 = \tfrac{5}{8}, \quad m_2 = \tfrac{1}{4}, \quad m_3 = \tfrac{19}{4}, \quad m_4 = \tfrac{50}{4}.$$

Denote the truncation error of $(odd, even)$ fine grid points as α_{mcg}. An equation on the error of X-odd grid view points, not the solution, is generated by Eq. (5.16) with above coefficients as

$$\frac{5}{8} \times 4\tau_{Extrapo} + \frac{1}{4} \times (\alpha_{mcg} + \alpha_{mcg}) + \frac{19}{4} \times (\tau_{Extrapo} + \tau_{Extrapo}) - \frac{50}{4}\alpha_{mcg} = -\tau_{x-odd}. \tag{5.32}$$

From Eq. (5.32), we get

$$\alpha_{mcg} = \tau_{Extrapo} + \frac{\tau_{x-odd}}{12}. \tag{5.33}$$

For the computation of $(even, odd)$ fine grid points on the Y-odd grid view, the mesh aspect ratio $\lambda_{y-odd} = 2$ and the coefficients in Eq. (5.16) are set as

$$m_1 = \tfrac{5}{2}, \quad m_2 = 19, \quad m_3 = 1, \quad m_4 = 50.$$

Denote the truncation error of $(even, odd)$ fine grid points as β_{mcg}. An equation on the error of Y-odd grid view points, not the solution, by using the Eq. (5.16) with above coefficients is generated as

$$\frac{5}{2} \times 4\tau_{Extrapo} + 19 \times (\tau_{Extrapo} + \tau_{Extrapo}) + 1 \times (\beta_{mcg} + \beta_{mcg}) - 50\beta_{mcg} = -\tau_{y-odd}. \quad (5.34)$$

From Eq. (5.34), we get

$$\beta_{mcg} = \tau_{Extrapo} + \frac{\tau_{y-odd}}{48}. \quad (5.35)$$

The update of (odd, odd) fine grid points uses the operator based interpolation Eq.(5.26) and $(even, even)$, $(odd, even)$ and $(even, odd)$ fine grid points with sixth-order solutions. Denote the truncation error of (odd, odd) fine grid points as γ_{mcg}. An equation on the error of fine grid points is generated by Eq. (5.28) as

$$4\tau_{Extrapo} + 4 \times (\alpha_{mcg} + \alpha_{mcg}) + 4 \times (\beta_{mcg} + \beta_{mcg}) - 20\gamma_{mcg} = -20\tau_{op}. \quad (5.36)$$

From Eq. (5.36), we get

$$\gamma_{mcg} = \tau_{Extrapo} + \frac{\tau_{x-odd}}{30} + \frac{\tau_{y-odd}}{120} + \tau_{op}. \quad (5.37)$$

5.2.3 Truncation error of completed Richardson extrapolation

Completed Richardson extrapolation uses the correction between the fourth-order solution and the extrapolated sixth-order solution to obtain a sixth-order solution on the entire fine grid [15]. In the sixth-order method with completed Richardson extrapolation for 2D problems, two kinds of second-order interpolations are used to approximate the fourth-order error terms. The rotated grid interpolation Eq. (4.21) is used for the (odd, odd) fine grid points and the standard grid interpolation Eq.

(4.31) is used for the $(odd, even)$ and $(even, odd)$ fine grid points. The coefficients As in Eqs. (4.21) and (4.31) can be viewed as a function of u which has the form of $A(u) = \tau_{FOC4}/h^4$. Based on the Taylor series, the $O(h^2)$ term in Eq.(4.21) has an explicit form as $\frac{2h^2}{4}(\frac{\partial^2(\tau_{FOC4}/h^4)}{\partial x^2} + \frac{\partial^2(\tau_{FOC4}/h^4)}{\partial y^2})$, and the $O(h^2)$ term in Eq.(4.31) has an explicit form as $\frac{h^2}{4}(\frac{\partial^2(\tau_{FOC4}/h^4)}{\partial x^2} + \frac{\partial^2(\tau_{FOC4}/h^4)}{\partial y^2})$.

Therefore, the second-order truncation error of rotated grid interpolation Eq. (4.21) is

$$\tau_{RotateInter} = \{\frac{1}{24}u_{x^4y^4} + \frac{1}{120}(u_{x^6y^2} + u_{x^2y^6}) - \frac{1}{80}(u_{x^8} + u_{y^8})\}h^2.$$

The second-order truncation error of standard grid interpolation Eq. (4.31) is

$$\tau_{StandInter} = \{\frac{1}{48}u_{x^4y^4} + \frac{1}{240}(u_{x^6y^2} + u_{x^2y^6}) - \frac{1}{160}(u_{x^8} + u_{y^8})\}h^2. \tag{5.38}$$

And, we find that $\tau_{RotateInter} = 2\tau_{StandInter}$.

First, consider (odd, odd) fine grid points. Eq. (4.21) can be re-written as

$$A_{i,j} = \frac{1}{4}(A_{i+1,j+1} + A_{i+1,j-1} + A_{i-1,j+1} + A_{i-1,j-1}) + \tau_{RotateInter}, \quad i = odd, j = odd \tag{5.39}$$

From Section 4.2, we know that the sixth-order computation for (odd, odd) fine grid points is only related to $(even, even)$ fine grid points. For the $(even, even)$ fine grid points, the definition of fourth-order solution gives

$$A_{even,even} = \frac{1}{h^4}[u^*_{even,even} - u^4_{even,even} - \tau_{FOC6}]. \tag{5.40}$$

After injecting the extrapolated coarse grid solution into the fine grid, we have

$$u^*_{even,even} = u^6_{even,even} + \tau_{Extrapo}. \tag{5.41}$$

Substituting Eq. (5.41) into Eq. (5.40) gives

$$\begin{aligned} A_{even,even} &= \frac{1}{h^4}[u^6_{even,even} - u^4_{even,even} - \tau_{FOC6} + \tau_{Extrapo}] \\ &= \frac{1}{h^4}[c_{even,even} - \tau_{FOC6} + \tau_{Extrapo}]. \end{aligned} \tag{5.42}$$

By using Eqs. (4.30), (5.39) and (5.42), we have the truncation error of (odd, odd) fine grid points as

$$\begin{aligned}
\tau_{CompEx1} &= u^*_{i,j} - u^6_{i,j} \\
&= (u^4_{i,j} + A_{i,j}h^4 + \tau_{FOC6}) - (u^4_{i,j} + \frac{1}{4}(c_{i+1,j+1} + c_{i+1,j-1} + c_{i-1,j+1} + c_{i-1,j-1})) \\
&= (u^4_{i,j} + A_{i,j}h^4 + \tau_{FOC6}) - (u^4_{i,j} + \frac{1}{4}(A_{i+1,j+1} + A_{i+1,j-1} + A_{i-1,j+1} \\
&\quad + A_{i-1,j-1})h^4 + \tau_{FOC6} - \tau_{Extrapo}) \\
&= \tau_{RotateInter}h^4 + \tau_{Extrapo}. \qquad i = odd, j = odd
\end{aligned} \tag{5.43}$$

Next, consider $(odd, even)$ and $(even, odd)$ fine grid points. Eq. (4.31) can be re-written as

$$A_{i,j} = \frac{1}{4}(A_{i+1,j} + A_{i-1,j} + A_{i,j+1} + A_{i,j-1}) + \tau_{StandInter}. \quad \begin{aligned} &i = odd, j = even \\ &i = even, j = odd \end{aligned} \tag{5.44}$$

The sixth-order computation for $(odd, even)$ and $(even, odd)$ fine grid points are related to both $(even, even)$ and (odd, odd) fine grid points.
For the updated (odd, odd) fine grid points with sixth-order solutions, we have

$$u^*_{odd,odd} = u^6_{odd,odd} + \tau_{CompEx1} = u^6_{odd,odd} + \tau_{RotateInter}h^4 + \tau_{Extrapo}. \tag{5.45}$$

By using the definition of fourth-order solution for (odd, odd) fine grid points, we have

$$A_{odd,odd} = \frac{1}{h^4}[u^*_{odd,odd} - u^4_{odd,odd} - \tau_{FOC6}]. \tag{5.46}$$

Substituting Eq. (5.45) into Eq. (5.46) gives

$$\begin{aligned}
A_{odd,odd} &= \frac{1}{h^4}[u^6_{odd,odd} - u^4_{odd,odd} - \tau_{FOC6} + \tau_{RotateInter}h^4 + \tau_{Extrapo}] \\
&= \frac{1}{h^4}[c_{odd,odd} - \tau_{FOC6} + \tau_{RotateInter}h^4 + \tau_{Extrapo}].
\end{aligned} \tag{5.47}$$

By using Eqs. (4.32), (5.42), (5.44) and (5.47), we obtain the truncation error of

$(even, odd)$ and $(odd, even)$ fine grid points as

$$\begin{aligned}
\tau_{CompEx2} &= u^*_{i,j} - u^6_{i,j} \\
&= (u^4_{i,j} + A_{i,j}h^4 + \tau_{FOC6}) - (u^4_{i,j} + \frac{1}{4}(c_{i+1,j} + c_{i-1,j} + c_{i,j+1} + c_{i,j-1})) \\
&= (u^4_{i,j} + A_{i,j}h^4 + \tau_{FOC6}) - (u^4_{i,j} + \frac{1}{4}(A_{i+1,j} + A_{i-1,j} + A_{i,j+1} \\
&\quad + A_{i,j-1})h^4 + \tau_{FOC6} - \frac{1}{2}\tau_{RotateInter}h^4 - \tau_{Extrapo}) \\
&= \tau_{StandInter}h^4 + \frac{1}{2}\tau_{RotateInter}h^4 + \tau_{Extrapo} \\
&= \tau_{RotateInter}h^4 + \tau_{Extrapo}. \qquad i = odd, j = even \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = even, j = odd
\end{aligned} \tag{5.48}$$

We find that the truncation errors at (odd, odd), $(odd, even)$ and $(even, odd)$ fine grid points share the same form as $\tau_{RotateInter}h^4 + \tau_{Extrapo}$, which is larger than the truncation error of $(even, even)$ fine grid points $\tau_{Extrapo}$ directly generated from Richardson extrapolation as we expect. It is because extra interpolations are involved, i.e., Eqs. (5.43) and (5.48).

In summary, all the three Richardson extrapolation-based methods are able to compute sixth-order accurate solutions for all fine grid points. The differences on accuracy among these methods are truncation errors at $(odd, even)$, $(even, odd)$ and (odd, odd) fine grid points. For $(even, even)$ fine grid points, Richardson extrapolation is used to compute the sixth-order solution with truncation error $\tau_{Extrapo}$. For other three groups of fine grid points, different computational strategies are applied to obtain sixth-order solutions, which add differernt magnitude error expressions to the truncation error $\tau_{Extrapo}$. Table 5.1 lists the truncation errors of different groups of fine grid points by groups after using three Richardson extrapolation-based methods for sixth-order solution computation, respectively. Since the error expressions involve various high-order partial derivatives on u, it is hard to conclude a quantita-

tive relationship. By comparing the coefficients of common items, we could estimate a possible qualitative relationship. The completed Richardson extrapolation method may have smaller truncation errors than the iterative operator based interpolation method, which may have smaller truncation errors than the MCG updating strategy.

Table 5.1: Truncation errors of three Richardson extrapolation-based sixth-order methods for solving the 2D Poisson equation.

Richardson extrapolation with iterative operator based interpolation	
$(even, even)$ points	$\tau_{Extrapo}$
$(odd, even)$ points	$\tau_{Extrapo} + \frac{7}{48}[\frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + u_{y^6})]\}h^6$
$(even, odd)$ points	$\tau_{Extrapo} + \frac{7}{48}[\frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + u_{y^6})]\}h^6$
(odd, odd) points	$\tau_{Extrapo} + \frac{8}{48}[\frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + u_{y^6})]\}h^6$
Richardson extrapolation with MCG updating strategy	
$(even, even)$ points	$\tau_{Extrapo}$
$(odd, even)$ points	$\tau_{Extrapo} + \frac{1}{12}\{4 \times \frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + 16 \times \ u_{y^6})\}h^6$
$(even, odd)$ points	$\tau_{Extrapo} + \frac{1}{12}\{4 \times \frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(16 \times u_{x^6} + u_{y^6})\}h^6$
(odd, odd) points	$\tau_{Extrapo} + \{\frac{1}{30}[4 \times \frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + 16 \times u_{y^6})]$ $+\frac{1}{30}[4 \times \frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(16 \times u_{x^6} + u_{y^6})]$ $+\frac{1}{20}[\frac{1}{24}(u_{x^4y^2} + u_{x^2y^4}) - \frac{1}{40}(u_{x^6} + u_{y^6})]\}h^6$
Richardson extrapolation with completed Richardson extrapolation	
$(even, even)$ points	$\tau_{Extrapo}$
$(odd, even)$ points	$\tau_{Extrapo} + \{\frac{1}{24}u_{x^4y^4} + \frac{1}{120}(u_{x^6y^2} + u_{x^2y^6}) - \frac{1}{120}(u_{x^8} + u_{y^8})\}h^6$
$(even, odd)$ points	$\tau_{Extrapo} + \{\frac{1}{24}u_{x^4y^4} + \frac{1}{120}(u_{x^6y^2} + u_{x^2y^6}) - \frac{1}{120}(u_{x^8} + u_{y^8})\}h^6$
(odd, odd) points	$\tau_{Extrapo} + \{\frac{1}{24}u_{x^4y^4} + \frac{1}{120}(u_{x^6y^2} + u_{x^2y^6}) - \frac{1}{120}(u_{x^8} + u_{y^8})\}h^6$

5.3 Numerical Results

We tested three Richardson extrapolation-based sixth-order methods and compared the accuracy and efficiency among them. The codes were written in Fortran 77 programming language and run on one login node of Lipscomb HPC Cluster at the University of Kentucky. The node has Dual Intel E5-2670 8 Core (totally 16 cores) with 2.6GHz and 128GB RAM.

The MSMG computation introdued in Section 1.3 is perfectly constructed for Richardson extrapolation-based methods. Therefore, we applied it to compute fourth-order solutions on the fine and coarse grids in all three Richardson extrapolation-based

sixth-order methods. The standard V(1,1)-cycle algorithm was chosen. The initial guess for the V-cycle on Ω_{4h} was the zero vector. In general, the V-cycles on Ω_{2h} and Ω_h stop when the L^2-norm of the difference of the successive solutions is less than 10^{-10}. The iterative operator based interpolation terminates when the L^2-norm of the correction vector of the approximate solution is less than 10^{-10}. However, it may change depends on the test case itself. In this section, for Problem 1, the stopping criteria for V-cycles and the iterative operator based interpolation procedure were set as 10^{-13}. As for Problems 2, 3, and 4, the stopping criteria for all iterative procedures were selected as 10^{-10}. All of the errors reported were the maximum absolute errors over the finest grid.

5.3.1 Test problems

We chose two 2D Poisson equations and two 3D convection-diffusion equations as test problems. The 9-point FOC scheme (2.9) was used to compute fourth-order solutions for the 2D test problems. The 19-point finite difference scheme (3.7) was used to compute fourth-order solutions for the 3D test problems. In the MCG fine grid updating computation for 3D problems, 2D sub-problems were solved by the alternating X-Y line Gauss-Seidel method.

For convection-diffusion equations, we present numerical results on small Reynolds number (Re). As we know, the success of Richardson extrapolation for improving the order of accuracy of numerical approximations depends on the influence of dispersion and the theoretical order of accuracy achieved before the extrapolation [9]. High Reynolds number means larger influence of dispersion and causes the failure of reaching fourth-order accuracy solutions from FOC schemes, which thereby affects the accuracy of computed solutions from Richardson extrapolation. Therefore, small Reynolds number can guarantee higher order solutions by using Richardson extrapolation. The problems with large Reynolds numbers require different solution strategies.

Problem 1.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\alpha \sin(\frac{\pi}{b} y), \qquad (x, y) \in \Omega = [0, \lambda] \times [0, b],$$

where the boundary conditions are

$$u(0, y) = u(\lambda, y) = u(x, 0) = u(x, b) = 0.$$

The parameters are chosen as

$$\alpha = \frac{F\pi}{Rb}, \lambda = 10^7 m, b = 2\pi \times 10^6 m, F = 0.3 \times 10^{-7} m^2 s^{-2}, R = 0.6 \times 10^{-3} m s^{-1}.$$

The analytical solution is

$$u = -\alpha(\frac{b}{\pi})^2 \sin(\frac{\pi y}{b})(e^{\frac{\pi x}{b}} - 1).$$

Problem 2.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -2\pi^2 \sin(\pi x) \cos(\pi y), \qquad (x, y) \in \Omega = [0, 4] \times [0, 1],$$

which has the Dirichlet boundary condition.

The analytical solution is

$$u(x, y) = \sin(\pi x) \cos(\pi y).$$

Problem 3.

$$u_{xx} + u_{yy} + u_{zz} + p(x, y, z)u_x + q(x, y, z)u_y + r(x, y, z)u_z = f(x, y, z),$$
$$(x, y, z) \in \Omega = [0, 1] \times [0, 1] \times [0, 1], \quad (5.49)$$

where the coefficients of Eq. (5.49) are set as

$$p(x, y, z) = q(x, y, z) = r(x, y, z) = Re.$$

The analytical solution is

$$u(x, y, z) = \cos(4x + 6y + 8z).$$

Problem 4.

$$u_{xx} + u_{yy} + u_{zz} + p(x,y,z)u_x + q(x,y,z)u_y + r(x,y,z)u_z = f(x,y,z),$$
$$(x,y,z) \in \Omega = [0,1] \times [0,1] \times [0,1], \quad (5.50)$$

where the coefficients of Eq. (5.50) are set as

$$\begin{cases} p(x,y,z) = Re \sin y \sin z \cos x \\ q(x,y,z) = Re \sin x \sin z \cos y \\ r(x,y,z) = Re \sin x \sin y \cos z \end{cases}.$$

The analytical solution is

$$u(x,y,z) = \cos(4x + 6y + 8z).$$

5.3.2 Accuracy and efficiency

In order to test the computed accuracy of three Richardson extrapolation-based sixth-order methods, we refined the grid from $N = 32$ to $N = 256$ for 2D Poisson equations (Problems 1 & 2) and from $N = 16$ to $N = 128$ for 3D convection-diffusion equations (Problems 3 & 4). For convenience, we use the following abbreviations: "Op-Six" reperents the Richardson extrapolation-based sixth-order method with iterative operator based interpolation; "MCG-Six" means the Richardson extrapolation-based sixth-order method with MCG updating strategy; "CR-Six" denotes the sixth-order method with completed Richardson extrapolation.

In Table 5.2, we find that all three Richardson extrapolation-based methods can achieve sixth-order in accuracy. The error comparison among these sixth-order methods shows that, in most situations (Problems 1, 3 and 4), the solutions computed by the CR-Six method are slightly more accurate than those computed by the Op-Six method, which are slightly more accurate than those from the MCG-Six method. This observation is consistent with the theoretical analysis in Section 5.2. We need to note that the qualitative relationship observed from Table 5.1 cannot be precisely applied to all of the problems. As for Problem 2, the MCG-Six method obtained

Table 5.2: Accuracy comparison among three Richardson extrapolation-based sixth-order methods

Test Problem	N	Op-Six		MCG-Six		CR-Six	
		Error	Order	Error	Order	Error	Order
Problem 1	32	2.824e-8	-	6.237e-8	-	1.861e-8	-
	64	4.421e-10	6.00	9.867e-10	5.98	3.885e-10	6.01
	128	6.891e-12	6.00	1.551e-11	5.99	4.478e-12	6.01
	256	1.014e-13	6.09	2.406e-13	6.01	6.351e-14	6.14
Problem 2	32	2.498e-6	-	2.278e-6	-	8.927e-7	-
	64	4.582e-8	5.77	3.624e-8	5.97	1.362e-8	6.03
	128	7.662e-10	5.90	5.710e-10	5.99	2.105e-10	6.02
	256	1.234e-11	5.96	8.961e-12	5.99	3.270e-12	6.01
Problem 3	16	3.547e-4	-	9.980e-4	-	1.595e-4	-
(Re=10)	32	9.234e-6	5.26	2.023e-5	5.62	2.517e-6	5.99
	64	1.764e-7	5.71	3.403e-7	5.89	3.842e-8	6.03
	128	3.062e-9	5.85	5.463e-9	5.96	9.324e-10	5.36
Problem 4	16	6.126e-5	-	2.056e-4	-	2.322e-5	-
(Re=10)	32	1.396e-6	5.46	3.694e-6	5.80	3.569e-7	6.02
	64	2.557e-8	5.77	6.035e-8	5.94	5.474e-9	6.03
	128	4.575e-10	5.80	9.816e-10	5.94	1.160e-10	5.56

more accurate solutions than the Op-Six method, although the CR-Six method still performed the best in solution accuracy among the three methods. The explanation for this lies in the uncertainty of high order partial differential terms involved in the truncation errors. It is hard to determine the magnitude and sign of these high order partial differential terms. Therefore, we cannot draw a certain qualitative relationship on accuracy among the three Richardson extrapolation-based sixth-order methods.

We also recorded the computing time for solving four test problems by three different sixth-order methods. In Table 5.3, we find that the MCG-Six method and the CR-Six method have better computational efficiency than the Op-Six method as we expected. It is because the Op-Six method involves the iterative refinement procedure which requires additional CPU time. There is no evident difference between the MCG-Six method and the CR-Six method on CPU cost in most situations (Problems 1, 2 and 3). Both are very efficient. However, for Problem 4, the CR-Six method ran much faster than the MCG-Six method. One possible reason is that the process of

Table 5.3: CPU time in seconds for three Richardson extrapolation-based sixth-order methods

N	Problem 1			Problem 2		
	Op-Six	MCG-Six	CR-Six	Op-Six	MCG-Six	CR-Six
32	0.006	0.005	0.005	0.007	0.004	0.004
64	0.025	0.020	0.021	0.031	0.021	0.022
128	0.101	0.091	0.092	0.124	0.090	0.090
256	0.433	0.395	0.394	0.491	0.403	0.393
	Problem 3 (Re=10)			Problem 4 (Re=10)		
	Op-Six	MCG-Six	CR-Six	Op-Six	MCG-Six	CR-Six
16	0.021	0.017	0.013	0.017	0.021	0.011
32	0.159	0.130	0.108	0.149	0.162	0.099
64	0.907	0.618	0.732	1.318	0.928	0.408
128	5.576	3.649	3.872	5.139	4.378	2.850

solving 2D sub-problems in the MCG-Six method took a lot of CPU time for this test problem.

5.4 Concluding Remarks

We studied three Richardson extrapolation-based sixth-order methods and analyzed the truncation errors of them respectively. All of the three methods are able to achieve the sixth-order accuracy on the fine grid. From the truncation error analysis, we summarized a general qualitative relationship on the accuracy among these methods. Four simple 2D and 3D problems are tested to compare the solution accuracy and computational efficiency among the three Richardson extrapolation-based sixth-order methods experimentally. The numerical results are basically consistent with the observation from the truncation error analysis.

From the theoretical and numerical comparison, we find that the Op-Six method can achieve relatively more accurate sixth-order solutions but ask for more computational cost. The MCG-Sixth order method computes sixth-order solutions with larger errors, but has high computational efficiency. The CR-Sixth method performs well both on accuracy and efficiency for "simple" problems with "good" conditions.

Here, the "simple" and "good" mean that the problems are not hard to solve (e.g., diffusion-dominated with small Reynolds number) and have very smooth solutions, forcing functions and coefficients in the domain.

6 Higher-Order ADI method with Completed Richardson Extrapolation for Unsteady-State Equations

6.1 Introduction

We consider the unsteady two dimensional (2D) convection-diffusion equation for a transport variable u

$$\frac{\partial u}{\partial t} - a\frac{\partial^2 u}{\partial^2 x} - b\frac{\partial^2 u}{\partial^2 y} + p\frac{\partial u}{\partial x} + q\frac{\partial u}{\partial y} = 0, \qquad (x, y, t) \in \Omega \times (0, T], \tag{6.1}$$

with initial condition

$$u(x, y, 0) = u_0(x, y), \qquad (x, y) \in \Omega,$$

and Dirichlet boundary condition

$$u(x, y, t) = g(x, y, t), \qquad (x, y, t) \in \partial\Omega \times (0, T],$$

where Ω is a rectangular domain with the boundary $\partial\Omega$, $(0, T]$ is the time interval, and g and u_0 are given functions of sufficient smoothness. In Eq. (6.1), p and q are constant, convective velocities and a and b are constant, positive diffusion coefficients in the x and y directions, respectively. In computational fluid dynamics, Eq. (6.1) is widely used to model the convection and diffusion of various physical quantities, such as mass, heat, energy, and vorticity [59].

The alternating direction implicit (ADI) methods, which aim to reduce multi-dimensional problems to a series of one dimensional (1D) problems and thus are only required to solve tridiagonal systems, are highly efficient for solving parabolic and hyperbolic initial-boundary value problems. The ADI scheme proposed by Peaceman and Rachford [55] is considered to be among the most popular methods for solving Eq. (6.1) because of its unconditional stability and high efficiency. However, the Peaceman-Rachford ADI scheme is second-order accuracy in space and may produce considerable dissipation and phase errors. To obtain more accurate solutions with

higher-order, many efforts are put to use high-order compact (HOC) schemes for spatial approximations of Eq. (6.1)[39, 54, 58, 68, 98]. Although these methods are generally able to achieve third or fourth-order accuracy in space, they have heavy computational cost because they do not apply ADI methods.

Due to the advantage of ADI methods in computational efficiency and the superiority of HOC schemes in solution accuracy, there has been growing interest in combining ADI methods with HOC schemes to develop numerical solutions for solving Eq. (6.1). Karaa and Zhang [41] proposed a high-order ADI (HOC-ADI) method for solving unsteady convection-diffusion equations, which reaches high-order accuracy and high computational efficiency simultaneously. You [83] proposed a Padé scheme-based ADI method for 2D unsteady convection-diffusion equations, which has better phase and amplitude properties. Tian and Ge [30, 74] proposed an exponential high-order compact alternating direction implicit (EHOC-ADI) method for solving 2D and 3D unsteady convection-diffusion equations, which performs better for solving convection-dominated equations with large Reynolds numbers. Later, Tian [75] derived a rational HOC scheme with ADI (RHOC-ADI) method for unsteady convection-diffusion equations and demonstrated its good performance in solution accuracy and computational efficiency. All these mentioned methods have fourth-order accuracy in space and second-order accuracy in time with high computational efficiency.

Recently, further improvements on a series of ADI methods have been achieved. One group of people use higher-order difference schemes with ADI methods to obtain sixth-order accuracy in space [45, 52]. Another group of people make efforts to improve the temporal accuracy and develop ADI methods with fourth-order accuracy in time [21, 31, 46, 76]. Among these methods, Richardson extrapolation [56] is a compelling method, which uses the computed solutions from different discretized computational domains to remove the leading truncation error terms and improve the

order of accuracy of numerical solutions. The Richardson extrapolation computation for high accuracy solutions has been applied to steady-state equations [69, 79, 81] and unsteady-state equations [9, 31, 46, 101].

In this work, we want to improve the solution accuracy in spatial and temporal domains simultaneously by using completed Richardson extrapolation and keep high computational efficiency by involving the Peaceman-Rachford ADI scheme. The completed Richardson extrapolation was proposed by Roache and Knupp [60] for the 1D Poisson equation and then extended to 1D unsteady convection-diffusion equations by Richards [57]. The main idea is to interpolate, not the higher-order solution, but rather the correction between the lower-order solution and the higher-order solution to reach the entire higher-order solution on the fine grid. We propose a higher-order ADI (ADI-CRE) method which uses the HOC-ADI method to solve Eq. (6.1) and applies completed Richardson extrapolation to improve the solution accuracy. Furthermore, we perform a stability analysis on the ADI-CRE method and discuss the impacts of Richardson extrapolation on the stability of numerical solutions. At last, numerical results are provided to show the effectiveness of the proposed method.

6.2 ADI Method with Completed Richardson Extrapolation

6.2.1 High-order ADI method

In this section, we review the high-order ADI (HOC-ADI) method proposed by Karaa and Zhang for solving 2D unsteady convection-diffusion equations [41]. In order to solve Eq. (6.1), a uniform grid is constructed in the computational domain with mesh-sizes Δx and Δy in the x and y directions, respectively. The time step size in the t direction is denoted as Δt.

For convenience, we define two finite difference operators about x

$$L_x = 1 + \frac{\Delta x^2}{12}(\delta_x^2 - \frac{p}{a}\delta_x), \qquad A_x = -(a + \frac{p^2\Delta x^2}{12a})\delta_x^2 + p\delta_x. \tag{6.2}$$

Similarly, two finite difference operators about y are defined as

$$L_y = 1 + \frac{\Delta y^2}{12}(\delta_y^2 - \frac{q}{b}\delta_y), \qquad A_y = -(b + \frac{q^2\Delta y^2}{12b})\delta_y^2 + q\delta_y. \tag{6.3}$$

By using the techniques for developing high-order compact schemes [65], Eq. (6.1) can be approximated by

$$L_x L_y \frac{\partial u}{\partial t} = -(L_y A_x + L_x A_y)u^n + O(\Delta^4), \tag{6.4}$$

where u^n is the approximate solution at time $t^n = n\Delta t$ ($n \geq 0$) and $O(\Delta^4)$ denotes the $O(\Delta x^4) + O(\Delta y^4)$ term.

Employing Crank-Nicolson time discretization, we have

$$L_x L_y \frac{u^{n+1} - u^n}{\Delta t} = -\frac{1}{2}(L_y A_x + L_x A_y)(u^{n+1} + u^n) + O(\Delta^4) + O(\Delta t^2). \tag{6.5}$$

After rearrangement and multiplying Eq. (6.5) by Δt, we have

$$(L_x L_y + \frac{\Delta t}{2}(L_y A_x + L_x A_y))u^{n+1} = (L_x L_y - \frac{\Delta t}{2}(L_y A_x + L_x A_y))u^n + O(\Delta t \Delta^4) + O(\Delta t^3). \tag{6.6}$$

By adding terms $\frac{\Delta t^2}{4} A_y A_x u^{n+1}$ and $\frac{\Delta t^2}{4} A_y A_x u^n$ to the left and right hand sides of Eq. (6.6) respectively and applying factorization, Eq. (6.6) is changed to a perturbed equation as

$$(L_x + \frac{\Delta t}{2}A_x)(L_y + \frac{\Delta t}{2}A_y)u^{n+1} = (L_x - \frac{\Delta t}{2}A_x)(L_y - \frac{\Delta t}{2}A_y)u^n, \tag{6.7}$$

where the perturbed term added to Eq. (6.6) has a truncation error of $(O(\Delta t^3 \Delta^2) + O(\Delta t^4))$. Details are referred to [41].

If $\Delta t \leq min(\Delta x, \Delta y)$, the extra term would not increase the order of truncation error of Eq. (6.6). The approximation (6.7) has second-order accuracy in time and fourth-order accuracy in space. By introducing an intermediate variable $\hat{u}$, the high-order ADI compact scheme is obtained as [41]

$$\begin{aligned} (L_x + \tfrac{\Delta t}{2}A_x)\hat{u} &= (L_x - \tfrac{\Delta t}{2}A_x)(L_y - \tfrac{\Delta t}{2}A_y)u^n, \\ (L_y + \tfrac{\Delta t}{2}A_y)u^{n+1} &= \hat{u}. \end{aligned} \tag{6.8}$$

6.2.2 Completed Richardson extrapolation in space and time

Completed Richardson extrapolation was first proposed by Roache and Knupp [60], which provides a higher-order solution on the entire fine grid by using the correction between the lower-order solution and the higher-order solution. Later, Richards [57] developed completed Richardson extrapolation for 1D time-dependent problems, which was applied to the Lax-Wendroff and Crank-Nicholson finite difference schemes. In this section, we will extend it to Karaa-Zhang's HOC-ADI scheme for solving unsteady 2D convection-diffusion equations.

For solving Eq. (6.1), a uniform fine grid with mesh-sizes Δx and Δy in the x and y directions is constructed on the rectangular spatial domain Ω. N_x and N_y denote the number of uniform intervals along the x and y directions, respectively. In the t direction, the fine time step size is Δt. The (i, j, n) point is set to correspond to the point $(i\Delta x, j\Delta y, n\Delta t)$. In order to use Richardson extrapolation, a uniform coarse grid with mesh-sizes $2\Delta x$ and $2\Delta y$ in the x and y directions on Ω is also constructed, and the coarse time step size in the t direction is $4\Delta t$. The coarse grid points then coincide in space and time with the $(2i, 2j, 4n)$ fine grid points. For the grid point (i, j, n), the exact solution is denoted by $u^*(i\Delta x, j\Delta y, n\Delta t)$. The numerical approximation on the fine grid point (i, j, n) is denoted by $uf_{i,j}^n$. The numerical approximation on the coinciding coarse grid point is denoted by $uc_{i,j}^n$, where i and j are both even and n is a multiple of 4. An example of a coarse and fine grids is given in Fig. 6.1. Black points represent the coinciding coarse grid points. The solid line grids denote the fine grid at coarse time steps, while dotted line grids denote the fine grid at fine time steps.

Consider the HOC-ADI scheme with truncation error E, which has the following

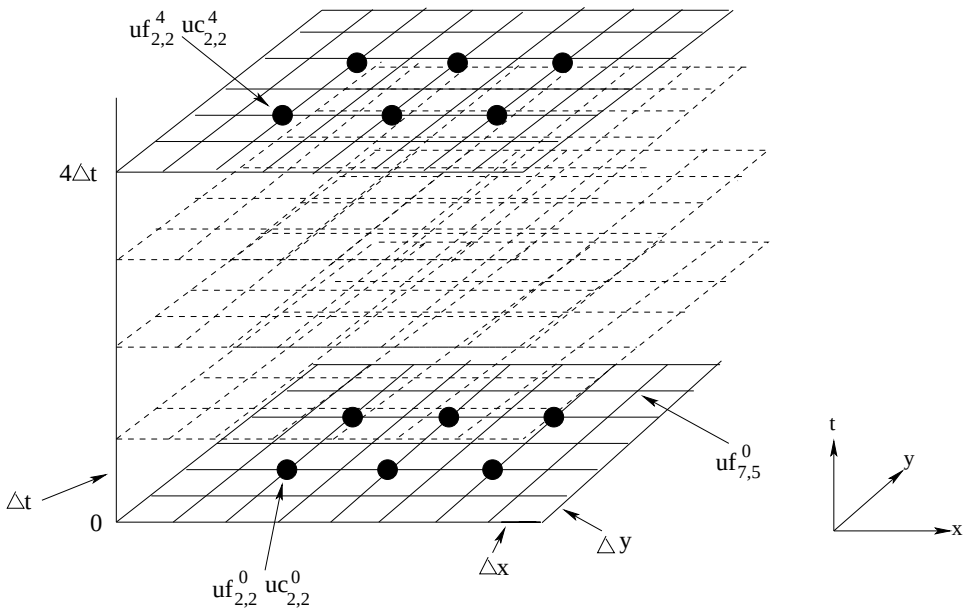


Figure 6.1: Example of a fine and coarse grid in space and time.

form as

$$E = O[(\Delta x)^4, (\Delta y)^4, (\Delta t)^2] \tag{6.9}$$
$$= (\Delta x)^4\tau_x + (\Delta y)^4\tau_y + (\Delta t)^2\tau_t + O[(\Delta x)^6, (\Delta y)^6, (\Delta t)^4].$$

The terms τ_x, τ_y and τ_t denote complex expressions involving u and its partial derivatives, which will be canceled in the Richardson extrapolation computation.

The error produced by the HOC-ADI scheme (6.8) over a time step Δt has the form of $\Delta t E$. Assume the exact solution u^* is known at the nth time step, then the error of approximate solution u at the $(i, j, n+1)$ fine grid point is

$$u^*(i\Delta x, j\Delta y, (n+1)\Delta t) - uf^{n+1}_{i,j} = \Delta t E$$
$$= \Delta t(\Delta x)^4\tau_x + \Delta t(\Delta y)^4\tau_y + (\Delta t)^3\tau_t + O[\Delta t(\Delta x)^6, \Delta t(\Delta y)^6, (\Delta t)^5]. \tag{6.10}$$

Because the error produced for a few subsequent time steps on the fine grid can be assumed to have the same magnitude as the one produced from the nth to the $(n+1)$th time step, the subsequent error is approximated to $O(\Delta t)$. And, the error contribution at each time step can be assumed to be cumulative. Therefore, the error

on the fine grid at the $(n+4)$th time step has the form as

$$\begin{aligned} & u^*(i\Delta x, j\Delta y, (n+4)\Delta t) - uf_{i,j}^{n+4} \\ & = 4\Delta t(\Delta x)^4\tau_x + 4\Delta t(\Delta y)^4\tau_y + 4(\Delta t)^3\tau_t \\ & \quad + O[(\Delta t)^2(\Delta x)^4, \Delta t(\Delta x)^6, (\Delta t)^2(\Delta y)^4, \Delta t(\Delta y)^6, (\Delta t)^5]. \end{aligned} \tag{6.11}$$

If i and j are chosen as even, the grid point at the $(n+4)$th time step can also be involved in the coarse grid approximation. The error produced after one time step on the coarse grid is

$$\begin{aligned} & u^*(i\Delta x, j\Delta y, (n+4)\Delta t) - uc_{i,j}^{n+4} \\ & = 4\Delta t(2\Delta x)^4\tau_x + 4\Delta t(2\Delta y)^4\tau_y + (4\Delta t)^3\tau_t + O[\Delta t(\Delta x)^6, \Delta t(\Delta y)^6, (\Delta t)^5]. \end{aligned} \tag{6.12}$$

For all coarse grid points $(i, j, n+4)$, we could use a linear combination of the approximations on the fine and coarse grids to remove all leading error terms and obtain a new extrapolated approximation as

$$\begin{aligned} \tilde{u}_{i,j}^{n+4} &= \frac{16uf_{i,j}^{n+4} - uc_{i,j}^{n+4}}{15} \\ &= u^*(i\Delta x, j\Delta y, (n+4)\Delta t) \\ &\quad + O[(\Delta t)^2(\Delta x)^4, \Delta t(\Delta x)^6, (\Delta t)^2(\Delta y)^4, \Delta t(\Delta y)^6, (\Delta t)^5]. \end{aligned} \tag{6.13}$$

Since the improved solution is on the coarse grid, only the coinciding fine grid points can obtain better approximations by directly interpolating the extrapolated coarse grid solution. In order to compute better approximations for the remaining fine grid points at the coarse time step (the non-black points on the solid line grids in Fig. 6.1), we use the correction between the previous solution and the extrapolated solution.

Let

$$\Gamma_{i,j}^n = 4\Delta t(\Delta x)^4\tau_x + 4\Delta t(\Delta y)^4\tau_y + 4(\Delta t)^3\tau_t. \tag{6.14}$$

The fine grid error at a coarse time step given by Eq. (6.11) can be written as

$$u^*(i\Delta x, j\Delta y, (n+4)\Delta t) - uf_{i,j}^{n+4}$$
$$= \Gamma_{i,j}^n + O[(\Delta t)^2(\Delta x)^4, \Delta t(\Delta x)^6, (\Delta t)^2(\Delta y)^4, \Delta t(\Delta y)^6, (\Delta t)^5]. \quad (6.15)$$

The coinciding coarse grid points obtain better extrapolated solutions by eliminating $\Gamma_{i,j}^n$ through Eq. (6.13). Substituting Eq. (6.13) into Eq. (6.15) gives

$$\Gamma_{i,j}^n = \tilde{u}_{i,j}^{n+4} - uf_{i,j}^{n+4} + O[(\Delta t)^2(\Delta x)^4, \Delta t(\Delta x)^6, (\Delta t)^2(\Delta y)^4, \Delta t(\Delta y)^6, (\Delta t)^5], \quad (6.16)$$

where i and j are both even, which can be viewed as the correction between the previous solution and the extrapolated solution on the coarse grid.

To compute better solutions at the fine grid points, we need to compute similar corrections $\Gamma_{i,j}^n$ for all fine grid points. The correction for $(even, even)$ fine grid points can be directly obtained from the correction of coarse grid points by using Eq. (6.16). The corrections for other fine grid points can be approximated from the correction of $(even, even)$ fine grid points. By using the idea presented in Section 4.2, the rotated grid interpolation is used to compute the correction for (odd, odd) fine grid points by

$$\Gamma_{i,j}^n = \frac{1}{4}[\Gamma_{i+1,j+1}^n + \Gamma_{i+1,j-1}^n + \Gamma_{i-1,j+1}^n + \Gamma_{i-1,j-1}^n]$$
$$+ O[\Delta t(\Delta x)^5\Delta y, \Delta t\Delta x(\Delta y)^5, (\Delta t)^4\Delta x\Delta y]. \quad (6.17)$$

Then, the standard grid interpolation is used to compute the correction for $(odd, even)$ and $(even, odd)$ fine grid points by

$$\Gamma_{i,j}^n = \frac{1}{4}[\Gamma_{i+1,j}^n + \Gamma_{i-1,j}^n + \Gamma_{i,j+1}^n + \Gamma_{i,j-1}^n]$$
$$+ O[\Delta t(\Delta x)^5\Delta y, \Delta t\Delta x(\Delta y)^5, (\Delta t)^4\Delta x\Delta y]. \quad (6.18)$$

By combining Eqs. (6.16), (6.17) and (6.18), an improved solution on the entire fine grid at coarse time step can be obtained by

$$\tilde{u}_{i,j}^{n+4} = uf_{i,j}^{n+4} + \Gamma_{i,j}^n. \quad (6.19)$$

The error in the above approximation has the form as

$$\begin{aligned} & u^*(i\Delta x, j\Delta y, (n+4)\Delta t) - \tilde{u}_{i,j}^{n+4} \\ & = O[(\Delta t)^2(\Delta x)^4, \Delta t(\Delta x)^6, (\Delta t)^2(\Delta y)^4, \Delta t(\Delta y)^6, (\Delta t)^5, \\ & \qquad \Delta t(\Delta x)^5\Delta y, \Delta t\Delta x(\Delta y)^5, (\Delta t)^4\Delta x\Delta y]. \end{aligned} \tag{6.20}$$

Thus, the improved fine grid solution, given by Eq. (6.19), at worst, has a truncation error of $O[\Delta t(\Delta x)^4, (\Delta x)^6, \Delta t(\Delta y)^4, (\Delta y)^6, (\Delta t)^4, (\Delta x)^5\Delta y, \Delta x(\Delta y)^5, (\Delta t)^3\Delta x\Delta y]$. If $\Delta t \leq min(\Delta x, \Delta y)^2$, the new approximation scheme (6.19) can achieve sixth-order in space and fourth-order in time.

6.2.3 Higher-order ADI method with completed Richardson extrapolation

In our ADI-CRE method, the HOC-ADI scheme is applied to compute both the coarse and fine grid solutions with corresponding coarse and fine time steps. At each coarse time step, the completed Richardson extrapolation is used to update the solutions on both coarse and fine grids. The updated solutions are continually used for computing on their respective grids with corresponding time steps. Therefore, the improved solution is obtained at all coarse time steps. We could also carry out the extrapolation procedure after any number of coarse time steps, even just once after the final coarse time step. However, one could expect that the extrapolation technique would be most effective at improving the accuracy of the numerical solution when it is applied after each coarse time step.

Suppose a solution is required at $t = T$ which can be calculated with N coarse time steps. Algorithm 4 describes the proposed method in which the extrapolation procedure is carried out after each coarse time step.

Algorithm 4 Higher-Order ADI Method with Completed Richardson Extrapolation for Solving 2D Unsteady Convection-Diffusion Equations

Construct and initialize the fine and coarse grids
for $n_c = 1$ to N **do**
  Compute the coarse grid solution with coarse time step using Eq. (6.8)
  for $n_f = 1$ to 4 **do**
    Compute the fine grid solution with fine time step using Eq. (6.8)
  end for
  Calculate the extrapolated solution on the coarse grid using Eq. (6.13)
  Calculate the extrapolated solution on the fine grid using Eq. (6.19) in the order:
    1. compute (even,even) fine grid nodes
    2. compute (odd,odd) fine grid nodes
    3. compute (even,odd) and (odd,even) fine grid nodes
  Both coarse and fine grids have improved solutions
end for

6.3 Stability Analysis

To study the stability of the ADI-CRE method, we use the von Neumann linear stability analysis. Assume that the numerical solution can be expressed by means of a Fourier series, whose typical term is

$$u_{ij}^n = \eta^n \exp\left[I\theta_x i\right] \exp\left[I\theta_y j\right], \tag{6.21}$$

where $I = \sqrt{-1}$, η^n is the amplitude at time step n, and $\theta_x (= k_x \Delta x)$ and $\theta_y (= k_y \Delta y)$ are phase angles with the wavenumbers k_x and k_y in the x and y directions, respectively. Therefore, for a stable method, the amplification factor $G(\theta_x, \theta_y) = \eta^{n+1}/\eta^n$ has to satisfy the stability condition $|G(\theta_x, \theta_y)| \leq 1$, for all (θ_x, θ_y) in $[-\pi, \pi]$.

We know that the improved fine grid solution is the sum of a fourth-order fine grid solution and an estimated correction. The correction is computed by a linear combination of the differences between the extrapolated higher-order coarse grid solution and the computed lower-order coarse grid solution. If completed Richardson extrapolation is only applied once after the final coarse time step, the stability depends on the method used to compute lower-order solutions on the fine and coarse grids. The involved HOC-ADI method is unconditionally stable [41], so the solution

is guaranteed to be stable. If completed Richardson extrapolation is used after each coarse time step, we only need to consider the stability of the extrapolated coarse grid solution. The process of computing improved fine grid solutions at coarse time steps does not need to be considered because linear combination procedures do not undermine the stability.

Assume

$$
\begin{aligned}
P_c &= (L_{xc} + \frac{\Delta t_c}{2} A_{xc})(L_{yc} + \frac{\Delta t_c}{2} A_{yc}), \\
Q_c &= (L_{xc} - \frac{\Delta t_c}{2} A_{xc})(L_{yc} - \frac{\Delta t_c}{2} A_{yc}), \\
P_f &= (L_{xf} + \frac{\Delta t_f}{2} A_{xf})(L_{yf} + \frac{\Delta t_f}{2} A_{yf}), \\
Q_f &= (L_{xf} - \frac{\Delta t_f}{2} A_{xf})(L_{yf} - \frac{\Delta t_f}{2} A_{yf}),
\end{aligned}
$$

where L_{xc}, A_{xc}, L_{yc}, and A_{yc} are the finite difference operators about x and y defined by Eqs. (6.2) and (6.3) for coarse grid, respectively, and similarly, L_{xf}, A_{xf}, L_{yf}, and A_{yf} are the finite difference operators with respect to x and y for fine grid, respectively.

According to the HOC-ADI scheme (6.7), we have

$$
\begin{aligned}
P_c u_c^{n+1} &= Q_c u_c^n, \\
P_f u_f^{n+1} &= Q_f u_f^n,
\end{aligned}
$$

which lead to the approximations on the coarse and fine grids, respectively, in the form as

$$
\begin{aligned}
u_c^{n+1} &= \frac{Q_c}{P_c} u_c^n, \\
u_f^{n+1} &= \frac{Q_f}{P_f} u_f^n.
\end{aligned} \tag{6.22}
$$

Algorithm 4 shows that the extrapolated computation is only conducted at the coarse time step and $\Delta t_c = 4\Delta t_f$ is established. The extrapolated solution on the

coarse grid at $n+1$ coarse time step is

$$\begin{aligned}\tilde{u}_c^{n+1} &= \frac{16u_f^{n+4}}{15} - \frac{u_c^{n+1}}{15} \\ &= \frac{16}{15}(\frac{Q_f}{P_f})^4 u_f^n - \frac{1}{15}\frac{Q_c}{P_c}u_c^n, \end{aligned} \tag{6.23}$$

where $\tilde{u}$ denotes the extrapolated solution.

Since u_f^n and u_c^n in Eq. (6.23) are both obtained from the extrapolated coarse grid solution at the nth coarse time step, we have

$$\tilde{u}_c^{n+1} = [\frac{16}{15}(\frac{Q_f}{P_f})^4 - \frac{1}{15}\frac{Q_c}{P_c}]\tilde{u}_c^n. \tag{6.24}$$

Assume $a = b = 1$ in Eq. (6.1). By substituting the discrete Fourier mode (6.21) into Eq. (6.24), the amplification factor $G(\theta_x, \theta_y)$ can be written as

$$G(\theta_x, \theta_y) = \frac{16}{15}g_{fx}^4(\frac{\theta_x}{2})g_{fy}^4(\frac{\theta_y}{2}) - \frac{1}{15}g_{cx}(\theta_x)g_{cy}(\theta_y), \tag{6.25}$$

where

$$\begin{aligned} g_{fx}(\frac{\theta_x}{2}) &= \frac{(\gamma_{1fx}(\frac{\theta_x}{2}) - \gamma_{2fx}(\frac{\theta_x}{2})) - (\gamma_{3fx}(\frac{\theta_x}{2}) + \gamma_{4fx}(\frac{\theta_x}{2}))I}{(\gamma_{1fx}(\frac{\theta_x}{2}) + \gamma_{2fx})(\frac{\theta_x}{2}) + (\gamma_{4fx}(\frac{\theta_x}{2}) - \gamma_{3fx}(\frac{\theta_x}{2}))I}, \\ g_{cx}(\theta_x) &= \frac{(\gamma_{1cx}(\theta_x) - \gamma_{2cx}(\theta_x)) - (\gamma_{3cx}(\theta_x) + \gamma_{4cx}(\theta_x))I}{(\gamma_{1cx}(\theta_x) + \gamma_{2cx}(\theta_x)) + (\gamma_{4cx}(\theta_x) - \gamma_{3cx}(\theta_x))I}, \end{aligned}$$

with

$$\begin{aligned} &\gamma_{1fx}(\frac{\theta_x}{2}) = \gamma_{1cx}(\frac{\theta_x}{2}), && \gamma_{2fx}(\frac{\theta_x}{2}) = \gamma_{2cx}(\frac{\theta_x}{2}), \\ &\gamma_{3fx}(\frac{\theta_x}{2}) = \frac{1}{2}\gamma_{3cx}(\frac{\theta_x}{2}), && \gamma_{4fx}(\frac{\theta_x}{2}) = \frac{1}{2}\gamma_{4cx}(\frac{\theta_x}{2}), \\ &\gamma_{1cx}(\theta_x) = 1 - \frac{1}{3}\sin^2\frac{\theta_x}{2}, && \gamma_{2cx}(\theta_x) = 2\Delta t(\frac{1}{\Delta x^2} + \frac{p^2}{12})\sin^2(\frac{\theta_x}{2}), \\ &\gamma_{3cx}(\theta_x) = \frac{p\Delta x}{12}\sin\theta_x, && \gamma_{4cx}(\theta_x) = \frac{p\Delta t}{2\Delta x}\sin\theta_x, \end{aligned}$$

being all non-negative. The other terms $g_{cy}(\theta_y)$ and $g_{fy}(\frac{\theta_y}{2})$ are defined similarly by replacing x by y and p by q in the above expressions. From [41], we know that $|g_{cx}(\theta_x)| \leq 1$ and $|g_{fx}(\frac{\theta_x}{2})| \leq 1$. $g_{cy}(\theta_y)$ and $g_{fy}(\frac{\theta_y}{2})$ have similar inequalities.

In order to find the condition for $|G(\theta_x, \theta_y)| \leq 1$, we need to solve the inequality

$$|\frac{16}{15} g_{fx}^4(\frac{\theta_x}{2}) g_{fy}^4(\frac{\theta_y}{2}) - \frac{1}{15} g_{cx}(\theta_x) g_{cy}(\theta_y)| \leq 1. \tag{6.26}$$

Since $\sin\theta$ has the same range as $\sin\frac{\theta}{2}$ with θ in $[-\pi, \pi]$ and $g_f(\theta)$ shares the same real parts as $g_c(\theta)$ for both x and y, the inequality (6.26) can be written as

$$|\frac{16}{15} g_{cx}^4(\theta_x) g_{cy}^4(\theta_y) - \frac{1}{15} g_{cx}(\theta_x) g_{cy}(\theta_y)| \leq 1. \tag{6.27}$$

Because $g_{cx}(\theta_x)$ has the same form as $g_{cy}(\theta_y)$ with both θ_x and θ_y in $[-\pi, \pi]$, it is reasonable to assume that $|g_{cx}(\theta_x)|$ and $|g_{cy}(\theta_y)|$ have the same upper bound M, where $|M| < 1$. The inequality (6.27) holds if

$$\frac{16}{15} M^8 + \frac{1}{15} M^2 \leq 1, \tag{6.28}$$

which is

$$M^2 \leq \sqrt[4]{\frac{7}{8}}. \tag{6.29}$$

We can choose a number which is close to but smaller than $\sqrt[4]{\frac{7}{8}}$ as the upper bound for M^2 to simplify analysis. Here, $\frac{29}{30}$ is selected and the problem becomes

$$M^2 \leq \frac{29}{30}. \tag{6.30}$$

For simplicity, we assume $\Delta = \Delta x = \Delta y$ and a general form $g(\theta)$ for $g_{cx}(\theta_x)$ and $g_{cy}(\theta_y)$ as

$$g(\theta) = \frac{(\gamma_1 - \gamma_2) - (\gamma_3 + \gamma_4) I}{(\gamma_1 + \gamma_2) + (\gamma_4 - \gamma_3) I}, \tag{6.31}$$

where

$$\gamma_1 = 1 - \frac{1}{3}\sin^2\frac{\theta}{2}, \gamma_2 = 2\Delta t(\frac{1}{\Delta^2} + \frac{c^2}{12})\sin^2(\frac{\theta}{2}), \gamma_3 = \frac{c\Delta}{12}\sin\theta, \gamma_4 = \frac{c\Delta t}{2\Delta}\sin\theta,$$

with θ in $[-\pi, \pi]$ and c representing the constant convection coefficients p and q in Eq. (6.1).

We have

$$|g(\theta)|^2 = \frac{A - 2B}{A + 2B} \leq |M|^2 \leq \frac{29}{30}, \tag{6.32}$$

where $A = \gamma_1^2 + \gamma_2^2 + \gamma_3^2 + \gamma_4^2$ and $B = \gamma_1\gamma_2 - \gamma_3\gamma_4$.

From (6.32), we get the inequality

$$A \leq 118B \tag{6.33}$$

First consider the lower bound of B. A simple calculation shows that

$$\gamma_1\gamma_2 = 2\Delta t(1 - \frac{1}{3}\sin^2\frac{\theta}{2})(\frac{1}{\Delta^2} + \frac{c^2}{12})\sin^2\frac{\theta}{2},$$
$$\gamma_3\gamma_4 = c^2\frac{\Delta t}{6}(1 - \sin^2\frac{\theta}{2})\sin^2\frac{\theta}{2} \leq c^2\frac{\Delta t}{6}(1 - \frac{1}{3}\sin^2\frac{\theta}{2})\sin^2\frac{\theta}{2}.$$

Hence, we have

$$B = \gamma_1\gamma_2 - \gamma_3\gamma_4 \geq \frac{2\Delta t}{\Delta^2}(1 - \frac{1}{3}\sin^2\frac{\theta}{2})\sin^2\frac{\theta}{2} = \frac{3\Delta t}{2\Delta^2}. \tag{6.34}$$

Substituting (6.34) into (6.33) gives

$$A \leq 177\frac{\Delta t}{\Delta^2}. \tag{6.35}$$

Then consider the upper bound of A as

$$A = \gamma_1^2 + \gamma_2^2 + \gamma_3^2 + \gamma_4^2 \leq 1 + 4(\frac{\Delta t}{\Delta^2} + \frac{\Delta t c^2}{12})^2 + (\frac{c\Delta}{12})^2 + \frac{c^2\Delta t}{4}\frac{\Delta t}{\Delta^2}. \tag{6.36}$$

If set $\lambda = \frac{\Delta t}{\Delta^2}$ and $\mu = c\Delta$, from (6.36) we have

$$\begin{aligned} A &\leq 1 + 4(\lambda + \frac{\lambda c^2\Delta^2}{12})^2 + \frac{c^2\Delta^2}{144} + \frac{\lambda^2 c^2\Delta^2}{4} \\ &= 1 + 4(\lambda^2 + \frac{\lambda^2 c^4\Delta^4}{144} + \lambda^2\frac{c^2\Delta^2}{6}) + \frac{(c\Delta)^2}{144} + \frac{\lambda^2 c^2\Delta^2}{4} \\ &= (4 + \frac{\mu^4}{144} + \frac{\mu^2}{6} + \frac{\mu^2}{4})\lambda^2 + (1 + \frac{\mu^2}{144}). \end{aligned} \tag{6.37}$$

Substituting (6.37) into (6.35) gives

$$(4 + \frac{5}{12}\mu^2 + \frac{\mu^4}{144})\lambda^2 - 177\lambda + (1 + \frac{\mu^2}{144}) \leq 0. \tag{6.38}$$

Therefore, the solution of (6.38) can guarantee the stability condition of the ADI-CRE method. If $c = 0$, the solution of (6.38) is easy to calculate and $0.00565 \leq \lambda \leq$

44.24435, which is $0.00565 \leq \frac{\Delta t}{\Delta^2} \leq 44.24435$. We notice that the stability range for pure diffusion equations with $p = q = 0$ is wide. If $c \neq 0$, there exists a solution of (6.38) when the following inequality is satisfied

$$177^2 - 4(4 + \frac{5}{12}\mu^2 + \frac{\mu^4}{144})(1 + \frac{\mu^4}{144}) \geq 0, \tag{6.39}$$

which leads to

$$\mu^2 \leq C, \tag{6.40}$$

where C is a constant.

Since $\mu^2 = \Delta^2 c^2$, in order to satisfy (6.40), a large c needs a finer mesh in spatial space. The closer the μ^2 approaches to the upper bound C, the narrower the solution range of λ becomes and a smaller time step size is required. Therefore, for convection-dominated equations with large convection coefficients p and q, it is difficult for the ADI-CRE method to obtain accurate solutions because of strict stability conditions. One possible reason is that the HOC-ADI method works not very well for convection-dominated equations [74]. If other ADI methods with better performance for convection-dominated equations, such as EHOC-ADI method, are involved in the ADI-CRE method, there might be able to have a better range of stability.

In summary, Richardson extrapolation affects the stability feature of the ADI method with which it combines. Even for the solution from an unconditionally stable ADI method, when the Richardson extrapolation procedure is applied at every coarse time step, the solution may become conditionally stable. Additionally, the range of stability is also influenced by the ADI method used for computation.

6.4 Numerical Results

In this section, we performed numerical experiments to show the accuracy and efficiency of the proposed ADI-CRE method and compared it with Karaa-Zhang's ADI scheme (HOC-ADI) [41]. Both ADI methods need to repeatedly solve a series of tridiagonal systems. The codes were written in Fortran 77 programming language and

all results were run on one login node of Lipscomb HPC Cluster at the University of Kentucky. The node has Dual Intel E5-2670 8 Core with 2.6GHz and 128GB RAM.

6.4.1 Test problem 1

The first test problem is a pure diffusion equation in the unit square domain $[0,1] \times [0,1]$, with diffusion coefficients $a = b = 1$ (and $p = q = 0$). The analytical solution to this problem is given by

$$u(x, y, t) = \exp(-2\pi^2 t) \sin(\pi x) \sin(\pi y).$$

The initial condition and Dirichlet boundary condition are directly taken from this solution.

We tested two ADI-CRE methods and compared them with the HOC-ADI method on a uniform grid with different mesh-sizes. Assume $\Delta x = \Delta y = h$. The numerical results are given in Table 6.1. The ADI-CRE(I) method uses the completed Richardson extrapolation once after the final coarse time step. The ADI-CRE(II) method uses the extrapolation technique after each coarse time step. We compared their accuracy under the L^2-norm error with respect to the analytic solution and the convergence rate. In Table 6.1, we chose $\Delta t = h^2$ and $T = 0.25$ for the verification of sixth-order accuracy in space. We find that the solutions from the ADI-CRE methods are more accurate than those from the HOC-ADI method under the same mesh-size. We also recorded the CPU time for them and noticed that the ADI-CRE methods took slightly longer time than the HOC-ADI method because of the extrapolation computation. This extra time is worth the evident improvement on accuracy. When we compared the two ADI-CRE methods, they had very close performances. Although the errors from the ADI-CRE(II) method are slightly smaller than those from the ADI-CRE(I) method, the price is to marginally increase computing time for more extrapolation procedures.

In Table 6.2, we fixed the mesh-size as $\Delta x = \Delta y = h = 1/40$ and computed

Table 6.1: L^2-norm errors, CPU time in seconds and the convergence rate in space with $h = 1/N$ and $\Delta t = h^2$ at $T = 0.25$ for Problem 1.

Strategy	N	Error	CPU time	Rate
HOC-ADI	20	8.525e-7	0.004	-
	40	5.345e-8	0.077	4.00
	80	3.341e-9	0.849	4.00
	160	2.088e-10	8.973	4.00
ADI-CRE(I)	20	1.856e-8	0.005	-
	40	2.881e-10	0.081	6.01
	80	4.493e-12	0.954	6.00
	160	6.671e-14	9.304	6.07
ADI-CRE(II)	20	1.567e-8	0.005	-
	40	2.604e-10	0.083	5.91
	80	4.133e-12	1.119	5.98
	160	6.080e-14	9.704	6.09

for $T = 0.5$ with various time step sizes. We observe that, for the errors, the ADI-CRE methods decrease faster than the HOC-ADI method with the reduction in time step size. Therefore, the completed Richardson extrapolation effectively improves the temporal accuracy. The convergence rate of ADI-CRE(I) verifies that the order of time accuracy can reach near to fourth-order. Moreover, the results of ADI-CRE(II) show that the extrapolation technique is more effective at improving the accuracy of numerical solutions when it is applied after each coarse time step.

Table 6.2: L^2-norm errors, CPU time in seconds and the convergence rate in space with $h = 1/40$ at $T = 0.5$ for Problem 1.

Strategy	Δt	Error	CPU time	Rate
HOC-ADI	1/40	1.274e-6	0.004	-
	1/80	3.224e-7	0.008	1.98
	1/160	8.082e-8	0.016	2.00
	1/320	2.019e-8	0.031	2.00
ADI-CRE(I)	1/40	3.109e-7	0.005	-
	1/80	2.167e-8	0.009	3.84
	1/160	1.665e-9	0.016	3.70
	1/320	1.846e-10	0.033	3.17
ADI-CRE(II)	1/40	8.170e-7	0.005	-
	1/80	6.625e-8	0.009	3.62
	1/160	3.101e-10	0.017	7.74
	1/320	6.012e-11	0.034	2.37

In Fig. 6.2, we plotted the L^2-norm errors at each coarse time step in each case. The figure shows the superiority of the ADI-CRE method over the HOC-ADI method. The error obtained on a 40×40 grid using the ADI-CRE method is much smaller than the one obtained using the HOC-ADI method on a 80×80 grid.

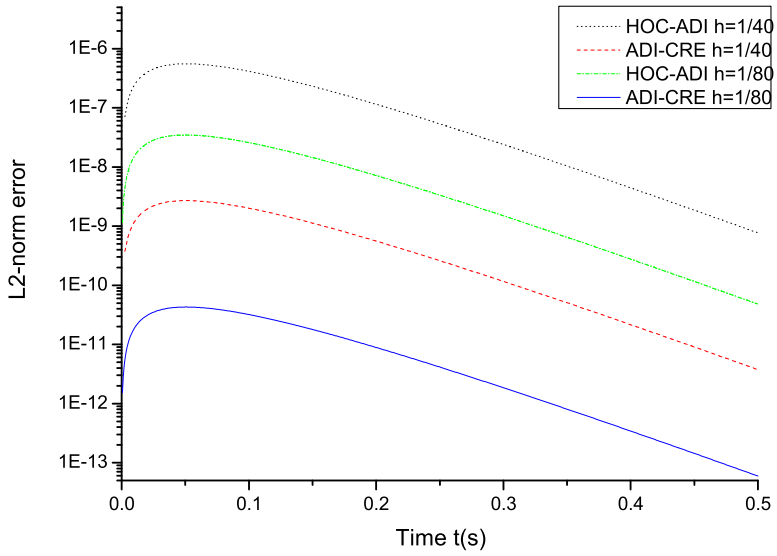


Figure 6.2: Comparison of the L^2-norm errors produced by the CRE-ADI(II) method and the HOC-ADI method at each coarse time step for Problem 1.

6.4.2 Test problem 2

Next, we consider a special problem defined in the square domain $[0, 2] \times [0, 2]$, with an analytical solution given, as in [41], by

$$u(x, y, t) = \frac{1}{4t + 1} \exp[-\frac{(x - pt - 0.5)^2}{a(4t + 1)} - \frac{(y - qt - 0.5)^2}{b(4t + 1)}].$$

The Dirichlet boundary and the initial conditions are directly taken from this solution. For the sake of comparison, we chose $a = b = 0.01$ and $p = q = 0.8$.

The comparison between the ADI-CRE method and the HOC-ADI method are presented in Tables 6.3 and 6.4. All computation were ran on a uniform grid with $\Delta x = \Delta y = h$. Analogously, we tested two ADI-CRE methods. The ADI-CRE(I)

Table 6.3: L^2-norm errors, CPU time in seconds and the convergence rate in space with $h = 2/N$ and $\Delta t = h^2$ at $T = 0.5$ for Problem 2.

Strategy	N	Error	CPU time	Rate
HOC-ADI	20	8.307e-3	0.005	-
	40	8.999e-4	0.078	3.21
	80	6.000e-5	0.868	3.91
	160	3.762e-6	8.967	4.00
ADI-CRE(I)	20	7.637e-3	0.005	-
	40	5.572e-4	0.085	3.78
	80	1.207e-5	0.916	5.53
	160	1.773e-7	9.266	6.09
ADI-CRE(II)	20	6.207e-3	0.005	-
	40	4.396e-4	0.087	3.82
	80	1.006e-5	0.980	5.45
	160	1.663e-7	9.741	5.92

applies the completed Richardson extrapolation once after the final coarse time step. The ADI-CRE(II) uses the extrapolation technique after each coarse time step. Both tables show that the solutions from the ADI-CRE methods are more accurate than those from the HOC-ADI method, yet the extrapolation procedure needs slightly more CPU time. Compared with the ADI-CRE(I) method, the ADI-CRE(II) method took more CPU time to compute more accurate solutions. In Table 6.3, we halved the mesh-size and computed numerical solutions by using different methods. We notice that, in the spatial domain, the ADI-CRE method achieves the sixth-order accuracy, while the HOC-ADI method has the fourth-order accuracy. Table 6.4 verifies that the proposed method effectively improves the accuracy in the temporal domain. The ADI-CRE method has the fourth-order accuracy in time, which is consistent with our expectation.

The L^2-norm errors at each coarse time step in each case were plotted in Fig. 6.3. This figure shows that the errors from both methods have the same behavior. The errors after applying the completed Richardson extrapolation remain smaller than the error directly from the HOC-ADI scheme.

Table 6.4: L^2-norm errors, CPU time in seconds and the convergence rate in space with $h = 1/80$ at $T = 1.0$ for Problem 2.

Strategy	Δt	Error	CPU time	Rate
HOC-ADI	1/20	3.103e-3	0.073	-
	1/40	7.653e-4	0.145	2.02
	1/80	1.900e-4	0.291	2.01
	1/160	4.699e-5	0.580	2.02
ADI-CRE(I)	1/20	3.446e-3	0.079	-
	1/40	4.000e-4	0.156	3.11
	1/80	2.593e-5	0.309	3.95
	1/160	1.616e-6	0.611	4.01
ADI-CRE(II)	1/20	1.264e-3	0.080	-
	1/40	1.057e-4	0.160	3.58
	1/80	8.962e-6	0.318	3.56
	1/160	1.096e-6	0.633	3.03

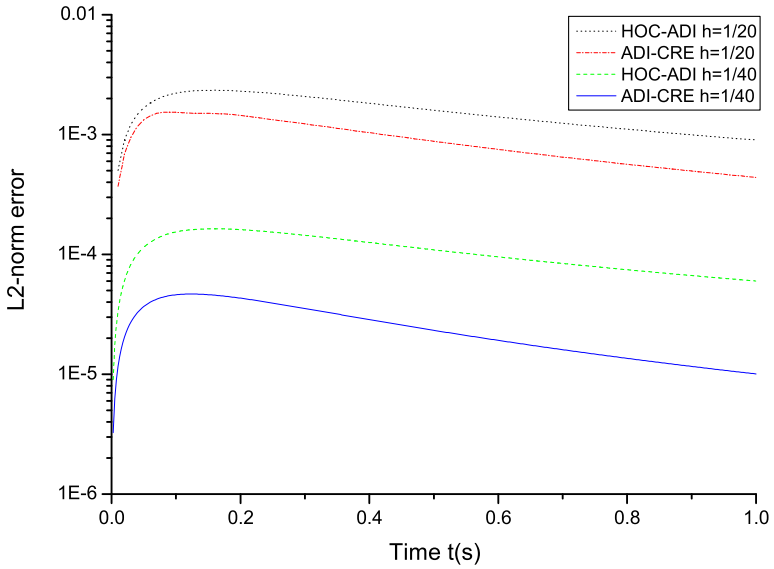


Figure 6.3: Comparison of the L^2-norm errors produced by the CRE-ADI(II) method and the HOC-ADI method at each coarse time step for Problem 2.

6.5 Concluding Remarks

We proposed a higher-order ADI method with completed Richardson extrapolation (ADI-CRE) for solving unsteady 2D convection-diffusion equations. The method has sixth-order accuracy in space and fourth-order accuracy in time. The von Neumann stability analysis is performed to show that the Richardson extrapolation computation affects the stability of the numerical solutions. The ADI-CRE method, which involves the extrapolation procedure after every coarse time step, has a wide stability range for diffusion-dominated equations, but strict stability conditions for convection-dominated equations. When only carrying out the extrapolation procedure once after the final coarse time step, the stability of ADI-CRE method depends on the high-order ADI method bound with. To demonstrate the high accuracy and efficiency of the ADI-CRE method, numerical experiments were conducted on two test problems. The computational results show that the present ADI-CRE method successfully improves the order of accuracy in spatial and temporal domains simultaneously. Finally, it is worth pointing out that the completed Richardson extrapolation can work with other ADI methods to compute high accuracy solutions for other types of time dependent equations. With different ADI methods, Richardson extrapolation computation will influence the stability in different ways. We will discuss these topics in the future.

7 Conclusion and Future Work

This dissertation presents the research work in scientific computing to develop high accuracy and high efficiency scalable numerical algorithms for solving large scale partial differential equations. This work involves Richardson extrapolation applications, high order discretization of partial differential equations, efficient solvers for discretizaed linear systems, truncation error analysis, von Neumann linear stability analysis, and numerical verifications. In this chapter, I will summarize my dissertation work and present some future research topics.

7.1 Research Accomplishments

In computational science and engineering (CSE) field, numerical solutions of partial differential equations (PDEs) play a vital role in various computer modeling and simulation applications. This dissertation proposed a series of numerical algorithms to achieve both high accuracy and high efficiency goals simultaneously. The high order accuracy is reached by using high order compact (HOC) difference discretization schemes and Richardson extrapolation. The high computational efficiency is attained by using efficient linear system solvers and multiple coarse grid (MCG) computation. The multiscale multigrid (MSMG) method is used to integrate high accuracy and high efficiency in the same framework. The HOC difference schemes mainly provide fourth-order accurate solutions on two different scale grids. Richardson extrapolation utilizes the two fourth-order solutions to obtain a sixth-order solution on the coarse grid. The linear system solvers, such as multigrid methods and alternative direction implicit (ADI) methods, are able to solve the resulting linear systems from the discretized PDEs very efficiently. The structure of multiple coarse grids enables an efficient computation for fine grid sixth-order solutions. The MSMG method combines the Richardson extrapolation-based high accuracy computation and the multigrid

computation in the same framework by using the multiscale strategy.

Multiple Coarse Grid Updating Strategy

A new fine grid updating strategy based on multiple coarse grids is developed to accelerate the Richardson extrapolation-based MSMG computation. The sixth-order solution from Richardson extrapolation is on the coarse grid. An existing strategy to obtain sixth-order solutions for fine grid points is to iteratively perform an operator based interpolation on the fine grid in a specific sequence. However, this procedure equals to an iterative refinement procedure, which has a slow convergence rate and requires a lot of CPU time. In order to curtail the CPU cost, a direct calculation method is proposed to replace the existing iterative procedure for computing fine grid sixth-order solutions. Through combining different fine grid points to virtually generate various non-uniform coarse grid views, sixth-order solutions of fine grid points can be directly solved group by group. Based on this idea, the MCG updating strategy is developed for 2D and 3D problems, respectively. The MSMG method with Richardson extrapolation and MCG updating strategy was tested to solve 2D and 3D steady-state PDEs, which are presented in Chapters 2 and 3.

Sixth-order Solution with Completed Richardson Extrapolation

The completed Richardson extrapolation was applied to compute sixth-order solutions on the entire fine grid. Although people usually use the extrapolated sixth-order coarse grid solution to seek appropriate interpolation process for computing sixth-order fine grid solution, the proposed method uses the correction between the fourth order solution and the extrapolated sixth-order solution on the coarse grid to estimate the fine grid fourth-order error, which can be added back to the fourth-order solution and thus obtain the sixth-order solution oh the fine grid. Since the completed

Richardson extrapolation involves simple calculations, the proposed method is able to achieve high accuracy with low CPU costs.

Truncation Error Comparison among Three Richardson extrapolation-based Sixth-Order Methods

We discussed three Richardson-extrapolated sixth-order methods for solving PDEs. Although all of these methods can reach sixth-order solutions on the entire fine grid, they generate different errors due to different strategies for computing improved fine grid solutions. These strategies are iterative operator based interpolation, multiple coarse grid updating strategy, and completed Richardson extrapolation. The truncation error analysis was conducted on these methods respectively for the purpose of accuracy comparison.

Higher-order ADI Method with Completed Richardson Extrapolation

The Richardson extrapolation technique was extended to high accuracy and high efficiency computation for unsteady 2D convection-diffusion equations. The proposed ADI-CRE method incorporates Karra-Zhang's high-order ADI (HOC-ADI) method and completed Richardson extrapolation. On one hand, by constructing special coarse grids, the completed Richardson extrapolation method can effectively improve the order of computed solution from the HOC-ADI method in spatial and temporal domains simultaneously. On the other hand, involving ADI scheme guarantees high computational efficiency. The ADI-CRE method has sixth-order accuracy in space and fourth-order accuracy in time.

Stability Analysis on Numerical Methods with Richardson Extrapolation

For numerical solutions of unsteady-state equations, stability is a key issue to be con-

sidered. In order to examine the influence of Richardson extrapolation upon other numerical methods on stability, the von Neumann linear stability analysis was conducted on the ADI-CRE method. We find that the Richardson extrapolation procedure undermines the stability of the underlying method. When Richardson extrapolation is applied to the solution computed from the HOC-ADI method, which is unconditionally stable, at every coarse time step, the solution becomes conditionally stable. The range of stability is affected by convection coefficients and the underlying ADI method.

7.2 Future Work

This dissertation has explored some fundamental work related to high accuracy high efficiency scalable numerical solutions for PDEs. However, it is the first step in developing useful computational frameworks for solving large scale PDEs in CSE applications. In the near future, I will continue my research on improving MSMG computation with multiple coarse grids, developing high accuracy and high efficiency numerical solutions for unsteady-state PDEs, and using numerical methods to solve application problems.

Multiple Coarse Grid MSMG Computational Framework

In Chapters 2 and 3, I used multiple coarse grids to eliminate the fine grid iterative refinement process for high-order solution computation. Besides this benefit, the multiple coarse grids could be used to build a scalable, reliable, and concurrent MSMG computational framework. Since the MSMG method involves multigrid computation, it leads to different problem sizes (number of unknowns) on different grid levels. The key issue for efficiently using multiple processor architectures in the current generation supercomputers is to maintain the amount of the computation (the problem size) at all levels of the multigrid computation. This entails at least two issues to

work: (1) Use multiple coarse grids to accelerate the convergence rate of multigrid computation like in the superconvergent multigrid method. (2) Use multiple coarse grids to independently compute sixth-order solutions for different groups of fine grid points.

High Accuracy High Efficiency Computation for Unsteady-State PDEs

In Chapter 6, completed Richardson extrapolation was combined with a special ADI method to improve the order of solution accuracy in time and space effectively, however, there are still many issues worthy of further study. For example: (1) How does the selection of underlying ADI methods affect the solution accuracy and stability? (2)Is it possible to apply the idea of multiple coarse grid updating strategy with Richardson extrapolation for high accuracy and high efficiency computation for unsteady-state PDEs? (3) What computational strategies are appropriate for high accuracy and high efficiency computation for hard problems, such as convection-dominated equations with large Reynolds number, Neumann boundary conditions, discontinuous coefficients, non-rectangular domains and etc.?

Numerical Methods Application in Financial Engineering

Numerical computational methods have been widely used in the attractive field of option pricing, which is a core task of financial engineering and risk analysis. The famous Balck-Scholes equation, which represents the most prominent financial market model, is essentially an unsteady-state convection-diffusion equation. Therefore, numerical methods for PDEs (such as finite difference methods and finite element methods) are very important methods for financial options [63]. The Richardson extrapolation techniques are also used to enhance the computational efficiency and/or accuracy of option pricing in the literature [3, 8, 10]. At present, Crank-Nicolson method has been

successfully used to solve Black-Scholes equation and reached second-order accuracy. The techniques discussed in this dissertation for computing high-order accuracy solutions for PDEs with high computational efficiency are possible to apply for solving PDEs like Black-Scholes equation in financial engineering. I would like to collaborate with experts in finance to study computational techniques for financial models.

Appendix A

The coefficients of the 9-point FOC scheme for the 2D convection-diffusion equation with unequal mesh-size discretization

For the 9-point FOC scheme Eq. (2.21), the coefficients and right-hand side are given by

$$
\begin{aligned}
\alpha_0 &= -\Delta x^2 \frac{p_0^2\lambda^2 + q_0^2\lambda^2}{\lambda^2} - \Delta x \frac{p_1\lambda^2 - p_3\lambda^2 + q_2\lambda - q_4\lambda}{\lambda^2} - \frac{10\lambda^2 + 10}{\lambda^2}, \\
\alpha_1 &= \frac{\Delta x^2}{8} \frac{(p_2q_0\lambda^3 - p_4q_0\lambda^3 + 4p_0^2\lambda^2 + p_0p_1\lambda^2 - p_0p_3\lambda^2)}{\lambda^2} \\
&\quad + \frac{\Delta x}{8} \frac{(12p_0\lambda^2 + 6p_1\lambda^2 + 2p_2\lambda^2 - 2p_3\lambda^2 + 2p_4\lambda^2 - 4p_0)}{\lambda^2} + \frac{1}{8}\frac{(40\lambda^2 - 8)}{\lambda^2}, \\
\alpha_2 &= \frac{\Delta x^2}{8} \frac{(4q_0^2\lambda^2 + q_0q_2\lambda^2 - q_0q_4\lambda^2 + p_0q_1\lambda - p_0q_3\lambda)}{\lambda^2} \\
&\quad + \frac{\Delta x}{8} \frac{(-4q_0\lambda^3 + 12q_0\lambda + 2q_1\lambda + 6q_2\lambda + 2q_3\lambda - 2q_4\lambda)}{\lambda^2} + \frac{1}{8}\frac{(-8\lambda^2 + 40)}{\lambda^2}, \\
\alpha_3 &= \frac{\Delta x^2}{8} \frac{(-p_2q_0\lambda^3 + p_4q_0\lambda^3 + 4p_0^2\lambda^2 - p_0p_1\lambda^2 + p_0p_3\lambda^2)}{\lambda^2} \\
&\quad + \frac{\Delta x}{8} \frac{(-12p_0\lambda^2 + 2p_1\lambda^2 - 2p_2\lambda^2 - 6p_3\lambda^2 - 2p_4\lambda^2 + 4p_0)}{\lambda^2} + \frac{1}{8}\frac{(40\lambda^2 - 8)}{\lambda^2}, \\
\alpha_4 &= -\frac{\Delta x^2}{8} \frac{(-4q_0^2\lambda^2 + q_0q_2\lambda^2 - q_0q_4\lambda^2 + p_0q_1\lambda - p_0q_3\lambda)}{\lambda^2} \\
&\quad - \frac{\Delta x}{8} \frac{(-4q_0\lambda^3 + 12q_0\lambda + 2q_1\lambda - 2q_2\lambda + 2q_3\lambda + 6q_4\lambda)}{\lambda^2} - \frac{1}{8}\frac{(8\lambda^2 - 40)}{\lambda^2}, \\
\alpha_5 &= \frac{\Delta x^2}{8} \frac{(p_0q_0\lambda^3 + p_0q_0\lambda)}{\lambda^2} \\
&\quad + \frac{\Delta x}{8} \frac{(2q_0\lambda^3 + 2p_0\lambda^2 + p_2\lambda^2 - p_4\lambda^2 + 2q_0\lambda + q_1\lambda - q_3\lambda + 2p_0)}{\lambda^2} + \frac{1}{8}\frac{(4\lambda^2 + 4)}{\lambda^2}, \\
\alpha_6 &= -\frac{\Delta x^2}{8} \frac{(p_0q_0\lambda^3 + p_0q_0\lambda)}{\lambda^2} \\
&\quad - \frac{\Delta x}{8} \frac{(-2q_0\lambda^3 + 2p_0\lambda^2 + p_2\lambda^2 - p_4\lambda^2 - 2q_0\lambda + q_1\lambda - q_3\lambda + 2p_0)}{\lambda^2} - \frac{1}{8}\frac{(-4\lambda^2 - 4)}{\lambda^2}, \\
\alpha_7 &= \frac{\Delta x^2}{8} \frac{(p_0q_0\lambda^3 + p_0q_0\lambda)}{\lambda^2} \\
&\quad + \frac{\Delta x}{8} \frac{(-2q_0\lambda^3 - 2p_0\lambda^2 + p_2\lambda^2 - p_4\lambda^2 - 2q_0\lambda + q_1\lambda - q_3\lambda - 2p_0)}{\lambda^2} + \frac{1}{8}\frac{(4\lambda^2 + 4)}{\lambda^2},
\end{aligned}
$$

$$\alpha_8 = -\frac{\Delta x^2}{8}\frac{(p_0 q_0 \lambda^3 + p_0 q_0 \lambda)}{\lambda^2}$$
$$- \frac{\Delta x}{8}\frac{(2q_0\lambda^3 - 2p_0\lambda^2 + p_2\lambda^2 - p_4\lambda^2 + 2q_0\lambda + q_1\lambda - q3\lambda - 2p_0)}{\lambda^2} - \frac{1}{8}\frac{(-4\lambda^2 - 4)}{\lambda^2},$$
$$F = \frac{\Delta x^2}{2}[8f_0 + f_1 + f_2 + f_3 + f_4] + \frac{\Delta x^3}{4}[p_0(f_1 - f_3) + q_0\lambda(f_2 - f_4)],$$

where $\lambda = \frac{\Delta y}{\Delta x}$.

Appendix B

The coefficients of the 19-point FOC scheme for the 3D convection-diffusion equation with unequal mesh-size discretization

For the 19-point FOC scheme Eq. (3.7), the coefficients and right-hand side are given by

$$\begin{aligned}
\alpha_0 &= -\frac{1}{6}\frac{(\lambda_1^2\lambda_2^2p_0^2+\lambda_1^2\lambda_2^2q_0^2+\lambda_1^2\lambda_2^2r_0^2)\Delta x^2}{\lambda_1^2\lambda_2^2}-\frac{1}{6}\frac{8\lambda_1^2\lambda_2^2+8\lambda_1^2+8\lambda_2^2}{\lambda_1^2\lambda_2^2}\\
&\quad -\frac{1}{6}\frac{(\lambda_1^2\lambda_2^2p_1-\lambda_1^2\lambda_2^2p_3+\lambda_1^2\lambda_2r_5-\lambda_1^2\lambda_2r_6+\lambda_1\lambda_2^2q_2-\lambda_1\lambda_2^2q_4)}{\lambda_1^2\lambda_2^2},\\
\alpha_1 &= \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((\lambda_1^3\lambda_2^2p_2q_0-\lambda_1^3\lambda_2^2p_4q_0+\lambda_1^2\lambda_2^3p_5r_0-\lambda_1^2\lambda_2^3p_6r_0+4\lambda_1^2\lambda_2^2p_0^2+\lambda_1^2\lambda_2^2p_0p_1\\
&\quad -\lambda_1^2\lambda_2^2p_0p_3)\Delta x^2)+\frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((4\lambda_1^2\lambda_2^2p_0+6\lambda_1^2\lambda_2^2p_1+2\lambda_1^2\lambda_2^2p_2-2\lambda_1^2\lambda_2^2p_3+2\lambda_1^2\lambda_2^2p_4\\
&\quad +2\lambda_1^2\lambda_2^2p_5+2\lambda_1^2\lambda_2^2p_6-4\lambda_1^2p_0-4\lambda_2^2p_0)\Delta x)+\frac{1}{48}\frac{32\lambda_1^2\lambda_2^2-8\lambda_1^2-8\lambda_2^2}{\lambda_1^2\lambda_2^2},\\
\alpha_2 &= \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((4\lambda_1^2\lambda_2^2q_0^2+\lambda_1^2\lambda_2^2q_0q_2-\lambda_1^2\lambda_2^2q_0q_4+\lambda_1\lambda_2^3q_5r_0-\lambda_1\lambda_2^3q_6r_0+\lambda_1\lambda_2^2p_0q_1\\
&\quad -\lambda_1\lambda_2^2p_0q_3)\Delta x^2)+\frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((-4\lambda_1^3\lambda_2^2q_0-4\lambda_1^3q_0+4\lambda_1\lambda_2^2q_0+2\lambda_1\lambda_2^2q_1+6\lambda_1\lambda_2^2q_2\\
&\quad +2\lambda_1\lambda_2^2q_3-2\lambda_1\lambda_2^2q_4+2\lambda_1\lambda_2^2q_5+2\lambda_1\lambda_2^2q_6)\Delta x)+\frac{1}{48}\frac{-8\lambda_1^2\lambda_2^2-8\lambda_1^2+32\lambda_2^2}{\lambda_1^2\lambda_2^2},\\
\alpha_3 &= -\frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((\lambda_1^3\lambda_2^2p_2q_0-\lambda_1^3\lambda_2^2p_4q_0+\lambda_1^2\lambda_2^3p_5r_0-\lambda_1^2\lambda_2^3p_6r_0-4\lambda_1^2\lambda_2^2p_0^2+\lambda_1^2\lambda_2^2p_0p_1\\
&\quad -\lambda_1^2\lambda_2^2p_0p_3)\Delta x^2)-\frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((4\lambda_1^2\lambda_2^2p_0-2\lambda_1^2\lambda_2^2p_1+2\lambda_1^2\lambda_2^2p_2+6\lambda_1^2\lambda_2^2p_3+2\lambda_1^2\lambda_2^2p_4\\
&\quad +2\lambda_1^2\lambda_2^2p_5+2\lambda_1^2\lambda_2^2p_6-4\lambda_1^2p_0-4\lambda_2^2p_0)\Delta x)-\frac{1}{48}\frac{-32\lambda_1^2\lambda_2^2+8\lambda_1^2+8\lambda_2^2}{\lambda_1^2\lambda_2^2},\\
\alpha_4 &= \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((4\lambda_1^2\lambda_2^2q_0^2-\lambda_1^2\lambda_2^2q_0q_2+\lambda_1^2\lambda_2^2q_0q_4-\lambda_1\lambda_2^3q_5r_0+\lambda_1\lambda_2^3q_6r_0-\lambda_1\lambda_2^2p_0q_1\\
&\quad +\lambda_1\lambda_2^2p_0q_3)\Delta x^2)+\frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((4\lambda_1^3\lambda_2^2q_0+4\lambda_1^3q_0-4\lambda_1\lambda_2^2q_0-2\lambda_1\lambda_2^2q_1+2\lambda_1\lambda_2^2q_2\\
&\quad -2\lambda_1\lambda_2^2q_3-6\lambda_1\lambda_2^2q_4-2\lambda_1\lambda_2^2q_5-2\lambda_1\lambda_2^2q_6)\Delta x)+\frac{1}{48}\frac{-8\lambda_1^2\lambda_2^2-8\lambda_1^2+32\lambda_2^2}{\lambda_1^2\lambda_2^2},
\end{aligned}$$

$$
\begin{aligned}
\alpha_5 &= \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((\lambda_1^3\lambda_2 q_0 r_2 - \lambda_1^3\lambda_2 q_0 r_4 + 4\lambda_1^2\lambda_2^2 r_0^2 + \lambda_1^2\lambda_2^2 r_0 r_5 - \lambda_1^2\lambda_2^2 r_0 r_6 + \lambda_1^2\lambda_2 p_0 r_1 \\
&\quad - \lambda_1^2\lambda_2 p_0 r_3)\Delta x^2) + \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((-4\lambda_1^2\lambda_2^3 r_0 + 4\lambda_1^2\lambda_2 r_0 + 2\lambda_1^2\lambda_2 r_1 + 2\lambda_1^2\lambda_2 r_2 + 2\lambda_1^2\lambda_2 r_3 \\
&\quad + 2\lambda_1^2\lambda_2 r_4 + 6\lambda_1^2\lambda_2 r_5 - 2\lambda_1^2\lambda_2 r_6 - 4\lambda_2^3 r_0)\Delta x) + \frac{1}{48}\frac{-8\lambda_1^2\lambda_2^2 + 32\lambda_1^2 - 8\lambda_2^2}{\lambda_1^2\lambda_2^2}, \\
\alpha_6 &= -\frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((\lambda_1^3\lambda_2 q_0 r_2 - \lambda_1^3\lambda_2 q_0 r_4 - 4\lambda_1^2\lambda_2^2 r_0^2 + \lambda_1^2\lambda_2^2 r_0 r_5 - \lambda_1^2\lambda_2^2 r_0 r_6 + \lambda_1^2\lambda_2 p_0 r_1 \\
&\quad - \lambda_1^2\lambda_2 p_0 r_3)\Delta x^2) - \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((-4\lambda_1^2\lambda_2^3 r_0 + 4\lambda_1^2\lambda_2 r_0 + 2\lambda_1^2\lambda_2 r_1 + 2\lambda_1^2\lambda_2 r_2 + 2\lambda_1^2\lambda_2 r_3 \\
&\quad + 2\lambda_1^2\lambda_2 r_4 - 2\lambda_1^2\lambda_2 r_5 + 6\lambda_1^2\lambda_2 r_6 - 4\lambda_2^3 r_0)\Delta x) - \frac{1}{48}\frac{8\lambda_1^2\lambda_2^2 - 32\lambda_1^2 + 8\lambda_2^2}{\lambda_1^2\lambda_2^2}, \\
\alpha_7 &= \frac{1}{48}\frac{(\lambda_1^3 p_0 q_0 + \lambda_1 p_0 q_0)\Delta x^2}{\lambda_1^2} + \frac{1}{48}\frac{4\lambda_1^2 + 4}{\lambda_1^2} \\
&\quad + \frac{1}{48}\frac{(2\lambda_1^3 q_0 + 2\lambda_1^2 p_0 + \lambda_1^2 p_2 - \lambda_1^2 p_4 + 2\lambda_1 q_0 + \lambda_1 q_1 - \lambda_1 q_3 + 2p_0)\Delta x}{\lambda_1^2}, \\
\alpha_8 &= -\frac{1}{48}\frac{(\lambda_1^3 p_0 q_0 + \lambda_1 p_0 q_0)\Delta x^2}{\lambda_1^2} - \frac{1}{48}\frac{-4\lambda_1^2 - 4}{\lambda_1^2} \\
&\quad - \frac{1}{48}\frac{1}{\lambda_1^2}((-2\lambda_1^3 q_0 + 2\lambda_1^2 p_0 + \lambda_1^2 p_2 - \lambda_1^2 p_4 - 2\lambda_1 q_0 + \lambda_1 q_1 - \lambda_1 q_3 + 2p_0)\Delta x), \\
\alpha_9 &= \frac{1}{48}\frac{(\lambda_1^3 p_0 q_0 + \lambda_1 p_0 q_0)\Delta x^2}{\lambda_1^2} + \frac{1}{48}\frac{4\lambda_1^2 + 4}{\lambda_1^2} \\
&\quad + \frac{1}{48}\frac{1}{\lambda_1^2}((-2\lambda_1^3 q_0 - 2\lambda_1^2 p_0 + \lambda_1^2 p_2 - \lambda_1^2 p_4 - 2\lambda_1 q_0 + \lambda_1 q_1 - \lambda_1 q_3 - 2p_0)\Delta x), \\
\alpha_{10} &= -\frac{1}{48}\frac{(\lambda_1^3 p_0 q_0 + \lambda_1 p_0 q_0)\Delta x^2}{\lambda_1^2} - \frac{1}{48}\frac{-4\lambda_1^2 - 4}{\lambda_1^2} \\
&\quad - \frac{1}{48}\frac{(2\lambda_1^3 q_0 - 2\lambda_1^2 p_0 + \lambda_1^2 p_2 - \lambda_1^2 p_4 + 2\lambda_1 q_0 + \lambda_1 q_1 - \lambda_1 q_3 - 2p_0)\Delta x}{\lambda_1^2}, \\
\alpha_{11} &= \frac{1}{48}\frac{(\lambda_2^3 p_0 r_0 + \lambda_2 p_0 r_0)\Delta x^2}{\lambda_2^2} + \frac{1}{48}\frac{4\lambda_2^2 + 4}{\lambda_2^2} \\
&\quad + \frac{1}{48}\frac{(2\lambda_2^3 r_0 + 2\lambda_2^2 p_0 + \lambda_2^2 p_5 - \lambda_2^2 p_6 + 2\lambda_2 r_0 + \lambda_2 r_1 - \lambda_2 r_3 + 2p_0)\Delta x}{\lambda_2^2}, \\
\alpha_{12} &= \frac{1}{48}\frac{(\lambda_1^3\lambda_2 q_0 r_0 + \lambda_1\lambda_2^3 q_0 r_0)\Delta x^2}{\lambda_1^2\lambda_2^2} \\
&\quad + \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((2\lambda_1^3 q_0 + 2\lambda_1^2\lambda_2 r_0 + \lambda_1^2\lambda_2 r_2 - \lambda_1^2\lambda_2 r_4 + 2\lambda_1\lambda_2^2 q_0 + \lambda_1\lambda_2^2 q_5 \\
&\quad - \lambda_1\lambda_2^2 q_6 + 2\lambda_2^3 r_0)\Delta x) + \frac{1}{48}\frac{4\lambda_1^2 + 4\lambda_2^2}{\lambda_1^2\lambda_2^2},
\end{aligned}
$$

$$\alpha_{13} = -\frac{1}{48}\frac{(\lambda_2^3 p_0 r_0 + \lambda_2 p_0 r_0)\Delta x^2}{\lambda_2^2} - \frac{1}{48}\frac{-4\lambda_2^2 - 4}{\lambda_2^2}$$
$$- \frac{1}{48}\frac{(-2\lambda_2^3 r_0 + 2\lambda_2^2 p_0 + \lambda_2^2 p_5 - \lambda_2^2 p_6 - 2\lambda_2 r_0 + \lambda_2 r_1 - \lambda_2 r_3 + 2p_0)\Delta x}{\lambda_2^2},$$
$$\alpha_{14} = -\frac{1}{48}\frac{(\lambda_1^3 \lambda_2 q_0 r_0 + \lambda_1 \lambda_2^3 q_0 r_0)\Delta x^2}{\lambda_1^2 \lambda_2^2}$$
$$- \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((2\lambda_1^3 q_0 - 2\lambda_1^2\lambda_2 r_0 + \lambda_1^2\lambda_2 r_2 - \lambda_1^2\lambda_2 r_4 + 2\lambda_1\lambda_2^2 q_0 + \lambda_1\lambda_2^2 q_5$$
$$- \lambda_1\lambda_2^2 q_6 - 2\lambda_2^3 r_0)\Delta x) - \frac{1}{48}\frac{-4\lambda_1^2 - 4\lambda_2^2}{\lambda_1^2\lambda_2^2},$$
$$\alpha_{15} = -\frac{1}{48}\frac{(\lambda_2^3 p_0 r_0 + \lambda_2 p_0 r_0)\Delta x^2}{\lambda_2^2} - \frac{1}{48}\frac{-4\lambda_2^2 - 4}{\lambda_2^2}$$
$$- \frac{1}{48}\frac{(2\lambda_2^3 r_0 - 2\lambda_2^2 p_0 + \lambda_2^2 p_5 - \lambda_2^2 p_6 + 2\lambda_2 r_0 + \lambda_2 r_1 - \lambda_2 r_3 - 2p_0)\Delta x}{\lambda_2^2},$$
$$\alpha_{16} = -\frac{1}{48}\frac{(\lambda_1^3 \lambda_2 q_0 r_0 + \lambda_1 \lambda_2^3 q_0 r_0)\Delta x^2}{\lambda_1^2 \lambda_2^2}$$
$$- \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((-2\lambda_1^3 q_0 + 2\lambda_1^2\lambda_2 r_0 + \lambda_1^2\lambda_2 r_2 - \lambda_1^2\lambda_2 r_4 - 2\lambda_1\lambda_2^2 q_0 + \lambda_1\lambda_2^2 q_5$$
$$- \lambda_1\lambda_2^2 q_6 + 2\lambda_2^3 r_0)\Delta x) - \frac{1}{48}\frac{-4\lambda_1^2 - 4\lambda_2^2}{\lambda_1^2\lambda_2^2},$$
$$\alpha_{17} = \frac{1}{48}\frac{(\lambda_2^3 p_0 r_0 + \lambda_2 p_0 r_0)\Delta x^2}{\lambda_2^2} + \frac{1}{48}\frac{4\lambda_2^2 + 4}{\lambda_2^2}$$
$$+ \frac{1}{48}\frac{(-2\lambda_2^3 r_0 - 2\lambda_2^2 p_0 + \lambda_2^2 p_5 - \lambda_2^2 p_6 - 2\lambda_2 r_0 + \lambda_2 r_1 - \lambda_2 r_3 - 2p_0)\Delta x}{\lambda_2^2},$$
$$\alpha_{18} = \frac{1}{48}\frac{(\lambda_1^3 \lambda_2 q_0 r_0 + \lambda_1 \lambda_2^3 q_0 r_0)\Delta x^2}{\lambda_1^2 \lambda_2^2}$$
$$+ \frac{1}{48}\frac{1}{\lambda_1^2\lambda_2^2}((-2\lambda_1^3 q_0 - 2\lambda_1^2\lambda_2 r_0 + \lambda_1^2\lambda_2 r_2 - \lambda_1^2\lambda_2 r_4 - 2\lambda_1\lambda_2^2 q_0 + \lambda_1\lambda_2^2 q_5$$
$$- \lambda_1\lambda_2^2 q_6 - 2\lambda_2^3 r_0)\Delta x) + \frac{1}{48}\frac{4\lambda_1^2 + 4\lambda_2^2}{\lambda_1^2\lambda_2^2},$$
$$F = \frac{1}{48}(2f_2\Delta x^3\lambda_1 q_0 - 2f_4\Delta x^3\lambda_1 q_0 + 2f_5\Delta x^3\lambda_2 r_0 - 2f_6\Delta x^3\lambda_2 r_0 + 2f_1\Delta x^3 p_0 - 2f_3\Delta x^3 p_0$$
$$+ 24f_0\Delta x^2 + 4f_1\Delta x^2 + 4f_2\Delta x^2 + 4f_3\Delta x^2 + 4f_4\Delta x^2 + 4f_5\Delta x^2 + 4f_6\Delta x^2),$$

where $\lambda_1 = \frac{\Delta y}{\Delta x}$ and $\lambda_2 = \frac{\Delta z}{\Delta x}$.

Bibliography

[1] Y. Adam, Highly accurate compact implicit methods and boundary conditions. *J. Comput. Phys.*, **24**(1), 10-22, 1977.

[2] U. Ananthakrishnaiah, R. Manohar, and J. W. Stephenson, Fourth-order difference methods for three-dimensional general linear elliptic problems with variable coefficients. *Numer. Methods Partial Differential Eq.*, **3**, 229-240, 1987.

[3] A. Arciniega and E. Allen, Extrapolation of difference methods in option valuation. *Appl. Math. Comput.*, **153**, 165-186, 2004.

[4] A. Brandt, Multi-level adaptive solutions to boundary-value problems. *Math. Comput.*, **31**(138), 333-390, 1977.

[5] A. Brandt, Multigrid solvers on parallel computers. In M. H. Schultz, editor, *Elliptic Problem Solvers*, 39-83, Academic Press, New York, 1981.

[6] C. Brezinski and M. R. Zaglia, *Extrapolation Method: Theory and Practice*. North Holland, Berlin, 1991.

[7] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*. SIAM, Philadelphia, PA, 2nd edition, 2000.

[8] M. Broadie and J. Detemple, American option valuation: New bounds, approximations, and a comparison of existing methods. *The Review of Financial Studies*, **9**(4), 1211-1250, 1996.

[9] C. Burg and T. Erwin, Application of Richardson extrapolation to the numerical solution of partial differential equations. *Numer. Methods Partial Differential Equations*, **25**, 810-832, 2009.

[10] C. C. Chang, S. L. Chung, and R. C. Stapleton, Richardson extrapolation technique for pricing American-style options. *J. Futures Markets*, **27**(8), 791-817, 2007.

[11] W. Cheney and D. Kincard, *Numerical Mathematics and Computing*. Brooks/Cole Publishing, Pacific Grove, CA, 4th edition, 1999.

[12] P. C. Chu and C. Fan, A three-point combined compact difference scheme. *J. Comput. Phys.*, **140**, 370-399, 1998.

[13] P. C. Chu and C. Fan, A three-point six-order nonuniform combined compact difference scheme. *J. Comput. Phys.*, **148**, 663-674, 1999.

[14] R. Dai, Y. Wang and J. Zhang, Fast and high accuracy multiscale multigrid method with multiple coarse grid updating strategy for 3D convection diffusion equation. *Comput. Math. Appl.*, **66**, 542-559, 2013.

[15] R. Dai, J. Zhang and Y. Wang, Sixth Order Compact Approximation with Completed Richardson Extrapolation. *2013 ICCIS*, Shiyan, Hubei, China, June 2013.

[16] R. Dai, J. Zhang and Y. Wang, Multiple Coarse Grid Acceleration for Multiscale Multigrid Computation. *J. Comput. Appl. Math.*, **269**, 75-85, 2014.

[17] W. Dai, A generalizaed Peaceman-Rachford ADI scheme for solving two-dimensional parabolic differential equations. *J. Sci. Comput.*, **12**, 353-360.

[18] W. Dai and R. Nassar, Compact ADI method for solving parabolic differential equations. *Numer. Methods Partial Differential Equations*, **18**, 129-142, 2002.

[19] P. Deuflhard, P. Leinen, and H. Yserentant, Concepts of an adaptive hierarchical finite element code. *IMPACT Compt. Sci. Engrg.*, **1**, 3-35, 1989.

[20] P. Deuflhard, Cascadic conjugate gradient method for elliptic partial differential equations. In D. Keyes and J. Xu, editors, *Proceedings of the 7th Int'l Conf. on Domain Decomposition Methods*, 29-42, 1993.

[21] H. Ding and Y. Zhang, A new difference scheme with high accuracy and absolute stability for solving convection-diffusion equations. *J. Comput. Appl. Math.*, **230**, 600-606, 2009.

[22] C. C. Douglas and W. L. Miranker, Constructive interference in parallel algorithms. *SIAM J. Numer. Anal.*, **25**, 376-398, 1988.

[23] C. C. Douglas, A review of numerous parallel multigrid methods. *SIAM News*, **25**(3), 1992.

[24] P. O. Frederickson and O. A. McBryan, Parallel superconvergent multigrid. In S. F. McCormick, editor, *Multigrid Methods: Theory, Applications, and Supercomputing*, 195-210, New York, NY, 1987.

[25] D. B. Gannon and J. R. Rosendale, On the structure of parallelism in a highhly concurrent PDE solver. *J. Parallel Distrib. Comput.*, **3**, 106-135, 1986.

[26] L. Ge and J. Zhang, Accuracy, robustness, and efficiency comparison in iterative computation of convection diffusion equation with boundary layers. *Numer. Methods, Partial Differential Eq.*, **16**(4), 379-394, 2000.

[27] L. Ge and J. Zhang, High accuracy iterative solution of convection diffusion equation with boundary layers on nonuniform grids. *J. Comput. Phys.*, **171**, 560-578, 2001.

[28] L. Ge and J. Zhang, Symbolic computation of high order compact difference schemes for three dimensional linear elliptic partial differential equations with variable coefficients. *J. Comput. Appl. Math.*, **143**(1), 9-27, 2002.

[29] Y. Ge and F. Cao, Multigrid method based on the transformation-free HOC scheme on nonuniform grids for 2D convection diffusion problems. *J. Comput. Phys.*, **230**, 4051-4070, 2011.

[30] Y. Ge, Z. F. Tian and J. Zhang, An exponential high-order compact ADI method for 3D unsteady convection-diffusion problems. *Numer. Methods Partial Differential Eq.*, **29**(1), 186-205, 2013.

[31] J. Geiser, Fourth-order splitting methods for time-dependent differential equations. *Numer. Math. Theor. Methodes Appl.*, **1**(3), 321-339, 2008.

[32] M. M. Gupta, R. P. Manohar, and J. W. Stephenson, A single cell high order scheme for the convection-diffusion equation with variable coefficients. *Int. J. Numer. Methods Fluids*, **4**, 641-651, 1984.

[33] M. M. Gupta, J. Kouatchou, and J. Zhang, Comparison of second and fourth order discretization for multigrid Poisson solver. *J. Comput. Phys.*, **132**, 226-232, 1997.

[34] M. M. Gupta, J. Kouatchou, and J. Zhang, A compact multigrid solver for convection-diffusion equations. *J. Comput. Phys.* , **132**, 123-129, 1997.

[35] M. M. Gupta and J. Kouatchou, Symbolic derivation of finite difference approximations for the three dimensional Poisson equation. *Numer. Methods Partial Differential Equations*, **18**(5), 593-606, 1998.

[36] M. M. Gupta and J. Zhang, High accuracy multigrid solution of the 3D convection-diffusion equation. *Appl. Math. Comput.*, **113**(2-3), 249-274, 2000.

[37] Wolfgang Hackbusch, The frequency decomposition multigrid method, part I: Application to anisotropic equations. *Numer. Math.*, **56**, 229-245, 1989.

[38] J. M. Hyman, Mesh refinement and local inversion of elliptic differential equations. *J. Comput. Phys.*, **23**, 124-134, 1977.

[39] J. C. Kalita, D. C. Datal, and A. K. Dass, A class of higher order compact scehmes for the unsteady two-dimensional convection diffusion equation with variable convection coeffcieints. *Int. J. Numer. Meth. Fl.*, **38**, 1111-1131, 2002.

[40] J. C. Kalita, A. K. Dass, and D. C. Dalal, A transformation-free HOC scheme for steady convection-diffusion on nonuniform grids. *Int. J. Numer. Methods Fluids*, **44**, 33-53, 2004.

[41] S. Karaa and J. Zhang, High order ADI method for solving unsteady convection-diffusion problems. *J. Comput. Phys.*, **198**, 1-9, 2004.

[42] Y. Kyei, J. P. Roop, and G. Tang, A family of sixth-order compact finite-difference schemes for the three-dimensional Poisson equation. *Advances in Numerical Analysis*, Vol. 2010, Article ID 352174, 17 pages, 2010.

[43] S. K. Lele, Compact finite difference schemes with spectral-like resolution. *J. Comput. Phys.*, **103**, 16-42,1992.

[44] M. Li, T. Tang, and B. Fornberg, A compact fourth-order finite difference scheme for the steady incompressible Navier-Stokes equations. *Int. J. Numer. Methods Fluids*, **20**, 1137-1151, 1995.

[45] J. Li, Y. Chen and G. Liu, High-order compact ADI methods for parabolic equations. *Comput. Math. Appl.*, **52**, 1343-1356, 2006.

[46] H. Liao and Z. Sun, Maximum norm error bounds of ADI and compact ADI methods for solving parabolic equations. *Numer. Methods Partial Differential Eq.*, **26**(1), 37-60, 2010.

[47] H. Liao and Z. Sun, Maximum norm error estimates of efficient difference schemes for second-order wave equations. *J. Comput. Appl. Math.*, **235**, 2217-2233, 2011.

[48] I. M. Llorente and N. D. Melson, Behavior of plane relaxation methods as multigrid smoothers. *Electron. T. Numer. Ana.*, **10**, 92-114, 2000.

[49] I. M. Llorente, M. Prieto-Matias, and B. Diskin, A parallel multigrid solver for 3D convection and convection-diffusion problems. *Parallel Comput.*, **27**(13), 1715-1741, 2001.

[50] Y. Ma and Y. Ge, A high order finite difference method with Richardson extrapolation for 3D convection diffusion equation. *Appl. Math. Comput.*, **215**, 3408-3417, 2010.

[51] J. M. McDonough, *Basic Computational Numerical Analysis*. Lecture Notes, University of Kentucky, Lexington, KY, 2007.

[52] N. Mishra and Y.Sanyasiraju, Exponential compact higher-order schemes and their stability analysis for unsteady convection-diffusion equations. *Int. J. Comput. Methods*, **11**, 1350053, DOI: 10.1142/S0219876213500539, 2014.

[53] M. Nabavi, M. H. Kamran Siddiqui, and J. Dargahia, A new 9-point sixth-order accurate compact finite-difference method for the Helmholtz equation. *J. Sound Vibrat.*, **307**, 972-982, 2007.

[54] B. J. Noye and H. H. Tan, Finite difference methods for solving the two-dimensional advection diffusion equation. *Int. J. Numer. Methods Fluids*, **26**, 1615-1629, 1988.

[55] D. W. Peaceman and H. H. Rachford Jr., Thenumerical solution of parabolic and elliptic differential equations. *J. Soc. Ind. Appl. Math*, **3**, 28-41, 1959.

[56] L. F. Richardson, The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Trans. Roy. Soc. London, Ser. A*, **210**, 307-357, 1910.

[57] S. A. Richards, Competed Richardson extrapolation in space and time. *Commun. Numer. Methods Eng.*, **13**, 573-582, 1997.

[58] A. Rigal, High order difference schemes for unsteady one-dimensional diffusion-convection problems. *J. Comput. Phys.*, **114**, 59-76, 1994.

[59] P. J. Roach, *Computational Fluid Dynamics*, Hermosa, Albuquerque, MN, 1976.

[60] P. J. Roache and P. M. Knupp, Completed Richardson Extrapolation. *Commun. Numer. Methods Eng.*, **9**, 365-374, 1993.

[61] Y. Saad, *Iterative Methods for Sparse Linear Systems, 2nd edition*. SIAM, Philadelphia, PA, 2003.

[62] Scalable Linear Solver. *https* : *//computation.llnl.gov/casc/linear_solvers/*.

[63] R. U. Seydel, *Tools for Computational Finance, 5th edition*. Springer, London, 2012.

[64] V. V. Shaidurov, Some estimates of the rate of convergence for the cascadic conjugate gradient method. *Comput. Math. Appl.*, **31**, 161-171, 1996.

[65] W. F. Spotz, *High-Order Compact Finite Difference Schemes for Computational Mechanics*. PhD thesis, University of Texas at Austin, Austin, TX, 1995.

[66] W. F. Spotz and G. F. Carey, High-order compact scheme for the steady stream-function vorticity equations. *Int. J. Numer. Methods Engrg.*, **38**, 3497-3512, 1995.

[67] W. F. Spotz and G. F. Carey, A high-order compact formulation for the 3D Poisson equation. *Numer. Methods Partial Differential Eq.*, **12**, 235-243, 1996.

[68] W. F. Spotz and G. F. Carey, Extension of high-order compact schemes to time-dependent problems. *Numer. Methods Partial Differential Eq.*, **17**, 657-672, 2001.

[69] H. Sun and J. Zhang, A high order finite difference discretization strategy based on extrapolation for convection diffusion equations. *Numer. Methods Partial Differential Eq.*, **20**(1), 18-32, 2004.

[70] G. Sutmann and B. Steffen, High-order compact solvers for the three-dimensional Poisson equation. *J. Comput. Appl. Math.*, **187**(2), 142-170, 2006.

[71] G. Sutmann, Compact finite difference schemes of sixth order for the Helmhotz equation. *J. Coput. Appl. Math.*, **203**(1), 15-31, 2007.

[72] J. W. Thomas, Numerical Partial Differential Equations: Finite Difference Methods. Springer, New York, 1995.

[73] C. Thole and U. Trottenberg, Basic smoothing procedures for the multigrid treatment of elliptic 3D operators. *Appl. Math. Comput.*, **19**, 333-345, 1986.

[74] Z. F. Tian and Y. B. Ge, A fourth-order compact ADI method for solving two-dimensional unsteady convection-diffusion problems. *J. Comput. Appl. Math.*, **198**, 268-286, 2007.

[75] Z. F. Tian, A rational high-order compact ADI method for unsteady convection-diffusion equations. *Comp. Phys. Commun.*, **182**, 649-662, 2011.

[76] Z. F. Tian and P. X. Yu, A high-order expoential scheme for solving 1D unsteady convection-diffusion equations. *J. Comput. Appl. Math.*, **235**, 2477-2491, 2011.

[77] E. Turkel, D. Gordon, R. Gordon, and S. Tsynkov, Compact 2D and 3D sixth order schemes for the Helmholtz equation with variable wave number. *J. Comput. Phys.*, **232**, 272-287, 2013.

[78] Y. Wang, *High Accuracy Multiscale Multigrid Computation for Partial Differential Equations*. PhD thesis, University of Kentucky, Lexington, KY, 2010.

[79] Y. Wang and J. Zhang, Sixth order compact scheme combined with multigrid method and extrapolation technique for 2D Poisson equation. *J. Comput. Phys.*, **228**(1), 137-146, 2009.

[80] Y. Wang and J. Zhang, Fast and robust sixth-order multigrid computation for the three dimensional convection-diffusion equation. *J. Comput. Appl. Math.*, **234**(12), 3496-3506, 2010.

[81] Y. Wang and J. Zhang, Integrated fast and high accuracy computation of convection diffusion equations using multiscale multigrid method. *Numer. Methods Partial Differential Eq.*, **27**(2), 399-414, 2011.

[82] P. Wesseling, *An Introduction to Multigrid Methods*. Wiley, Chichester, England, 1992.

[83] D. You, A high-order padé ADI method for unsteady convection-diffusion equations. *J. Comput. Phys.*, **214**, 1-11, 2006.

[84] J. Zhang, *Multigrid Acceleration Techniques and Applications to the Numerical Solution of Partial Differential Equations*. PhD thesis, The George Washington University, Washington, DC, 1997.

[85] J. Zhang, Residual scaling techniques in multigrid, I: equivalence proof. *Appl. Math. Comp.*, **86**(2-3):283-303, 1997.

[86] J. Zhang, Accelerated high accuracy multigrid solution of the convection-diffusion equation with high Reynolds number. *Numer. Methods Partial Differential Eq.*, **77**, 73-89, 1997.

[87] J. Zhang, Residual scaling techniques in multigrid, II: practical applications. *Appl. Math. Comp.*, **90**(2-3):229-252, 1998.

[88] J. Zhang, An explicit fourth-order compact finite difference scheme for three dimensional convection-diffusion equation. *Commun. Numer. Methods Engrg.*, **14**, 209-218, 1998.

[89] J. Zhang, Fast and high accuracy multigrid solution of the three dimensional Poisson equation. *J. Comput. Phys.*, **143**, 449-461, 1998.

[90] J. Zhang, On convergence and performance of iterative methods with fourth-order compact schemes. *Numer. Methods Partial Differential Eq.*, **14**, 262-238, 1998.

[91] J. Zhang, L. Ge, and J. Kouatchou, A two colorable fourth-order compact difference scheme and parallel iterative solution of the 3D convection diffusion equation. *Math. Comput. Simulation*, **54**(1-3), 65-80, 2000.

[92] J. Zhang, A note on accelerated high accuracy multigrid solution of convection-diffusion equation with high Reynolds number. *Numer. Methods Partial Differential Eq.*, **16**(1):1-10, 2000.

[93] J. Zhang, Multigrid method and fourth order compact difference scheme for 2D Poisson equation with unequal meshsize discretization. *J. Comput. Phys.*, **179**, 170-179, 2002.

[94] J. Zhang and L. Ge, Symbolic computation of high order compact difference schemes for three dimensional linear elliptic partial differential equations with variable coefficients. *J. Comput. Appl. Math.*, **143**(1), 9-27, 2002.

[95] J. Zhang, H. Sun and J. J. Zhao, High order compact scheme with multigrid local mesh refinement procedure for convection diffusion problems. *Comput. Methods Appl. Mech. Engrg.*, **191**(41-42), 4661-4674, 2002.

[96] J. Zhang and J. J. Zhao, Truncation error and oscillation property of the combined compact difference scheme. *Appl. Math. Comput.*, **161**(1), 241-251, 2005.

[97] J. Zhao, W. Dai and T. Niu, Fouth-order compact schemes of a heat conduction problem with Neumann boundary conditions. *Numer. Methods Partial Differential Eq.*, **23**, 949-959, 2007.

[98] J. Zhao, W. Dai and S. Zhang, Fourth order compact schemes for solving multidimensional heat problems with Neumann boundary conditions. *Numer. Methods Partial Differential Eq.*, **24**(1), 165-178, 2008.

[99] X. Zhi and Z. Lu, Universal parallel numerical computing for 3D convection-diffusion equation with variable coefficients. *Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing*, Beijing, China, Oct. 2002.

[100] J. Zhu, *Solving Partial Differential Equations on Parallel Computers.* World Scientific Publishing, River Edge, NJ, 1994.

[101] Z. Zlatev, I. Dimov, I. Faragó, K. Georgiev, Á. Havasi and T. Ostromsky, Solving advection equations by applying the Crank-Nicolson scheme combined with the Richardson extrapolation. *Intl. J. Differential Eq.*, DOI:10.1155/2011/520840, 2011.

Vita

Personal Data:

Name: Ruxin Dai

Place of Birth: Nanchang, Jiangxi, China

Educational Background:

- Master of Science in Computer Science, University of Kentucky, 2013.
- Master of Engineering in Computer Application Technology, East China University of Science and Technology, Shanghai, China, 2010.
- Bachelor of Engineering in Computer Science and Technology, Hunan University, Changsha, China, 2007.

Professional Experience:

- Teaching Assistant, 01/2010 - 05/2014. Department of Computer Science, University of Kentucky, Lexington, KY, USA.
- Research Assistant, 09/2008 - 06/2009. Software Testing Center, East China University of Science and Technology, Shanghai, China.
- Teaching Assistant, 09/2007 - 06/2008. College of Information Science and Engineering and College of Science, East China University of Science and Technology, Shanghai, China.
- Undergraduate Research Assistant, 12/2005-12/2006. College of Computer and Communication, Hunan University, Changsha, China.

Awards:

- Computer Science Department Thaddeus B. Curtz Memorial Scholarship award, University of Kentucky, 2014.
- Second prize in the 26th Annual Eastern Kentucky University Symposium in Mathematical, Statistical and Computer Sciences, Richmond, KY, March 2013.
- Student travel support from Graduate School Fellowship, University of Kentucky, 2013 and 2010.
- Fellowship from Lawrence Berkeley National Laboratory to attend the 13th DOE Advanced Computational Software (ACTS) Collection Workshops, Berkeley, CA, August 2012.
- Student travel support from Computing Research Association-Women (CRA-W) to attend Grad Cohort Workshop, 2011 and 2010.
- Outstanding Teaching Assistant, East China University of Science and Technology, 2008.
- Second prize in Designing Innovative Graduation Project, Hunan University, 2007.
- Hugo Shong Fellowship, Hunan University, 2006.
- Academician Sun Jia-guang Fellowship, Hunan University, 2005.
- Youth Leadership Honor, Hunan University, 2006.
- Outstanding Student Fellowship, Hunan University, 2004-2006.

Refereed Publications:

- **Ruxin Dai**, Jun Zhang, and Yin Wang. Multiple Coarse Grid Acceleration for Multiscale Multigrid Computation, *Journal of Computational and Applied Mathematics*, DOI:10.1016/j.cam.2014.03.021, 2014.

- Yin Wang, Su Yu, **Ruxin Dai** and Jun Zhang, A 15 Point High Order Compact Scheme with Multigrid Computation for Solving 3D Convection Diffusion Equations, *International Journal of Computer Mathematics*, DOI:10.1080/00207160.2014.893296, 2014.

- **Ruxin Dai**, Yin Wang, and Jun Zhang. Fast and High Accuracy Multiscale Multigrid Method with Multiple Coarse Grid Updating Strategy for 3D Convection Diffusion Equation, *Computers & Mathematics with Applications*, 66:542-559, 2013.

- **Ruxin Dai**, Jun Zhang, and Yin Wang. Sixth Order Compact Approximation with Completed Richardson Extrapolation, *2013 International Conference on Computational and Information Sciences*, Shiyan, Hubei, China, June 2013.

- Jun Zhang, Xinyu Geng, and **Ruxin Dai**. Analysis on Two Approaches for High Order Accuracy Finite Difference Computation, *Applied Mathematics Letters*, 25:2081-2085, 2012.

- **Ruxin Dai**, Yin Wang, Jun Zhang and Yongbin Ge. Comparison of Two Higher Order Compact Computation Strategies for Handling Boundary Layers, emph15th Copper Mountain Conference on Multigrid Method, Copper Mountain, Colorado, March 2011.

- **Ruxin Dai** and Chunhua Gu. An Improved Test Case Prioritization Algorithm for Black Box Test (Chinese), Computer Engineering and Design, 31(20):4343-4346, 2010.