2010

HIGH ACCURACY MULTISCALE MULTIGRID COMPUTATION FOR PARTIAL DIFFERENTIAL EQUATIONS

Yin Wang
University of Kentucky, ywang12@ltu.edu

ABSTRACT OF DISSERTATION

Yin Wang

The Graduate School

University of Kentucky

2010

HIGH ACCURACY MULTISCALE MULTIGRID COMPUTATION FOR PARTIAL DIFFERENTIAL EQUATIONS

ABSTRACT OF 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

Yin Wang

Lexington, Kentucky

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

Lexington, Kentucky

2010

ABSTRACT OF DISSERTATION

HIGH ACCURACY MULTISCALE MULTIGRID COMPUTATION FOR PARTIAL DIFFERENTIAL EQUATIONS

Scientific computing and computer simulation play an increasingly important role in scientific investigation and engineering designs, supplementing traditional experiments, such as in automotive crash studies, global climate change, ocean modeling, medical imaging, and nuclear weapons. The numerical simulation is much cheaper than experimentation for these application areas and it can be used as the third way of science discovery beyond the experimental and theoretical analysis. However, the increasing demand of high resolution solutions of the Partial Differential Equations (PDEs) with less computational time has increased the importance for researchers and engineers to come up with efficient and scalable computational techniques that can solve very large-scale problems. In this dissertation, we build an efficient and highly accurate computational framework to solve PDEs using high order discretization schemes and multiscale multigrid method.

Since there is no existing explicit sixth order compact finite difference schemes on a single scale grids, we used Gupta and Zhang's fourth order compact (FOC) schemes on different scale grids combined with Richardson extrapolation schemes to compute the sixth order solutions on coarse grid. Then we developed an operator based interpolation scheme to approximate the sixth order solutions for every find

grid point. We tested our method for 1D/2D/3D Poisson and convection-diffusion equations.

We developed a multiscale multigrid method to efficiently solve the linear systems arising from FOC discretizations. It is similar to the full multigrid method, but it does not start from the coarsest level. The major advantage of the multiscale multigrid method is that it has an optimal computational cost similar to that of a full multigrid method and can bring us the converged fourth order solutions on two grids with different scales. In order to keep grid independent convergence for the multiscale multigrid method, line relaxation and plane relaxation are used for 2D and 3D convection diffusion equations with high Reynolds number, respectively. In addition, the residual scaling technique is also applied for high Reynolds number problems.

To further optimize the multiscale computation procedure, we developed two new methods. The first method is developed to solve the FOC solutions on two grids using standard W-cycle structure. The novelty of this strategy is that we use the coarse level grid that will be generated in the standard geometric multigrid to solve the discretized equations and achieve higher order accuracy solution. It is more efficient and costs less CPU and memory compared with the V-cycle based multiscale multigrid method.

The second method is called the multiple coarse grid computation. It is first proposed in superconvergent multigrid method to speed up the convergence. The basic idea of multigrid superconvergent method is to use multiple coarse grids to generate better correction for the fine grid solution than that from the single coarse grid. However, as far as we know, it has never been used to increase the order of solution accuracy for the fine grid. In this dissertation, we use the idea of multiple coarse grid computation to approximate the fourth order solutions on every coarse grid and fine grid. Then we apply the Richardson extrapolation for every fine grid point to get the sixth order solutions.

For parallel implementation, we studied the parallelization and vectorization potential of the Gauss-Seidel relaxation by partitioning the grid space with four colors for solving 3D convection-diffusion equations. We used OpenMP to parallelize the loops in relaxation and residual computation. The numerical results show that the parallelized and the sequential implementation have the same convergence rate and the accuracy of the computed solutions.

KEYWORDS: Partial differential equations, multigrid method, finite difference method, Richardson extrapolation

Yin Wang

November 30, 2010

HIGH ACCURACY MULTISCALE MULTIGRID COMPUTATION FOR PARTIAL DIFFERENTIAL EQUATIONS

By

Yin Wang

Director of Dissertation

Director of Graduate Studies

RULES FOR THE USE OF DISSERTATIONS

Unpublished dissertations submitted for the Doctor's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgements.

Extensive copying or publication of the dissertation in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this dissertation for use by its patrons is expected to secure the signature of each user.

Name	Date

DISSERTATION

Yin Wang

The Graduate School

University of Kentucky

2010

HIGH ACCURACY MULTISCALE MULTIGRID 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

Yin Wang

Lexington, Kentucky

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

Lexington, Kentucky

2010

ACKNOWLEDGEMENTS

It is a pleasure to thank those who made this thesis possible. I would never have been able to finish my dissertation without the guidance of my committee members, help from friends, and support from my family.

First of all, I would like to express my deepest gratitude to my advisor, Dr. Jun Zhang, for his excellent guidance, patience, caring, and leading me to do research in scientific computing and computer simulation. Dr. Zhang has been a great mentor on every account, and his broad knowledge and constructive suggestions to this dissertation are sincerely appreciated.

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

Also, I would like to thank Dr. Debby Keen, Dr. Grzegorz Wasilkowski, Dr. Jeonghwa Lee, and Dr. Wensheng Shen for their kind assistance with writing me 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, Dr. Shuting Xu, Dr. Ning Kang, Dr. Eun-Joo Lee, Dr. Jie Wang, Mr. Ning Cao, Mr. Hao Ji, Mr. Dianwei Han, Mr. Cheng Qin, Mr. Zhenmin Lin, Mr. Qi Zhuang, Mr. Changjiang Zhang, Mr. Xuwei Liang, Mr. Lian Liu, Mr. Xiwei Wang, and Ms. Ruxin Dai for their kind help.

Finally I would like to thank my family members. I need to thank my parents for giving my life and their continuous support. Most important, I would like to thank

my wife, Jiaxi Wang, for her endless love and great patience during my study at the University of Kentucky.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

The area of scientific computing and computer simulation is to construct mathematical models and quantitative analysis techniques and use computers to analyze and solve scientific problems. It plays an increasingly important role in scientific investigations and engineering designs, supplementing traditional experiments. Since many simulations and modeling applications in computational science and engineering and industry (CSEI), such as automotive crash studies, human brain medical imaging, global climate change, and ocean modeling, can be formulated in the form of partial differential equations (PDEs), the numerical solution of systems of PDEs on distributed memory supercomputers is one of the most challenging and long-standing problems. Over the last several decades, computational mathematicians and engineers have developed many efficient fast algorithms to reduce the computation time. However, the increasing demand for higher resolution simulations in less computer time have continuously challenged the computational scientists to come up with more efficient, scalable numerical algorithms to solve PDEs.

A typical way to solve PDEs is to discretize them, changing a continuous problem into a discrete problem, then solve the arising linear systems. Traditionally, numerical solutions to PDEs have been studied by two camps of researchers. One works on discretization, the other belongs to numerical linear algebra. This dissertation mainly focuses on combining these two approaches together using high order discretization schemes with scalable linear system solvers to build a computational framework for

solving PDEs.

1.1 Classification of PDEs

Partial differential equations can be used to formulate, and thus aid the solution of, problems involving functions of several variables. They have broad applications in mechanical engineering, theoretical physics, fluid dynamics and other fields. Before computer simulations, the PDEs governing the continuous problems need to be represented and evaluated as algebraic equations.

A general two dimensional (2D) second order partial differential equation is in the form of

$$A\frac{\partial^2 u}{\partial x^2} + 2B\frac{\partial^2 u}{\partial xy} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu = g(x, y), \tag{1.1}$$

where the coefficients A, B, C, D, E and F may depend on x and y. Such equations can be classified as parabolic ($B^2 = AC$), hyperbolic ($B^2 > AC$) and elliptic ($B^2 < AC$).

Equations that are parabolic at every grid point can be transformed into a form analogous to the heat equation by changing the independent variables. Hyperbolic equations retain any discontinuities of functions or derivatives in the initial data; an example is wave equation. Solutions of elliptic PDEs are as smooth as the coefficients allow within the interior of the region where the equation and solution are defined.

In this dissertation, I consider efficient numerical solutions for elliptic equations like Poisson equation and convection-diffusion equation. Detailed information regarding the classification of PDEs can be found in [4].

1.2 Discretization of PDEs

To discretize PDEs, the following methods are frequently used: the finite difference methods, the finite volume methods, and the finite element methods. Compared to

the other two methods, the finite difference methods are easier to implement and higher order accuracy can be obtained by using higher order finite difference schemes [57]. But the finite difference methods may not be able to handle irregular domain problems very well. In many finite volume and finite element methods, the partial differential equations need to be written as an integral equation before discretization. Both the finite element and the finite volume methods are more convenient to be used for unstructured meshes, but the associated computation time increases.

Selection of a discretization method for solving a particular PDE depends on the problem itself. The finite difference method is very useful for simple geometry domain. The finite volume method is used in many computational fluid dynamics packages. The finite element method is a good choice for solving partial differential equations over complex domains (like cars and oil pipelines), when the domain changes (as during a solid state reaction with a moving boundary), or when the desired precision varies over the entire domain. In this dissertation, I consider the simple geometry cases, as I am mainly interested in developing a computational methodology associated with the finite difference discretization techniques. For example, the rectangular domain for the 2D PDEs will be assumed in this study. So I will focus my attention on the finite difference method.

High order finite difference schemes. In many CSEI applications, such as in the global ocean modeling and wide area weather forecasting, the computational domains are huge and the grid spaces are not small. In the context of the finite difference methods, the standard second order discretization schemes or the first order upwind difference schemes yield unsatisfactory results because they may need fine mesh griddings to compute approximate solutions of acceptable accuracy. The resulting large size linear systems have to be solved, which may consume a lot of memory space and CPU time. In addition, the second order scheme may also produce numerical

solutions with nonphysical oscillations for the convection dominated problems.

Higher order (more than two) discretization methods are considered to be useful to reduce computational cost in very large scale modelings and simulations, which use relatively coarser mesh griddings to yield approximate solutions of comparable accuracy, compared with lower order discretization. Generally, higher order discretization schemes need more complicated procedures and more preprocessing costs to construct the coefficient matrix, but they usually yield linear systems of much smaller size, compared with those from the lower order methods.

Due to the advantage of higher order discretization methods, they have been used to develop computational tools for solving various application problems [17, 20, 40]. In the past two decades, there has been growing interest in developing higher order accurate discretization methods, especially the higher order compact difference schemes, to solve PDEs [32, 51, 55, 75, 77, 78]. I call them "compact" because these schemes only use the minimum three grid points in one dimension in the discretization formulas. Errors that occur due to truncations in these compact difference schemes are usually four to six times smaller than that of the non-compact schemes of the same order [1].

For the development of high order compact difference schemes, there have been two main tracks. The first one is best outlined in the Lele's paper [32] and in the paper by Chu and Fan [14, 15]. Chu and Fan proposed a three point combined compact difference (CCD) scheme for solving two dimensional Stommel Ocean model, which is a special two dimensional convection-diffusion equation. They used Hermitian polynomial approximation to achieve sixth order accuracy for the inner grid points and fifth order accuracy for the boundary grid points. The advantage of the CCD scheme is that it can be used to solve many types of PDEs without major modifications. This is because they do not approximate the PDEs directly, instead, they approximate the solution variables and the derivative variables. The Alternating Direction Implicit

(ADI) [43] method can be used to reduce the higher dimensional problems to a series of lower dimensional problems. So, their schemes are referred to as implicit high order compact schemes because they do not compute the solution variables of the PDEs directly. Instead, the first derivative and the second derivative of the solution variables are also computed at the same time.

The major disadvantage of these implicit schemes is that the approximations of the first and second derivatives are unnecessarily computed for some applications without the need of derivative approximations. In addition, the CCD scheme has a stability problem that for certain problems, if a large mesh size is used, the computed solution may be oscillatory [82]. Numerical oscillations may be avoided by using finer meshsize. However, the use of fine mesh discretization is in contrary to the motivation of using higher order compact schemes.

In contrary, the explicit fourth order compact schemes [24, 27, 33, 37, 55] compute the solution of the variables directly; no redundant computation is needed. For stability, it has been proved that the computed solutions for a 1D convection-diffusion model problem is nonoscillatory [55]. In addition, Some accelerating iterative methods like multigrid method and preconditioned iterative methods have been used to efficiently solve the resulting sparse linear systems arising from the high order compact finite difference discretizations [71, 76, 77]. From [56] I know that some explicit fourth order schemes are stable and will suppress the nonphysical oscillations. However, the higher order explicit compact schemes are more complicated to develop in higher dimensions [26, 81], compared with the implicit compact schemes. As far as I know, there is no existing explicit compact scheme on a single scale grid that has higher than the fourth order accuracy [59].

Multiscale grid computation. Computation of high accuracy solution on a uniscale is inefficient and sometimes impractical for modeling very large scale problems.

Multiscale computation (MSC) is appeared to be the only route to take for developing efficient and scalable computational framework to handle very large scale CSEI simulations. MSC is concerned with methods for computing, manipulating, and analyzing information at different resolution levels. It is widely used in many areas and it appears under several different names like multi-resolution analysis in wavelet theory, compression in signal analysis, progressive meshing in computer graphics, and clustering in database. In this dissertation, a sixth order compact finite difference scheme using multiscale grid techniques combined with Richardson extrapolation is developed to solve PDEs on two grids with different scales.

1.3 Linear System Solver

In computing the numerical solution of PDEs, it is also important that the resulting linear systems arising from the high order compact discretizations

$$Ax = b \tag{1.2}$$

be solved efficiently. Here, A is a real (or complex) valued sparse matrix of order n, x is an unknown n-vector, and b is a known n-vector.

1.3.1 Direct methods

Eq. (1.2) can be solved by direct methods and iterative methods. Direct methods, based on the factorization of the coefficient matrix A into invertible matrices, allow exact computation up to the machine accuracy. However, direct methods, like Gaussian elimination, often become inefficient for solving large scale problems. The reason is that the Gaussian elimination has a computational complexity of $O(n^3)$ on an $n \times n$ matrix and the problems arising from the discretization of PDEs like the 3D convection-diffusion equation sometimes lead to a linear system with hundreds

of millions of unknowns. In addition, the matrices from discretized PDEs are often sparse, and special storage strategy can be used to store only the nonzeros of the sparse matrices. But the LU factorization in direct methods can create fill-ins during factorization and hence does not take advantage of the sparsity of the original matrices.

1.3.2 Iterative methods

In contrary to direct methods, iterative methods attempt to solve a problem by finding successive approximations to the solution starting from an initial guess. Iterative methods are often useful for even solving linear problems involving a large number of variables, where direct methods would be prohibitively expensive even with the best available computing power. The purpose of this subsection is to give a literature review of some basic iterative methods, including Jacobi method, Gauss-Seidel method, SOR method as well as two other efficient iterative methods: Krylov subspace based iterative methods and multigrid method. We list these methods in Fig. 1.1

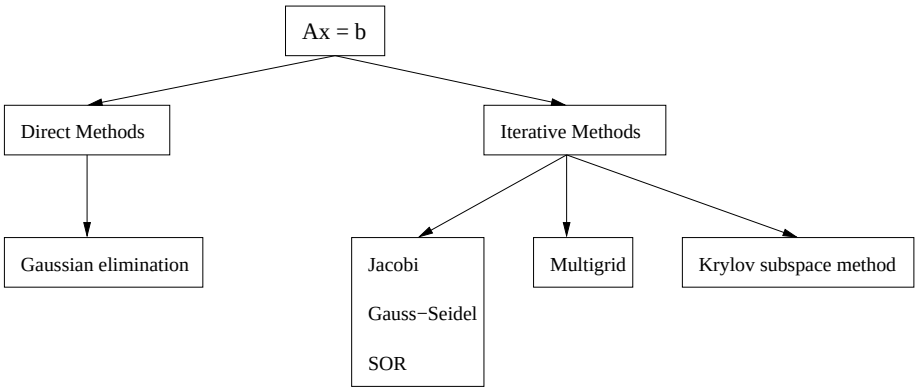


Figure 1.1: Different linear system solvers.

Basic iterative methods. The basic iterative methods like Jacobi and Gauss-Seidel methods are based on relaxation of the unknowns [6]. Beginning with a given approximate solution, these methods modify the components of the approximation,

one or a few at a time and in a certain order, until convergence is reached. Each of these modifications, called relaxation steps, is aimed at annihilating one or more components of the residual vector.

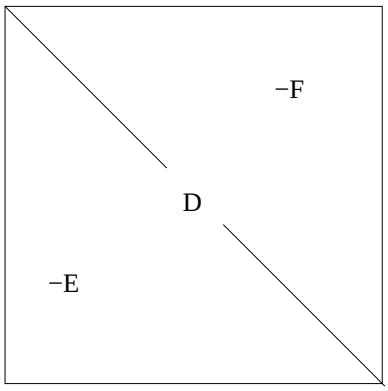


Figure 1.2: Partition of the coefficient matrix A.

For solving Eq. (1.2), the coefficient matrix A can be decomposed into three parts [50] as shown in Fig. 1.2

$$A = D - E - F,$$

where D is the diagonal of A, $-E$ and $-F$ are its lower triangular part and upper triangular part, respectively.

For the Jacobi iteration, the i-th component of vector x at $(k+1)st$ iteration can be written in the form of

$$x_i^{(k+1)} = \frac{1}{A_{ii}}(b_i - \sum_{j=1, j\neq i}^{n} A_{ij}x_j^{(k)}), \qquad A_{ii} \neq 0,$$

which can also be written in matrix form as

$$x^{(k+1)} = D^{-1}(E+F)x^{(k)} + D^{-1}b.$$

For the Gauss-Seidel iteration, the approximate solution is updated immediately after the new component is determined. The newly computed components can be

changed within a working vector which is redefined at each relaxation step. So, the Gauss-Seidel iteration is in the form of

$$x_i^{(k+1)} = \frac{1}{A_{ii}}(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} A_{ij}x_j^{(k)}),$$

or

$$x^{(k+1)} = (D-E)^{-1}Fx^{(k)} + (D-E)^{-1}b.$$

Even faster convergence can be obtained using successive over relaxation (SOR) method following the Gauss-Seidel iteration as

$$x_i^{(k+1)} = \omega x_i^{(k+1)} + (1-\omega)x_i^{(k)},$$

where ω is the SOR parameter and $0 < \omega < 2$.

From above formulas, we can find out that both the Jacobi and the Gauss-Seidel iterations can be written in matrix form

$$x^{(k+1)} = Gx^{(k)} + f \tag{1.3}$$

with

$$G_{Jacobi} = D^{-1}(E+F)$$

$$G_{Gauss-Seidel} = (D-E)^{-1}F.$$

If the iteration process converges to x, then

$$x = Gx + f. \tag{1.4}$$

Let $x^{(*)}$ be a solution to Eq. (1.4), subtracting Eq. (1.4) from Eq. (1.3) we get

$$\begin{aligned} x^{(k+1)} - x^{(*)} &= G(x^{(k)} - x^{(*)}) \\ &= G^2(x^{(k-1)} - x^{(*)}) \\ &= ... \\ &= G^{k+1}(x^{(0)} - x^{(*)}) \end{aligned}$$

Hence, in order for the iteration to converge, the spectral radius of matrix G must satisfy

$$\rho(G) \leq ||G|| < 1.$$

Krylov subspace methods. The Krylov subspace methods, such as General Minimum Residual (GMRES) [48] and Bi-Conjugate Gradient Stabilized (BiCGSTAB) [64] were designed to solve general sparse linear systems. The GMRES method is a projection method to approximate the solution by the vector in a Krylov subspace with minimum residual. Arnoldi iteration [50] is used to find this vector. BiCGSTAB method was proposed by van der Vorst with the purpose of extending the Conjugate Gradient (CG) algorithm to non-symmetric matrices.

Krylov subspace methods do not require discretized partial differential equations and therefore can be applied in much more general situations where other iterative methods like multigrid methods do not work or perform poorly. However, the drawback of the Krylov subspace methods are that their convergence rates are usually dependent on the size of the linear systems and in some cases they are not robust. A common strategy to improve robustness is to use a suitable preconditioner before a Krylov subspace method is applied [50].

A good preconditioner can transfer the original linear system to a new "easier to

solve" linear system like

$$M^{-1}Ax = M^{-1}b,$$

where M^{-1} is the preconditioner and should be inexpensive to apply to a matrix-vector product. In many CSEI applications, it has been recognized that the choice of the preconditioner is more important than the choice of a Krylov subspace method in designing an iterative solution method [49]. Most existing preconditioners are based on the sparse approximate inverse (SAI) and incomplete LU (ILU) factorization of the coefficient matrix and these ILU preconditioners do not have grid-independent convergence rate [50]. Stabilized singular value decomposition (SVD) is also a good technique to construct an efficient preconditioner, especially for some applications in computational electromagnetics [70].

Multigrid methods. Convergence of the basic iterative methods like Jacobi and Gauss-Seidel methods slow down after a few iterations. That is because the basic iterative methods remove high frequency errors efficiently, but are inefficient for low frequency errors. High frequency errors are rough errors and low frequency errors are smooth errors, as shown in Fig. 1.3. Hence, basic iterative methods (relaxation methods) are said to have smoothing effect (smoother). In order to speed up the convergence when the error becomes smooth after a few iterations, the error can be projected to a coarse grid as it becomes rough on a coarse scale grid as in Fig. 1.4.

Based on the above idea, multigrid algorithm iterates on a hierarchy of successively coarser grids until the convergence is reached. For problems such as the Poisson equation on a rectangular domain, the convergence rate of the multigrid method is independent of the grid size [9, 12, 65]. Various multigrid implementation strategies with the fourth order compact schemes to solve the 2D and 3D Poisson equations or other PDEs like the convection-diffusion equations are discussed in [25, 27, 52].

One iteration of a general multigrid cycling procedure includes smoothing the error

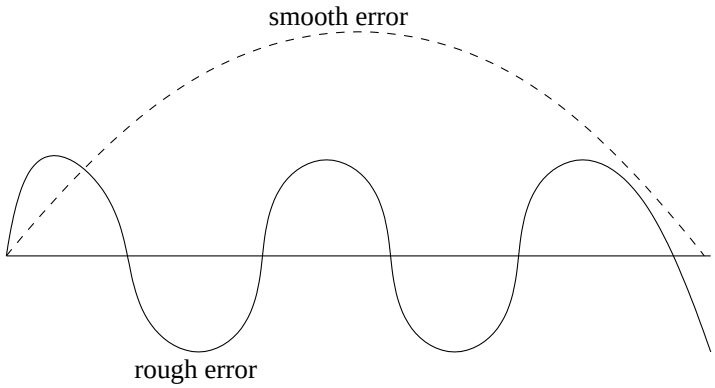


Figure 1.3: Error distribution.

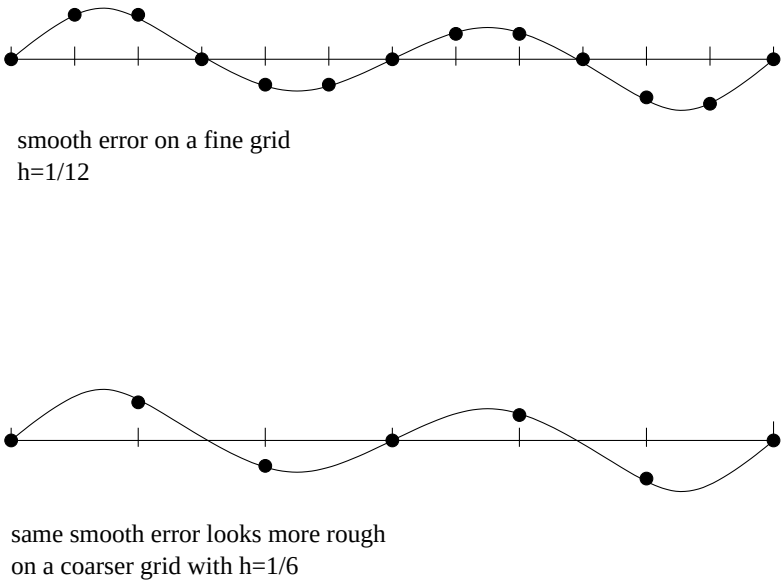


Figure 1.4: Same error distribution on two scale grids.

using a basic iterative method, which is also called the smoother. This procedure is done by restricting the residuals to the coarse grid, relaxing the error equation on the coarse grid, prolongating the coarse grid error correction to the fine grid, and adding the error correction to the fine grid solution. Relaxation scheme and coarse grid correction scheme are complimentary to each other. Relaxation on the fine grid eliminates rough (oscillatory) error components, leaving relatively smooth error components to be solved on the coarse grid. Intergrid transfer operators and coarse grid relaxation work well on smooth error components. They together remove both oscillatory and smooth error components.

Multigrid method is a recursive method in that the coarse grid computation is also carried out by the multigrid idea. A multigrid V-cycle is the computational process that goes from the fine grid down to the coarsest grid and then comes back from the coarsest grid up to the fine grid. The linear system solver developed in this dissertation is based on V-cycle on different scale grids. A V-cycle algorithm may include the following steps [50]:

Algorithm 1 Multigrid V-cycle Algorithm

1: Relax ν_1 times on $A_h u_h = b_h$ with initial guess $u_h^{(k)}$
2: Compute $b_{2h}^{(k)} = r_{2h}^{(k)} = I_h^{2h}(b_h - A_h u_h^{(k)})$
3: Relax ν_1 times on $A_{2h} u_{2h}^{(k)} = b_{2h}^{(k)}$ with initial guess 0
4: Compute $b_{4h}^{(k)} = r_{4h}^{(k)} = I_{2h}^{4h}(b_{2h} - A_{2h} u_{2h}^{(k)})$
5: Relax ν_1 times on $A_{4h} u_{4h}^{(k)} = b_{4h}^{(k)}$ with initial guess 0
6: Compute $b_{8h}^{(k)} = r_{8h}^{(k)} = I_{4h}^{8h}(b_{4h} - A_{4h} u_{4h}^{(k)})$
7: .....
8: Solve $A_{Lh} u_{Lh}^{(k)} = b_{Lh}^{(k)}$
9: .....
10: Correct $u_{4h}^{(k*)} = u_{4h}^{(k)} + I_{8h}^{4h} u_{8h}^{(k+1)}$
11: Relax ν_2 times on $A_{4h} u_{4h} = b_{4h}$ with initial guess $u_{4h}^{(k*)}$
12: Correct $u_{2h}^{(k*)} = u_{2h}^{(k)} + I_{4h}^{2h} u_{4h}^{(k+1)}$
13: Relax ν_2 times on $A_{2h} u_{2h} = b_{2h}$ with initial guess $u_{2h}^{(k*)}$
14: Correct $u_h^{(k*)} = u_h^{(k)} + I_{2h}^{h} u_{2h}^{(k+1)}$
15: Relax ν_2 times on $A_h u_h = b_h$ with initial guess $u_h^{(k*)}$

In Algorithm 1, I_{2h}^{h} and I_{h}^{2h} are two intergrid transfer operators used to transfer information between grids. From coarse grid to fine grid: linear interpolation (prolongation) operator is used as I_{2h}^{h}, where, for one dimensional problem and $n = 8$, we have

$$I_{2h}^{h} u_{2h} = \frac{1}{2} \begin{pmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \\ & & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}_{2h} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{pmatrix}_{h} = u_h.$$

From fine grid to coarse grid: the simplest restriction operator is injection, which takes the fine grid point values directly. A more accurate restriction operator is called full weighting. It transfers a weighted average of value at neighboring fine grid points. For 1D problem, $n = 8$, the full weighting has the form

$$I_{h}^{2h} r_h = \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \end{pmatrix}_{h} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}_{2h} = r_{2h}.$$

Note that relaxations are performed before and after the coarse grid correction. They are called (ν_1) presmoothing sweeps and (ν_2) postsmoothing sweeps. Relaxation scheme and coarse grid correction scheme are complimentary to each other. Relaxation on the fine grid eliminates rough (oscillatory) error components, leaving

relatively smooth error components to be solved on the coarse grid.

W-cycle is the multigrid method with two corrections. We may also start with the coarsest grid in order to provide a good initial guess for the finer grids. Such an algorithm is called the full multigrid V-cycle algorithm[50]. The structure of the computation flow of these three multigrid schemes is shown in Fig. 1.5

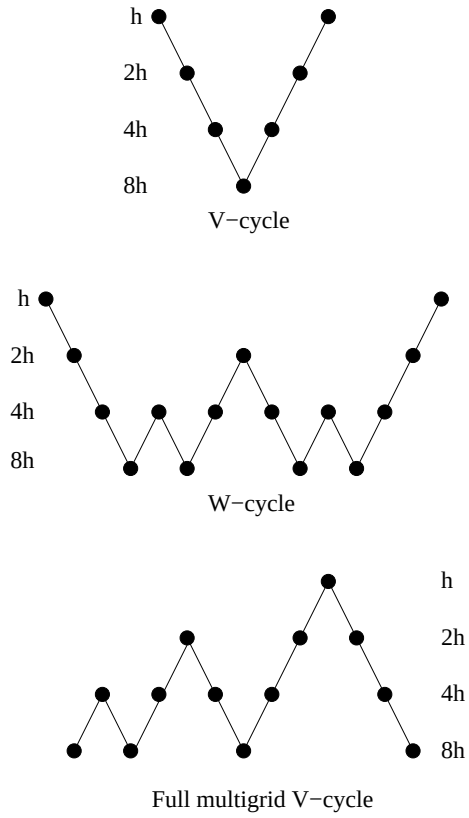


Figure 1.5: Illustration of different multigrid method.

The multigrid methods, when they work, can yield a solution with computational cost proportional to the size of the problems. In other words, both CPU time and storage space are of order $O(n)$, where n is the size of the problem [22]. However, implementation of classical multigrid methods necessitates a multi-level grids and special multigrid methods are needed for particular applications. Although algebraic multigrid methods can relax some of these requirements to certain extent, they still need underlying the partial differential equations which are sometimes unavailable in

practical applications.

1.4 Organization

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

- In Chapter 2, I use multiscale multigrid computing techniques to compute sixth order fine grid solutions for both the 1D and 2D Poisson equations. This method is based on the fourth order discretization scheme combined with operator based interpolation scheme and Richardson extrapolation technique. Multiscale multigrid method is proposed to solve the arising linear systems. Numerical results are provided to show the efficiency and robustness of our algorithms.

- In Chapter 3, the sixth order discretization method is extended to solve the 2D convection-diffusion equations with variable coefficients. The extended scheme can handle the problem with very high Reynolds number. Residual scaling strategy is also applied to keep the grid independency of our multiscale multigrid method.

- In Chapter 4, I extend the sixth order scheme from the 2D convection-diffusion equation to 3D convection-diffusion equation. A new operator based interpolation scheme is proposed to approximate the sixth order solutions on fine grid. Plane relaxation method is developed to solve the linear systems with high Reynolds number.

- Chapter 5 presents two new techniques: multiple coarse grids computation and W-cycle multiscale multigrid method. The multiple coarse grids computation can approximate the fourth order solutions on every coarse grid. The extrapolation is applied for every fine grid point to compute the sixth order solution. The W-cycle multiscale multigrid method is an improved multiscale multigrid

method which requires less CPU cycles and achieves more accurate sixth order solutions.

- In Chapter 6, I use OpenMP to develop a parallel multiscale multigrid solver. In order to achieve parallelism, I use multicolor relaxation scheme in multigrid method. I tested the parallel solver for the 3D convection-diffusion equations.

- Chapter 7 summarizes conclusions of this dissertation and proposed future research work.

Chapter 2

Sixth Order Solution for 2D Poisson Equation

In this chapter, I discuss the sixth order solutions for Poisson equation. Poisson equation is a partial differential equation with broad utility in electrostatics, mechanical engineering and theoretical physics. It is named after the French mathematician, geometer and physicist Siméon-Denis Poisson.

The efforts to compute more accurate solution using limited grid sizes have directed researcher's attention to developing high order compact schemes for Poisson equations. In context of the fourth order compact finite difference discretizations, much research and applications have been focused on equal or unequal meshsize discretization [25, 56, 77].

I believe that using highly accurate discretization schemes with scalable linear system solver is a better strategy to achieve fast and high resolution numerical solution of PDEs, than relying on either one of these approaches alone. Unfortunately, numerical discretization and fast linear system solvers are traditionally studied by two separate groups of people. In this dissertation, I integrate the advantages of both approaches and build an efficient framework to solve Poisson equations by incorporating high accuracy discretization into multigrid solution process.

For the sixth order solutions, I combine the fourth order compact discretization, multigrid method, Richardson extrapolation technique, and an operator based interpolation scheme. I use multigrid V-Cycle procedure to build our multiscale multigrid algorithm, which is similar to the full multigrid method (FMG). The multigrid com-

putation yields fourth order accurate solution on both the fine grid and the coarse grid. A sixth order accurate coarse grid solution is computed by using Richardson extrapolation technique. Then I apply our operator based interpolation scheme to compute sixth order accurate solution on the fine grid. Numerical experiments are conducted to demonstrate the accuracy of the solution obtained and the computational efficiency of our new method compared to Sun-Zhang's sixth order Richardson extrapolation compact (REC) discretization strategy [59] using Alternating Direction Implicit (ADI) method and the standard fourth order compact difference (FOC) scheme using a multigrid method.

This chapter is organized as follows: In Section 2.1, I present a sixth order compact difference discretization strategy for the 2D Poisson equation. In Section 2.2, I develop our modified multigrid method to compute the fourth order accurate solution on the fine and the coarse grids. Section 2.3 contains the numerical experiments to demonstrate the high accuracy of the sixth order compact difference scheme, as well as the computational efficiency of our modified multigrid method. Concluding remarks are given in Section 2.4.

2.1 Compact Finite Difference Approximations

I want to develop an explicit sixth order compact finite difference scheme to discretize the 2D Poisson equation. Since a sixth order explicit compact scheme may be impossible to develop on a single scale grid, the multiscale grid method has been considered to achieve the sixth order accuracy for the explicit compact formulations. Sun and Zhang [59] first proposed a sixth order explicit finite difference discretization strategy for solving the 2D convection-diffusion equation. They used ADI method to compute the fourth order accurate solution on the fine and the coarse grids first, then apply Richardson extrapolation technique and an operator based interpolation scheme in each ADI iteration to achieve the sixth order accurate solution on the fine grid. The

major disadvantage of Sun-Zhangs method is that the ADI iteration is not scalable with respect to the meshsize. When the mesh becomes finer, the number of ADI iterations needed for convergence increases quickly.

By using the idea of two scale grid computation from Sun-Zhang's method, I intend to develop a new explicit sixth order compact computing strategy for the 2D Poisson equation, which can efficiently solve the resulting linear system and is scalable with respect to the problem size. Our explicit sixth order compact finite difference scheme is based on a fourth order compact discretization on the two scale grids. In this section, I first introduce the fourth order compact scheme for the 2D Poisson equation. The basic idea is from Zhang's previous work [59, 77]. More detailed discussions about the fourth order finite difference schemes can be found in [24, 55].

2.1.1 1D Poisson equation

I first give a brief introduction to the fourth order compact (FOC) difference scheme for solving a 1D Poisson equation of the form

$$\frac{\partial^2 u}{\partial x^2} = f(x), \qquad 0 \le x \le l. \tag{2.1}$$

Eq. (2.1) is considered to have suitable boundary conditions. I denote the domain by Ω and use uniform mesh spacing $h = l/n$, where n is the number of grid points. Here, $f(x)$ is the forcing function that is assumed to have the necessary derivatives up to certain orders. I denote $x_j = jh$, $u_j = u(x_j)$, and $f_j = f(x_j)$, where $j = 0, 1, ..., n$.

I now write the well-known second order central difference operators as [57]

$$\delta_{xx}^h u_j = \frac{u_{j+1} + u_{j-1} - 2u_j}{h^2}, \qquad j = 1, 2, ..., n-1.$$

By using the Taylor series expansions, we can rewrite this operator as

$$\delta_{xx}^h u_j = u_{xx} + \frac{h^2}{12} u_{x^4} + \frac{h^4}{360} u_{x^6} + O(h^6), \tag{2.2}$$

and second order solution can be achieved by dropping the last three terms. In Eq. (2.2), u_{x^4} is the $4th$ partial derivative evaluated at the grid point x_i. The idea behind the high order compact approximation scheme is to approximate the term u_{x^4}. The fourth derivatives for the function $u(x)$ can be obtained by taking derivatives on both sides of Eq. (2.1) like

$$u_{x^4} = f_{xx}. \tag{2.3}$$

By applying central difference operator on f_{xx} at grid point x_j, Eq. (2.3) can be rewritten as

$$(u_{x^4})_j = \delta_{xx}^h f_j + O(h^2). \tag{2.4}$$

Taking Eq. (2.4) into Eq. (2.2), we have

$$\begin{aligned} u_{xx} &= \delta_{xx}^h u_j - \frac{h^2}{12}(\delta_{xx}^h f_j + O(h^2)) + \tau h^4 + O(h^6) \\ &= \delta_{xx}^h u_j - \frac{h^2}{12}\delta_{xx}^h f_j + \tau h^4 + O(h^6), \end{aligned} \tag{2.5}$$

where τ is some complicated representation not related to h and will be dropped in the Richardson extrapolation procedure. Hence, the fourth order compact approximation of the 1D Poisson equation at grid point x_j is

$$\delta_{xx}^h u_j = (1 + \frac{h^2}{12}\delta_{xx}^h) f_j + \tau h^4 + O(h^6). \tag{2.6}$$

The linear system consisting of equations at all interior grid points in Eq. (2.6) is

tridiagonal and can be solved easily [13]. We can write the solution symbolically as

$$u_j^h = (\delta_{xx}^h)^{-1}((1 + \frac{h^2}{12}\delta_{xx}^h)f_j + \tau h^4) + O(h^6). \tag{2.7}$$

A FOC difference scheme is obtained by dropping the h^6 and h^4 terms in Eq. (2.7). The u_j^h indicates the FOC solution on the Ω_h grid. By changing the grid size to $2h$, we can compute the FOC solution u_j^{2h} on the coarse grid Ω_{2h}.

Fourth order to sixth order. Using standard Richardson extrapolation strategy [13], a sixth order accuracy solution on the coarse Ω_{2h} grid can be computed as

$$\tilde{u}_j^{2h} = \frac{16u_{2j}^h - u_j^{2h}}{15}. \tag{2.8}$$

For the fine grid solution, I first interpolate the sixth order coarse grid solution $\tilde{u}_j^{2h}$ to the even indexed grid points on the Ω_h grid. And the Eq. (2.6) can be written in a 3-point stencil as

$$u_{j+1} + u_{j-1} - 2u_j = F_j,$$

where $F_j = h^2 f_j(1 + \frac{h^2}{12}\delta_{xx}^h)$.

So, the odd indexed grid points x_{2j-1} with $j = 1, 2, ..., n/2$ on the grid Ω_h can be computed as

$$\tilde{u}_{2j-1} = \frac{1}{2}[\tilde{u}_{2j-2} + \tilde{u}_{2j} - F_{2j-1}] + O(h^6), \tag{2.9}$$

where $\tilde{u}_{2j}$ is a sixth order solution on Ω_{2h} computed by Eq. (2.8). It follows that a sixth order solution $\tilde{u}_j$ on Ω_h. The procedure is shown in Fig. 2.1.

We can make a summary of the Richardson extrapolation (REC) algorithm for computing a sixth order approximate solution of Eq. (2.1) as in Algorithm 2.

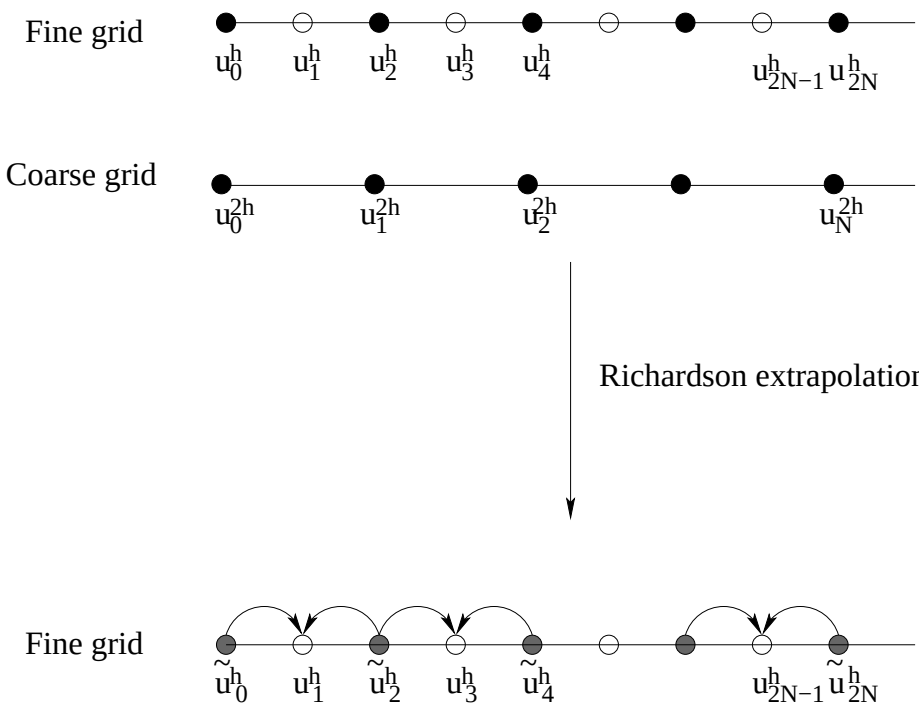


Figure 2.1: The procedure of how to get sixth order solution on the fine grid for 1D problem. The gray colored grid is the even grid point with sixth order accuracy after extrapolation.

2.1.2 2D Poisson equation

For solving PDEs in higher dimensions, there are two general approaches. Consider the 2D Poisson equation for example. The 2D PDEs can be viewed as 1D problems by lagging the partial derivatives with respect to the y variables [14]. The 1D sixth order compact scheme can be used with Alternating Direction Implicit (ADI) method to solve 2D PDEs. But as I commented in Chapter 1, the ADI method is not scalable

Algorithm 2 Sixth order approximation for 1D equation

1: Solve an n-by-n tridiagonal linear system on Ω_h with FOC scheme to get fine grid solution u_h;
2: Solve an $n/2$-by-$n/2$ tridiagonal linear system on Ω_{2h} with FOC scheme to get coarse grid solution u_{2h};
3: Using Richardson extrapolation technique to compute the sixth order solution $\tilde{u}_j^{2h}$ on the Ω_{2h};
4: Interpolate the $\tilde{u}_j^{2h}$ to the even grid points of Ω_h and use formula (2.9) to compute the odd grid points to get the sixth order solution $\tilde{u}_j^h$.

with respect to the meshsize, when the number of grid points increases, the ADI method is likely to take lots of iterations to converge and is therefore computationally inefficient for solving large scale problems. Our approach is to directly solve the linear system arising from explicit discretization schemes. Like the 1D case, the sixth order explicit compact scheme for the 2D Poisson equation is also based on the FOC scheme.

The 2D Poisson equation can be written in the form of

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \qquad (x, y) \in \Omega, \tag{2.10}$$

where Ω is a rectangular region, or a union of rectangular regions, 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 the necessary continuous partial derivatives up to certain orders.

For simplicity, I assume $\Omega = [0, L_x] \times [0, L_y]$. I discretize Ω with uniform meshsizes $\Delta x = L_x/N_x$ 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 as x and y axes. The mesh points are (x_i, y_j) with $x_i = i\Delta x$ and $y_j = j\Delta y$, $0 \le i \le N_x$, $0 \le j \le N_y$.

Discretization of the 2D PDE is similar to the 1D PDE, I first write the standard second order central difference operators as

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

By using Taylor series expansion, at the grid point (x_i, y_j), we have

$$\delta_x^2 u_{i,j} = u_{xx} + \frac{\Delta x^2}{12} u_x^4 + \frac{\Delta x^4}{360} u_x^6 + O(\Delta x^6), \tag{2.11}$$

and

$$\delta_y^2 u_{i,j} = u_{yy} + \frac{\Delta y^2}{12} u_y^4 + \frac{\Delta y^4}{360} u_y^6 + O(\Delta y^6). \tag{2.12}$$

From previous studies [59, 77, 80], we can apply the symbolic fourth order compact approximation operator to the second derivatives u_{xx} and u_{yy} in Eq. (2.10), respectively. Then discretization of the 2D Poisson equation can be formulated symbolically as [57]

$$(1 + \frac{\Delta x^2}{12}\delta_x^2)^{-1}\delta_x^2 u + (1 + \frac{\Delta y^2}{12}\delta_y^2)^{-1}\delta_y^2 u = f + O(\Delta^4), \tag{2.13}$$

where Δ^4 denotes the truncated terms in the order of $O(\Delta x^4 + \Delta y^4)$. And Eq. (2.13) can be rewritten as

$$\begin{aligned}
&(1 + \frac{\Delta y^2}{12}\delta_y^2)\delta_x^2 u + (1 + \frac{\Delta x^2}{12}\delta_x^2)\delta_y^2 u \\
&= (1 + \frac{\Delta x^2}{12}\delta_x^2)(1 + \frac{\Delta y^2}{12}\delta_y^2)f + \tau_1\Delta x^4 + \tau_2\Delta y^4 + O(\Delta^6) \\
&= [1 + \frac{1}{12}(\Delta x^2\delta_x^2 + \Delta y^2\delta_y^2)]f + \tau_1\Delta x^4 + \tau_2\Delta y^4 + \tau_0(\Delta x^2.\Delta y^2) + O(\Delta^6).
\end{aligned} \tag{2.14}$$

Here τ_1, τ_2 and τ_0 are some complicated representations that will be dropped in the Richardson extrapolation procedure, (we absorbed the $\Delta x^2.\Delta y^2$ term into the $O(\Delta^4)$ term). If I drop the Δ^6 term and set the τ_1, τ_2 and τ_0 equal to zero, I will get the general fourth order compact scheme for the 2D Poisson equation as

$$(\delta_x^2 + \delta_y^2)u + \frac{1}{12}(\Delta x^2 + \Delta y^2)\delta_x^2\delta_y^2 u = f + \frac{1}{12}(\Delta x^2\delta_x^2 + \Delta y^2\delta_y^2)f. \tag{2.15}$$

If I set the mesh aspect ratio $\gamma = \Delta x/\Delta y$, we can rewrite the Eq. (2.15) in the following form [77]

$$\begin{aligned}
&au_{i,j} + b(u_{i+1,j} + u_{i-1,j}) + c(u_{i,j+1} + u_{i,j-1}) + d(u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1}) \\
&= \frac{\Delta x^2}{2}(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}),
\end{aligned} \tag{2.16}$$

which has a nine point computational stencil as shown in Fig. 2.2. Here the coeffi-

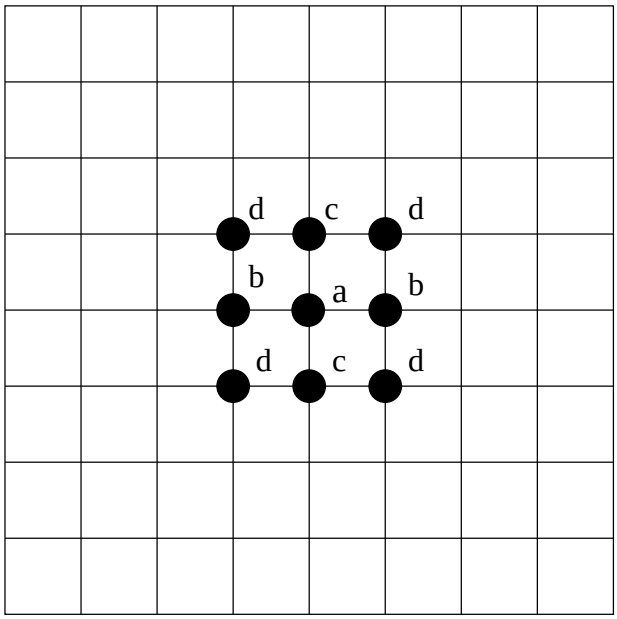


Figure 2.2: Fourth order 9-point computational stencil for 2D Poisson equation.

cients are

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

When $\Delta_x = \Delta_y = \Delta$, the 9-point stencil formula (2.16) can be

$$\begin{aligned} &u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} + 4(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - 20u_{i,j} \\ &= \frac{\Delta^2}{2}(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}). \end{aligned} \tag{2.17}$$

I want to mention here that all the interior grid points have the same 9 point computational stencil, no special boundary conditions are needed.

Anisotropic 2D Poisson equation. The anisotropic 2D Poisson equation is

$$u_{xx} + \beta u_{yy} = f(x,y), \qquad (x,y) \in \Omega, \tag{2.18}$$

where $\beta \neq 1$. But one can use same discretization scheme as for the isotropic Poisson

equation to discretize it. In fact, the anisotropic Poisson equation can be transformed into an isotropic equation. Suppose that, $y \in [0, L_y]$, I just map the $\bar{y} = y/\sqrt{\beta}$ and change the domain $[0, L_y]$ to $[0, y/\sqrt{\beta}]$, then the Eq. (2.18) can be rewritten as

$$u_{xx} + u_{\bar{y}\bar{y}} = f(x, \bar{y}).$$

Thus, the anisotropic Poisson equation can be considered as an isotropic Poisson equation with a scaled domain.

2D operator based interpolation. By solving the linear system arising from the FOC scheme for the 2D Poisson equation, I have the fourth order solution $u^h_{i,j}$ on the grid Ω_h and $u^{2h}_{i,j}$ on the grid Ω_{2h}. Like the 1D problem, we first use the Richardson extrapolation to achieve the sixth order solution on the coarse grid like [11, 45]

$$\tilde{u}^{2h}_{i,j} = \frac{(16u^h_{2i,2j} - u^{2h}_{i,j})}{15}. \tag{2.19}$$

The 2D problem is different from the 1D problem, we first divide the 2D grid points into four groups $(even, even)$, (odd, odd), $(even, odd)$, and $(odd, even)$. We can get the sixth order coarse grid solution u_{2h} on th coarse grid Ω_{2h}, which can be directly copied to the $(even, even)$ grid points on the fine grid. But we cannot just use the one step interpolation as we did for the 1D case to achieve the sixth order solution for other three groups of grid points. Instead, we developed a mesh-refinement [29] iterative method as shown in Fig. 2.3.

I assume that N_x and N_y are both even numbers, our operator based interpolation scheme is an iterative procedure. In each iteration, it will run the Richardson extrapolation first to get the sixth order solution on the coarse grid, then it will use a different interpolation strategy to interpolate the sixth order solution for different grid points on the fine grid. One interpolation iteration (from step k to step $k+1$) is

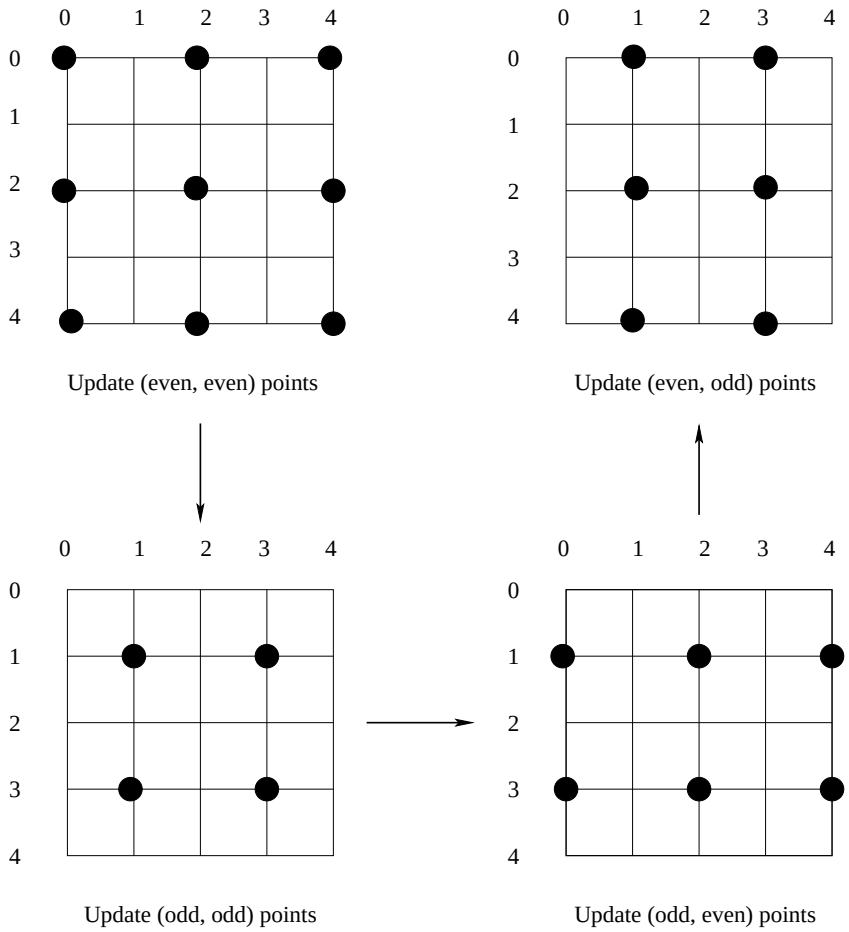


Figure 2.3: Illustration of the operator based interpolation scheme for a 5×5 fine grid.

outlined in Algorithm 3.

In Algorithm 3, Ω_h^4 and Ω_{2h}^4 denote the fourth order accurate solution space, Ω_h^6 and Ω_{2h}^6 mean the sixth order accurate solution space. $\tilde{u}^{h,k}$ is the approximate solution for the fine grid after k iterations. The operator based interpolation iteration will continue until the 2-norm R of the correction vector is reduced to below a certain tolerance.

I want to mention here that in Sun-Zhang's sixth order method [59] for the 2D convection-diffusion equation, they also applied an operator based interpolation scheme together with the Richardson extrapolation in each ADI iterations. Since the number of ADI iterations will increase quickly when the mesh size becomes finer, their

Algorithm 3 Operator based interpolation iteration combined with sixth order Richardson extrapolation technique.

1: Let $u^h_{old} = \tilde{u}^{h,k}$.
2: Update every $(even, even)$ grid point on Ω_h.
From $\tilde{u}^{2h,k}_{i,j} \in \Omega^4_{2h}$ and $\tilde{u}^{h,k}_{2i,2j} \in \Omega^4_h$, I first compute $\tilde{u}^{2h,k+1}_{i,j} \in \Omega^6_{2h}$ by Eq. (2.19), then use direct interpolation to get $\tilde{u}^{h,k+1}_{2i,2j} \in \Omega^6_h$.
3: Update every (odd, odd) grid point on Ω_h.
From Eq. (2.16), for each (odd, odd) point (i, j), the updated solution is

$$\tilde{u}^{h,k+1}_{i,j} = \frac{1}{a}[F_{i,j} - b(\tilde{u}^{h,k}_{i+1,j} + \tilde{u}^{h,k}_{i-1,j}) - c(\tilde{u}^{h,k}_{i,j+1} + \tilde{u}^{h,k}_{i,j-1}) - d(\tilde{u}^{h,k+1}_{i+1,j+1} + \tilde{u}^{h,k+1}_{i+1,j-1} + \tilde{u}^{h,k+1}_{i-1,j+1} + \tilde{u}^{h,k+1}_{i-1,j-1})]$$

Here, $F_{i,j}$ represents the right-hand side part of Eq. (2.16).
4: Update every $(odd, even)$ grid point on Ω_h.
From Eq. (2.16), the idea is similar to the (odd, odd) grid point.
5: Update every $(even, odd)$ grid point on Ω_h.
From Eq. (2.16), the idea is similar to the $(odd, even)$ grid point.
6: Compute the 2-norm $R = ||\tilde{u}^{h,k+1} - u^h_{old}||_2$. If not converged, go back to Step 1.

extrapolation and interpolation parts will take a large amount of CPU time. In order to avoid this unscalable computation, my new operator based interpolation procedure and the Richardson extrapolation are carried out only after I get the converged fine and coarse grid fourth order accurate solutions.

Convergence analysis. From Steps 3-5 in Algorithm 3, we can see that the interpolation scheme employs a Gauss-Seidel type of iterative method to approximate the sixth order solutions for every fine grid point. The coefficient matrix A that I use is generated from the fourth order compact difference scheme, and A is usually large and sparse. The convergence rate of this Gauss-Seidel type of iterative methods depends on properties of matrix A.

Lemma 2.1.1. *The point Jacobi and the point Gauss-Seidel type of iterative methods associated with A for $\sqrt{5}/5 < \gamma < \sqrt{5}$ are convergent for any initial guess, where γ is the mesh aspect ratio defined in Eq. (2.15).*

Proof. From Zhang [72] and Varga [63], we know that we need to prove that matrix A is irreducible and weakly diagonally dominant.

Since A is discretized from FOC scheme, it is easy to find that the directed graph of A is strongly connected. So, A is irreducible.

To prove A is weakly diagonally dominant, we need to prove that

$$|A_{i,i}| \geq \sum_{j=1, j\neq i}^{n} |A_{i,j}|, \qquad for\ i = 1, ..., n. \tag{2.20}$$

By applying Eq. (2.15) to (2.20), it is sufficient to prove

$$|\alpha_0| \geq \sum_{j=1}^{8} |\alpha_j|, \tag{2.21}$$

where α_j are the coefficients from 9-point computational stencil. We have

$$\begin{aligned} & |\alpha_0| - \sum_{j=1}^{8} |\alpha_j| \\ =& 10(1+\gamma^2) - 2|5-\gamma^2| - 2|5\gamma^2 - 1| - 2(1+\gamma^2) \\ =& 8(1+\gamma^2) - 2(|5-\gamma^2| + |5\gamma^2 - 1|). \end{aligned} \tag{2.22}$$

By solving (2.22), we have

$$|\alpha_0| - \sum_{j=1}^{8} |\alpha_j| = \begin{cases} 20\gamma^2 - 4 < 0, & \text{if } 0 < \gamma < \sqrt{5}/5, \\ 0, & \text{if } \sqrt{5}/5 \leq \gamma \leq \sqrt{5}, \\ 20 - 4\gamma^2 < 0, & \text{if } \gamma > \sqrt{5}, \end{cases}$$

So, A keeps the diagonal dominance only when $\sqrt{5}/5 \leq \gamma \leq \sqrt{5}$ holds.

■

2.2 Special Multigrid Method

As indicated in Chapter 1, the convergence rate of the multigrid method is independent of the grid size. Considerable computational time is saved by doing major computational work on the coarse grids. In the past two decades, multigrid method and many of its variants have been used in almost every scientific computing applications, solving both linear and nonlinear systems of PDEs [5, 8, 41, 44]. Various multigrid implementation strategies with the fourth order compact schemes to solve the 2D or 3D Poisson equations or other PDEs like convection-diffusion equations are discussed in [25, 27, 52].

I point out that multigrid method has traditionally been used as convergence acceleration method to solve the sparse linear systems arising from the discretized PDEs. In this section, I introduce a geometric multiscale multigrid method [66] to elevate the order of accuracy of the computed solution using this already existing multilevel grid hierarchy. The major advantage of multiscale multigrid method is that it has an optimal computational cost similar to that of a full multigrid method and can bring us the converged fourth order solutions on two grids with different scales.

2.2.1 Multiscale multigrid method

I use the notations u_{lh}, f_{lh} and L_{lh} to represent the approximate solution, the right-hand side vector and the finite difference operator for the grid Ω_{lh}, respectively. $I^{lh}_{(l-1)h}$ is the restriction operator from the grid $\Omega_{(l-1)h}$ to the grid Ω_{lh}, and $I^{(l-1)h}_{lh}$ is the interpolation operator from the grid Ω_{lh} to the grid $\Omega_{(l-1)h}$. The procedure of our multiscale multigrid method is shown in Fig. 2.4. The gray colored circle indicates the unconverged solution u_{4h} and the black colored circles are the fourth order converged solutions u_{2h} and u_h.

Below I describe a multigrid V-cycle based algorithm to solve the 2D Poisson

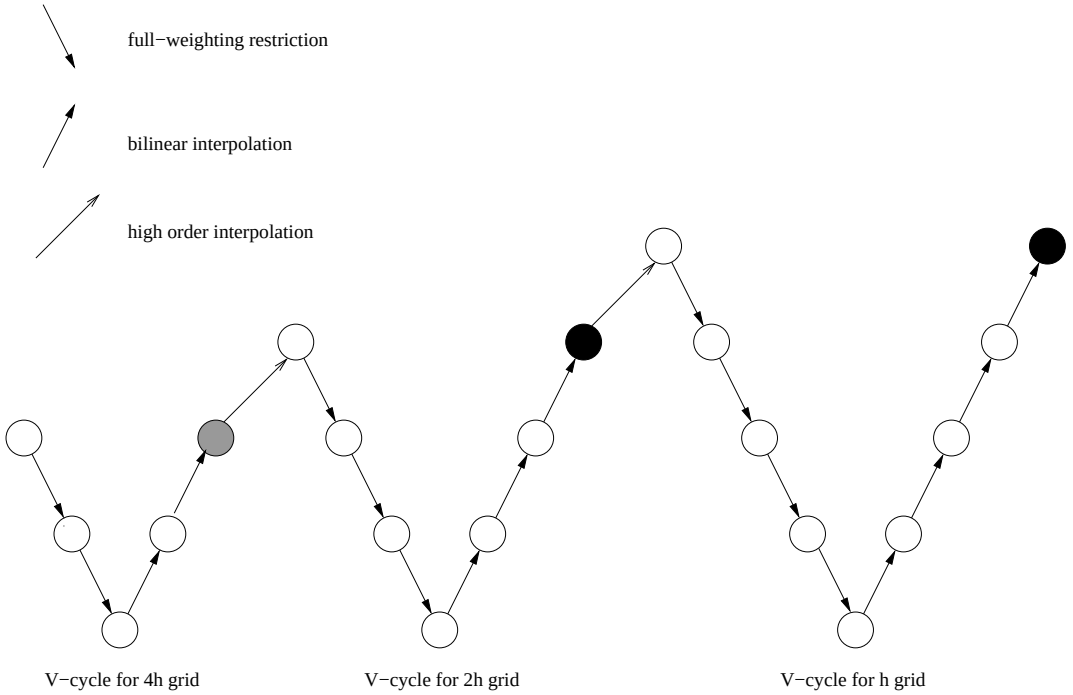


Figure 2.4: Representation of our multiscale multigrid method.

equation.

The Algorithm 4 is similar to the full multigrid method, but I do not start from the coarsest grid. Since I use the interpolated coarse grid solution as the initial guess for the fine grid V-Cycle, this algorithm will need fewer number of multigrid cycles than I run the V-Cycle on Ω_h and Ω_{2h} separately to get the converged fourth order accurate solutions u_h and u_{2h} [12, 52].

In the multiscale multigrid method, I use standard bilinear interpolation to transfer corrections from the coarse grid to the fine grid, full weighting scheme to project residual from the fine grid to the coarse grid, and bicubic interpolation to interpolate the initial guess in Step 2 and Step 5.

2.2.2 Relaxation method

For relaxation methods (smoothers), the standard multigrid method with point Gauss-Seidel type relaxation is widely used and is simple to implement. However, for

Algorithm 4 Multiscale Multigrid Method

1: Run the multigrid V-Cycle algorithm $MG(u_{4h}, f_{4h})$ on the coarser grid Ω_{4h} as in Fig. 2 for one or two cycles to get an approximate solution u_{4h}.
2: Use some high order interpolation schemes, like the bicubic interpolation or operator based interpolation, to interpolate u_{4h} to the coarse grid Ω_{2h}, $u_{2h} = I_{4h}^{2h} u_{4h}$.
3: Relax ν_1 times on $L_{2h} u_{2h} = f_{2h}$.
4: Use u_{2h} from the previous step as the initial guess to run the multigrid V-Cycle algorithm $MG(u_{2h}, f_{2h})$ on the coarse grid Ω_{2h} until it converges. I can get the converged fourth order accurate solution u_{2h}.
5: Use a high order interpolation to interpolate u_{2h} to the fine grid Ω_h like $u_h = I_{2h}^h u_{2h}$.
6: Relax ν_1 times on $L_h u_h = f_h$.
7: Use u_h from the previous step as the initial guess to run the multigrid V-Cycle algorithm $MG(u_h, f_h)$ on the fine grid Ω_h until it converges. We can get the converged fourth order accurate solution u_h.

solving the anisotropic Poisson equation, or equivalently, Poisson equation with unequal meshsizes, standard point Gauss-Seidel relaxation and standard mesh coarsening strategy (the coarse grid meshsizes double that of the fine grid) does not work very well [65, 44]. Two strategies can be used to treat these anisotropic equations: semicoarsening [38, 39] and line relaxation. For semicoarsening strategy, the mesh coarsening is only performed along the dominant direction. For the line relaxation, it is very efficient to dump the high frequency errors in the dominant direction with large coefficients. The line relaxation scheme has three implementations, X-Line scheme, Y- Line scheme and alternating (X-Y) Line scheme. If the x is the dominant direction, I only perform line relaxation along the x direction on each successive grids (X-Line scheme); If y is the dominant direction, I choose the Y-Line scheme; if it is difficult to determine the dominant direction, I prefer the X-Y Line scheme. For the particular problems considered in this dissertation, I prefer using line relaxation strategy.

Line relaxation. Assuming x is the dominant direction, line relaxation will be performed along x direction on each successive grids as in Fig. 2.5. On the coarsest level, $N_x = N_y = 2$ and there is a 1D linear system with one unknown along x

direction. The relaxation will become a direct solver on the coarsest grid.

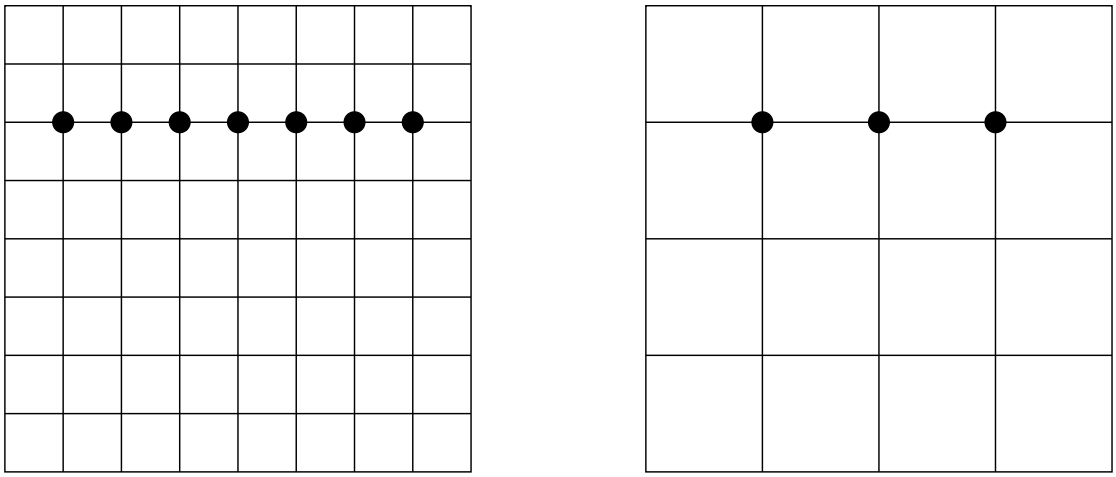


Figure 2.5: Line relaxation on a fine and coarse grids.

The coefficient matrix of the FOC scheme can be written as block tridiagonal matrix of block order N_y [72],

$$A = diag[A_1, A_0, A_1],$$

where

$$A_0 = diag[5-\gamma^2, -10(1+\gamma^2), 5-\gamma^2], \qquad A_1 = diag[(1+\gamma^2)/2, 5\gamma^2-1, (1+\gamma^2)/2]$$

are symmetric tridiagonal submatrices representing each line along x direction. The line relaxation is carried out for each line j $(1 \leq j \leq N_y - 1)$ as

$$A_0 u_j = f_j - A_1(u_{j-1} + u_{j+1}), \tag{2.23}$$

where u_j, f_j are part of the solution vector and right hand side representing the grid points on the jth line, respectively.

2.3 Numerical Experiments

In this section, I compare our new sixth order multigrid method with Richardson extrapolation (MG-Six) strategy with Sun-Zhang's sixth order REC method (REC-ADI) [59] and with the standard fourth order compact difference scheme using multigrid (MG-FOC). I used Fortran 77 programming language to implement and run on one processor of an IBM HS21 blade cluster at the University of Kentucky. The processor has 2GB of local memory and runs at 2.0GHZ.

The initial guess for the V-Cycle on Ω_{4h} is the zero vector. For Problem 1, the multigrid V-Cycle for the Ω_{2h} and Ω_h grids will stop when the 2-norm of the residual vector is reduced by 10^{-13}, the iterative interpolation procedure will stop when the 2-norm of the correction vector of the approximate solution is less than 10^{-13}. For the Problem 2, both of the stopping criteria will be changed to 10^{-10}. The errors reported are the maximum absolute errors over the discrete grid of the finest level.

Generally, 10^{-10} is our standard stopping criteria to check the 2-norm of the residual or the correction vector. For Problem 1, if we look at the experimental results from Table 2.1, we will find that when $n = 256$ the maximum error of our FOC scheme has dropped to 10^{-10}. If we still use 10^{-10} as our stopping tolerance, we may not get enough accuracy when the iteration stops. In order to get sufficient accuracy, I choose 10^{-13} as the stopping criteria for Problem 1.

For the line Gauss-Seidel relaxation schemes for these two test cases, I choose X-Y Line relaxation scheme for the Problem 1. For the Problem 2, since x is its dominant direction, I only perform line relaxation along the x direction, which is the X-Line relaxation scheme.

I also compute the estimated order of accuracy for every computing strategy in different grid size. Let us consider two meshsizes 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 to be 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

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

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

2.3.1 Test problem 1

In order to compare with Sun-Zhang's sixth order method, we consider one of the test cases in Sun-Zhang's paper [59]. Sun and Zhang used a 2D convection-diffusion equation, I set the convection coefficients to be zero, then the equation becomes a 2D Poisson equation. The test Problem 1 can be written as

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\alpha \sin(\frac{\pi}{b} y), \qquad (x, y) \in \Omega = [0, \lambda] \times [0, b], \tag{2.24}$$

where the boundary conditions are

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

In this equation, the parameter α is chosen as

$$\alpha = \frac{F\pi}{Rb}.$$

The analytic solution of Eq. (2.24) is:

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

The other parameters are chosen as

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

In the following, I define $N_x = N_y = n$. The meshsizes Δ_x and Δ_y are equal to λ/n and b/n, respectively. Table 2.1, Fig. 2.6 and Fig. 2.7 compare the results for the Problem 1.

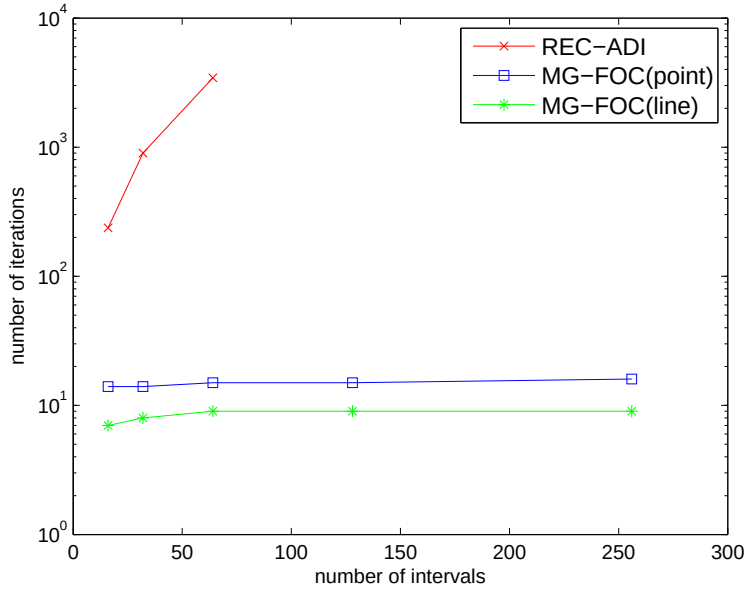


Figure 2.6: Comparison of the number of iterations with REC-ADI, MG-FOC(point) and MG-FOC(line) methods for the Problem 1. Each symbol with increasing number of iterations corresponds to an increasing fine grid: 16, 32, 64, 128, and 256 intervals.

Table 2.1 shows the number of iterations and other information for different solution strategies that I compared. We can find that when the mesh becomes finer, the number of ADI iterations increases very quickly. When $n > 64$, the ADI iteration

Table 2.1: Numerical comparison results for the Problem 1.

n	strategy	# iteration	CPU	error	order
16	REC-ADI	237	0.020	1.32e-6	5.7
	MG-Six(point)	(11,11), **40**	0.001	1.32e-6	5.7
	MG-FOC(point)	14	0.002	1.63e-5	3.9
	MG-Six(line)	(5,6), **40**	0.001	1.32e-6	5.7
	MG-FOC(line)	7	0.001	1.63e-5	3.9
32	REC-ADI	901	0.302	2.27e-8	5.9
	MG-Six(point)	(11,12), **39**	0.007	2.27e-8	5.9
	MG-FOC(point)	14	0.004	1.02e-6	4.0
	MG-Six(line)	(6,7), **39**	0.008	2.27e-8	5.9
	MG-FOC(line)	8	0.006	1.02e-6	4.0
64	REC-ADI	3447	4.662	3.68e-10	5.9
	MG-Six(point)	(12,13), **37**	0.029	3.67e-10	6.0
	MG-FOC(point)	15	0.022	6.39e-8	4.0
	MG-Six(line)	(7,7), **36**	0.033	3.67e-10	6.0
	MG-FOC(line)	9	0.024	6.40e-8	4.0
128	REC-ADI	not converged	–	–	–
	MG-Six(point)	(13,13), **33**	0.129	5.26e-12	6.1
	MG-FOC(point)	15	0.093	3.99e-9	4.0
	MG-Six(line)	(7,8), **34**	0.161	5.87e-12	6.0
	MG-FOC(line)	9	0.140	4.00e-9	4.0
256	REC-ADI	not converged	–	–	–
	MG-Six(point)	(14,14), **30**	0.880	1.27e-13	5.4
	MG-FOC(point)	16	0.465	2.47e-10	4.0
	MG-Six(line)	(8,8), **30**	1.202	1.11e-13	5.7
	MG-FOC(line)	8	0.737	2.50e-10	4.0

cannot converge within the maximum number of iterations we set, which is 5000. For the MG-Six method, the number of iterations contains three parts. They are the number of V-Cycles for Ω_{2h}, the number of V-Cycles for Ω_h, and the number of iterations for the iterative interpolation combined with the Richardson extrapolation. These three numbers are listed in the iteration columns for the MG-Six(point) and MG-Six(line) strategies in Table 2.1. We can see that, by using our new sixth order compact scheme, the number of V-Cycles for Ω_h and Ω_{2h} are reduced, compared to the traditional multigrid V-Cycle with the FOC scheme. We can also see that

the REC-ADI method takes much more iterations and CPU time than the MG-FOC strategies and the MG-Six strategies from Fig. 2.6 and Fig. 2.7,.

The data in Table 2.1 and Fig. 2.7 also indicate that the accuracy of the approximate solutions computed by our new sixth order method and the Sun-Zhang's REC-ADI method is comparable. When $n > 64$, the REC-ADI method cannot converge but our multigrid method can still compute the highly accurate solution.

Since the grid is almost isotropic for the Problem 1, we can see that the point Gauss-Seidel relaxation scheme remains competitive compared with the line Gauss-Seidel relaxation. The point Gauss-Seidel relaxation scheme actually needs less CPU time than the line Gauss-Seidel relaxation scheme to compute numerical solution of comparable accuracy.

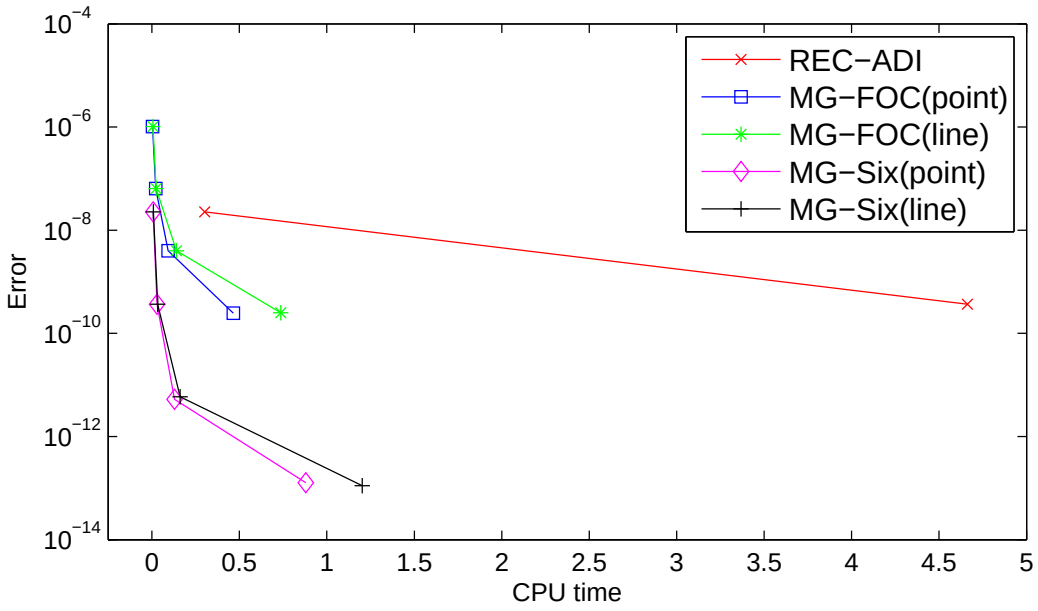


Figure 2.7: Comparison of the maximum error and the CPU cost for the Problem 1. Each symbol with increasing CPU cost corresponds to an increasing fine grid: 16, 32, 64, 128, and 256 intervals.

2.3.2 Test problem 2

In order to better compare the line Gauss-Seidel relaxation scheme and the point Gauss-Seidel relaxation scheme, I consider an anisotropic Poisson equation to show the efficiency and scalability of the line relaxation scheme in solving 2D Poisson equation.

I choose the following equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -2\pi^2 \sin(\pi x)\cos(\pi y), \qquad (x, y) \in \Omega = [0, 4] \times [0, 1], \tag{2.25}$$

which has the Dirichlet boundary condition.

The analytic solution of Eq. (2.25) is:

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

We choose X-Line Gauss-Seidel relaxation scheme.

As for the Problem 1, I also compare different solution strategies indexed by the number of multigrid cycles or iterations, CPU time, the maximum absolute errors and the estimated order of accuracy. The results are shown in Table 2.2, Fig. 2.8, and Fig. 2.9. We can clearly see that, even with the anisotropy, the convergence rates of our MG-Six(line) and MG-FOC(line) are barely affected. These two schemes can maintain both scalability and efficiency when the number of intervals increases. For MG-Six(point) and MG-FOC(point), they need much more iterations and CPU time than the line relaxation schemes. When $n < 64$, even the REC-ADI method can converge with less CPU time than the MG-Six(point) method. So, multigrid method with the line relaxation scheme is the most efficient way to solve the anisotropic 2D Poisson equation compared with the other methods I tested.

Again in this test case, our new sixth order accurate method can solve the problem

Table 2.2: Numerical comparison results for the Problem 2.

n	strategy	# iteration	CPU	error	order
16	REC-ADI	14	0.002	1.12e-4	5.4
	MG-Six(point)	(14,43), **55**	0.005	1.12e-4	5.4
	MG-FOC(point)	39	0.003	2.45e-4	4.2
	MG-Six(line)	(1,7), **55**	0.001	1.12e-4	5.4
	MG-FOC(line)	6	0.001	2.45e-4	4.2
32	REC-ADI	42	0.014	2.50e-6	5.5
	MG-Six(point)	(43,60), **85**	0.025	2.50e-6	5.5
	MG-FOC(point)	58	0.016	1.43e-5	4.1
	MG-Six(line)	(7,9), **85**	0.011	2.50e-6	5.5
	MG-FOC(line)	10	0.005	1.43e-5	4.1
64	REC-ADI	155	0.271	4.58e-8	5.8
	MG-Six(point)	(60,73), **93**	0.122	4.58e-8	5.8
	MG-FOC(point)	72	0.095	8.70e-7	4.0
	MG-Six(line)	(9,9), **93**	0.044	4.58e-8	5.8
	MG-FOC(line)	11	0.021	8.70e-7	4.0
128	REC-ADI	607	6.849	7.66e-10	5.9
	MG-Six(point)	(73,79), **89**	0.571	7.66e-10	5.9
	MG-FOC(point)	78	0.423	5.37e-8	4.0
	MG-Six(line)	(9,9), **89**	0.188	7.66e-10	5.9
	MG-FOC(line)	12	0.091	5.37e-8	4.0
256	REC-ADI	2411	130.393	1.22e-11	6.0
	MG-Six(point)	(79,83), **80**	3.774	1.24e-11	6.0
	MG-FOC(point)	82	3.201	3.33e-9	4.0
	MG-Six(line)	(9,9), **80**	1.982	1.24e-11	6.0
	MG-FOC(line)	13	1.649	3.33e-9	4.0

with high order accuracy which is comparable with Sun and Zhang's method and use less CPU time. It is clear that the MG-Six method outperforms other methods.

2.4 Concluding Remarks

I designed a new sixth order compact computation scheme with a multigrid method and Richardson extrapolation to solve the 2D Poisson equation. This new idea is based on designing a geometric multiscale multigrid method, similar to the full multigrid method, to compute the approximate solution using the fourth order compact

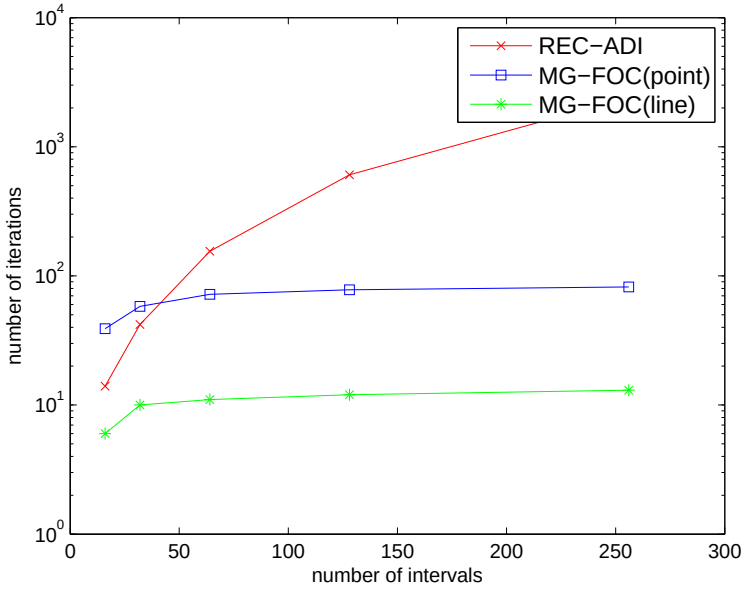


Figure 2.8: Comparison of the number of iterations with REC-ADI, MG-FOC(point) and MG-FOC(line) methods for the Problem 2. Each symbol with increasing number of iterations corresponds to an increasing fine grid: 16, 32, 64, 128, and 256 intervals.

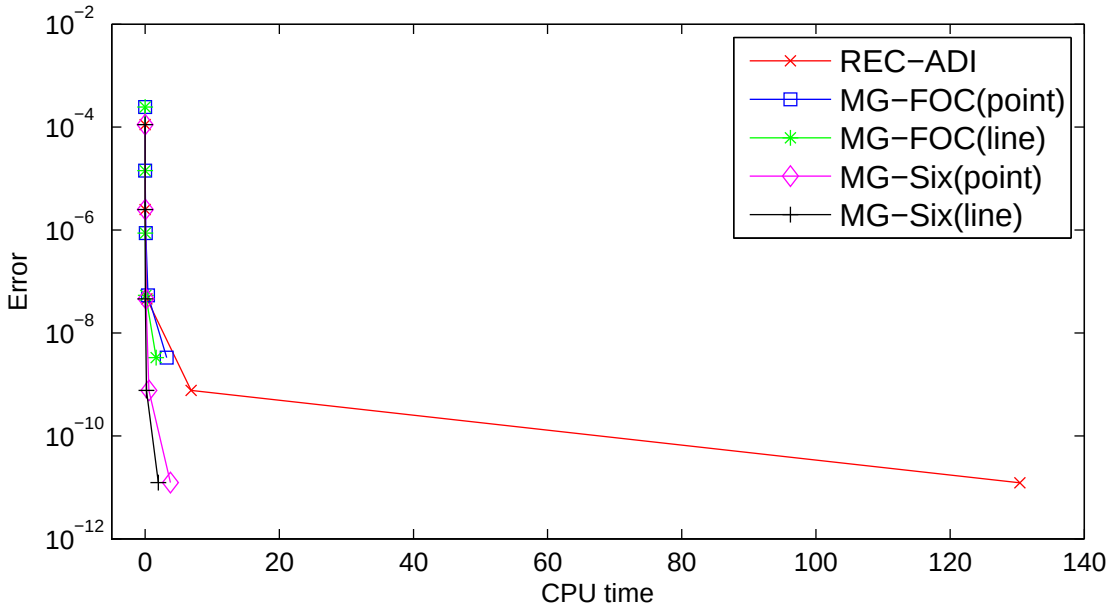


Figure 2.9: Comparison of the maximum error and the CPU cost for the Problem 2. Each symbol with increasing CPU cost corresponds to an increasing fine grid: 16, 32, 64, 128, and 256 intervals.

scheme in both the fine and the coarse grids. I also presented a new iterative interpolation scheme, which combined with the Richardson extrapolation scheme to approximate sixth order accuracy on the fine grid.

Numerical results show that the new numerical solution method can solve the 2D Poisson equation with high accuracy compared with other sixth order compact schemes, and also require low CPU time. This two scale grid idea can also be extended to solve other PDEs such as the 3D Poisson equation, 2D and 3D convection-diffusion equations. For the convection-dominated problems, multigrid methods with the line relaxation schemes will be expected to work well. For various multigrid algorithms with the high order compact schemes to solve convection-diffusion equations, we refer readers to [71].

Chapter 3

Sixth Order Solution for 2D Convection Diffusion Equation

In this chapter, I extend our explicit sixth order compact finite difference scheme to get fast high accuracy numerical solutions of the two dimensional convection-diffusion equation with variable coefficients. The convection-diffusion equation is a partial differential equation, which describes physical phenomena where particles or energy (or other physical quantities) are transferred inside a physical system due to two processes: diffusion and convection.

3.1 Introduction

I consider the two dimensional (2D) steady convection-diffusion equation with the Dirichlet boundary condition, which can be written as

$$\begin{aligned} u_{xx} + u_{yy} + p(x,y)u_x + q(x,y)u_y &= f(x,y), \qquad (x,y) \in \Omega, \\ u(x,y) &= g(x,y), \qquad (x,y) \in \partial\Omega, \end{aligned} \tag{3.1}$$

where Ω is a 2D rectangular domain and $\partial\Omega$ is the boundary of Ω. I assume that the convection coefficients $p(x,y)$ and $q(x,y)$ are sufficiently smooth on Ω. The convection-diffusion equations appear in a variety of applications involving the modeling of transport phenomena [47]. Efficient numerical solution of Eq. (3.1) plays an increasingly important role in computational fluid dynamics [23]. The magnitudes of $p(x,y)$ and $q(x,y)$ determine the ratio of the convection to diffusion. If they become

zero simultaneously at some point in Ω, this point will be called a stagnation point. In fluid dynamics, a stagnation point is a point in a flow field where the local velocity of the fluid is zero. Stagnation points exist at the surface of objects in the flow field, where the fluid is brought to rest by the object. The convection-diffusion equation with stagnation points are usually used to model recirculating flow problems.

Similar to the Poisson equation, Eq. (3.1) can be discretized using some finite difference schemes to result in a system of linear equations

$$A^h u^h = f^h, \tag{3.2}$$

where h is the uniform grid spacing of the discretized domain Ω^h. In many real CSEI applications, the coefficient matrix A is usually large, sparse, nonsymmetric, and indefinite for large cell Reynolds number (Re) [30]. The Reynolds number is used to determine the ratio of convection to diffusion as

$$Re = \max(\sup_{(x,y)\in\Omega} |p(x,y)|, \sup_{(x,y)\in\Omega} |q(x,y)|)h/2. \tag{3.3}$$

We say that the discrete problem (3.2) is convection-dominated if $Re > 1$ and diffusion-dominated if $Re \leq 1$. Normally, the numerical solutions of Eq. (3.2) based on iterative methods become increasingly difficult as Re increases [21, 73].

In general, for the convection-dominated problems, traditional finite difference discretization schemes such as the five-point second order central difference scheme (CDS) and the upwind difference scheme (UDS) cannot yield satisfactory results [76]. CDS has a truncation error of order $O(h^2)$ but may produce nonphysical oscillations for large Re. UDS usually prevents oscillations but reduces the order of accuracy to $O(h)$ [54].

I want to develop higher order accurate discretization schemes for the 2D convection-diffusion equation. In the past two decades, there has been growing interest in us-

ing higher order compact schemes to solve the partial differential equations. Gupta *et al.* proposed a nine-point fourth order compact (FOC) scheme to discretize the 2D convection-diffusion equation with variable coefficients [24]. There are also some other similar fourth order compact schemes that have been developed for the 2D convection-diffusion equations. Readers are referred to [33, 54, 55] and the references therein for more details.

For the sixth order schemes, I successfully extend the SOC scheme from the 2D Poisson equation to 2D convection-diffusion equation [67]. Discretization strategy is similar to the Poisson equation and I prove that the FOC discretization will be degraded to second order when Re reaches infinity. I used line relaxation and residual scaling techniques in our multiscale multigrid method to handle the problems with large Reynolds number which are very difficult to solve by using standard multigrid method.

3.2 Compact Finite Difference Approximation

Similar to the 2D Poisson equation, my explicit sixth order compact scheme for the 2D convection-diffusion equation is based on the fourth order compact (FOC) discretization on two scale grids. The FOC schemes for the 2D convection-diffusion equation have been proposed by several authors [24, 33, 54] in different ways. I believe that these schemes are mathematically equivalent, although they were derived using different approaches. In this paper, I use the FOC scheme by Gupta *et al.* [24].

3.2.1 Order of accuracy for 1D problem with large Re

For convection-dominated problems, the large convection coefficient will affect the order of accuracy of computed solutions. For better understanding, I consider the one dimensional (1D) convection-diffusion type equation.

I consider the following 1D model convection-diffusion equation

$$u_{xx} + bu_x = 0, \qquad 0 \le x \le l, \tag{3.4}$$

where b is the constant convection coefficient.

I denote h to be the meshsize, $x_j = jh$ and $u_j = u(x_j)$. The standard first and second order central difference operators are

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

By using Taylor series, we have

$$\delta_{xx}^h u_j = u_{xx} + \frac{h^2}{12}u_{x^4} + \frac{h^4}{360}u_{x^6} + \frac{h^6}{20160}u_{x^8} + O(h^8), \tag{3.5}$$

and

$$\delta_x^h u_j = u_x + \frac{h^2}{6}u_{x^3} + \frac{h^4}{120}u_{x^5} + \frac{h^6}{5040}u_{x^7} + O(h^8), \tag{3.6}$$

in which I denoted the mth derivative of the function $u(x)$ as

$$u_{x^m} = \frac{\partial^m u}{\partial x^m}.$$

From Eqs. (3.5) and (3.6) I can discretize Eq. (3.4) at the grid point x_j as

$$\begin{aligned}\delta_{xx}^h u_j + b\delta_x^h u_j =& \frac{bh^2}{6}u_{x^3} + \frac{h^2}{12}u_{x^4} + h^4(\frac{b}{120}u_{x^5} + \frac{1}{360}u_{x^6}) \\ &+ h^6(\frac{b}{5040}u_{x^7} + \frac{1}{20160}u_{x^8}) + O(h^8).\end{aligned} \tag{3.7}$$

By taking derivatives on both sides of Eq. (3.4), we have

$$u_{x^3} = -bu_{x^2}, \qquad u_{x^4} = b^2 u_{x^3}. \tag{3.8}$$

So, at the grid point x_j, I have

$$(u_{x^3})_j = -b\left[\delta^h_{xx}u_j - \frac{h^2}{12}(u_{x^4})_j - \frac{h^4}{360}(u_{x^6})_j - O(h^6)\right], \tag{3.9}$$

and

$$(u_{x^4})_j = b^2\left[\delta^h_{xx}u_j - \frac{h^2}{12}(u_{x^4})_j - \frac{h^4}{360}(u_{x^6})_j - O(h^6)\right]. \tag{3.10}$$

Then I use Eqs. (3.9) and (3.10) to replace the u_{x^3} and u_{x^4} terms in Eq. (3.7) as

$$\begin{aligned}\delta^h_{xx}u_j + b\delta^h_x u_j = &-\frac{h^2b^2}{12}\left[\delta^h_{xx}u_j - \frac{h^2}{12}(u_{x^4})_j - \frac{h^4}{360}(u_{x^6})_j\right] \\ &+ h^4(\frac{1}{360}u_{x^6} + \frac{b}{120}u_{x^5}) + h^6(\frac{1}{20160}u_{x^8} + \frac{b}{5040}u_{x^7}) + O(h^8).\end{aligned} \tag{3.11}$$

Eq. (3.11) can be rewritten in the form of

$$\left[(1 + \frac{h^2b^2}{12})\delta^h_{xx} + b\delta^h_x\right]u_j = (\tau_1)_j + (\tau_2)_j + O(h^8), \tag{3.12}$$

where $(\tau_1)_j$ is in the form of

$$(\tau_1)_j = \left[\frac{b^2}{144}(u_{x^4})_j + \frac{b}{120}(u_{x^5})_j + \frac{1}{360}(u_{x^6})_j\right]h^4,$$

and $(\tau_2)_j$ is in the form of

$$(\tau_2)_j = \left[\frac{b^2}{4320}(u_{x^6})_j + \frac{b}{5040}(u_{x^7})_j + \frac{1}{20160}(u_{x^8})_j\right]h^6.$$

If I drop $(\tau_2)_j$ and $O(h^8)$ terms in Eq. (3.12), I get the fourth order truncation error for the FOC scheme as $(\tau_1)_j$. After using our operator based interpolation scheme combined with Richardson extrapolation technique, the fourth order error term $(\tau_1)_j$ will be dropped. So, I will get a sixth order truncation error in proportion to $(\tau_2)_j$.

Order of accuracy with large convection coefficient. The order of accuracy of the computed solution is formally defined when the meshsize h approaches zero, but the actual order of solution accuracy of the FOC scheme and the sixth order (SOC) scheme is affected by the convection coefficient.

I first consider the fourth order truncation error from the FOC scheme. From Eq. (3.3), I can get the cell Reynolds number for the 1D convection equation as

$$Re = \frac{bh}{2}.$$

So, the convection coefficient can be represented as

$$b = \frac{2Re}{h}. \tag{3.13}$$

By using Eq. (3.13) to replace every b term in $(\tau_1)_j$ we can rewrite $(\tau_1)_j$ as

$$(\tau_1)_j = \frac{Re^2}{36}(u_{x^4})_j h^2 + \frac{Re}{60}(u_{x^5})_j h^3 + \frac{1}{360}(u_{x^8})_j h^4. \tag{3.14}$$

If I set $b = Re = 0$, the convection diffusion equation reduces to the Poisson equation. The truncation error $(\tau_1)_j$ will be reduced to

$$(\tau_1)_j = \frac{1}{360}(u_{x^8})_j h^4,$$

which yields an exactly fourth order truncation error.

When b increases, Re also increases. The second and third order terms in Eq. (3.14) will degrade the order of accuracy. When b is bigger than certain tolerance, I can consider $Re = bh/2$ as some constant. So, the order of accuracy for truncation error $(\tau_1)_j$ decreases to the second order. In practice, the computed order of accuracy of the FOC scheme varies from 4 to 2 as the Reynolds number increases.

Since the large convection coefficient (Reynolds number) will affect the accuracy of the computed solution, I need to compute the exact order of accuracy from the FOC scheme before I run the extrapolation procedure. In practice, we first get the maximum absolute errors $E(2h)$ and $E(h)$ over the entire discretized grid points for the coarse and the fine grids, respectively. These errors are easy to get because we can get the converged solutions from multiscale multigrid method for both the $2h$ and h grids. The accuracy order of the FOC scheme can be estimated by the following formula

$$\begin{aligned} &\frac{E(2h)}{E(h)} = \frac{(2h)^m}{h^m} \\ &\Longrightarrow m = \log_2(E(2h)/E(h)). \end{aligned} \tag{3.15}$$

For the truncation error $(\tau_2)_j$ of the sixth order scheme, it is also affected by large convection coefficient. For better understanding, I denote λ to be the ratio of $(\tau_1)_j$ to $(\tau_2)_j$, so λ can be used to represent the improvement of the extrapolation procedure. Generally, a larger λ means the extrapolation procedure improves the solution more accurately and a smaller λ means the extrapolation procedure cannot improve the accuracy much.

The ratio λ is related to the value of b. When $b \to +\infty$, λ can be computed as

$$\begin{aligned} \lambda &= \lim_{b\to+\infty} \frac{(\frac{b^2}{144}(u_{x^4})_j + \frac{b}{120}(u_{x^5})_j + \frac{1}{360}(u_{x^6})_j)h^4}{(\frac{b^2}{4320}(u_{x^6})_j + \frac{b}{5040}(u_{x^7})_j + \frac{1}{20160}(u_{x^8})_j)h^6} \\ &= \lim_{b\to+\infty} \frac{(\frac{b^2}{144}(u_{x^4})_j}{\frac{b^2}{4320}(u_{x^6})_j h^2} \\ &= \frac{30(u_{x^4})_j}{(u_{x^6})_j h^2}. \end{aligned}$$

If I consider $\frac{1}{\lambda}$, then I have

$$\frac{1}{\lambda} = \frac{(u_{x^6})_j}{30(u_{x^4})_j} h^2.$$

So, the accuracy improvement from the FOC scheme using the extrapolation scheme will level off at the second order when Re (b) is beyond some threshold.

3.2.2 Approximation for 2D problems

The FOC discretization for the 2D convection-diffusion is more complicated than the 2D Poisson equation, but the basic idea behind these high order schemes is: to find approximations of the second order terms in the truncation order $\tau_{i,j}$ using immediate neighboring grid points of (x_i, y_j). For the FOC scheme I used [24], the local truncation error by standard centered difference scheme is

$$\tau_{i,j} = \frac{h^2}{12}[-(\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4}) + 2(p(x,y)\frac{\partial^3 u}{\partial x^3} + q(x,y)\frac{\partial^3 u}{\partial y^3})]_{i,j} + O(h^4).$$

The truncation error is approximated by using eight neighboring grid points.

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

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

I use p_i, q_i and f_i $(i = 0, 1, ..., 8)$ to denote the function values at the corresponding grid points. The nine-point fourth order compact finite difference formula for the mesh point (x, y) can be written as

$$\sum_{j=0}^{8} \alpha_j u_j = \frac{h^2}{2}[8f_0 + f_1 + f_2 + f_3 + f_4] + \frac{h^3}{4}[p_0(f_1 - f_3) + q_0(f_2 - f_4)], \qquad (3.16)$$

where h is the mesh spacing, $\alpha_i (i = 0, 1, ..., 8)$ are the coefficients given by

$$
\begin{aligned}
\alpha_0 &= -[20 + h^2(p_0^2 + q_0^2) + h(p_1 - p_3) + h(q_2 - q_4)], \\
\alpha_1 &= 4 + \frac{h}{4}[4p_0 + 3p_1 - p_3 + p_2 + p_4] + \frac{h^2}{8}[4p_0^2 + p_0(p_1 - p_3) + q_0(p_2 - p_4)], \\
\alpha_2 &= 4 + \frac{h}{4}[4q_0 + 3q_2 - q_4 + q_1 + q_3] + \frac{h^2}{8}[4q_0^2 + p_0(q_1 - q_3) + q_0(q_2 - q_4)], \\
\alpha_3 &= 4 - \frac{h}{4}[4p_0 - p_1 + 3p_3 + p_2 + p_4] + \frac{h^2}{8}[4p_0^2 - p_0(p_1 - p_3) - q_0(p_2 - p_4)], \\
\alpha_4 &= 4 - \frac{h}{4}[4q_0 - q_2 + 3q_4 + q_1 + q_3] + \frac{h^2}{8}[4q_0^2 - p_0(q_1 - q_3) - q_0(q_2 - q_4)], \\
\alpha_5 &= 1 + \frac{h}{2}(p_0 + q_0) + \frac{h}{8}(q_1 - q_3 + p_2 - p_4) + \frac{h^2}{4}p_0 q_0, \\
\alpha_6 &= 1 - \frac{h}{2}(p_0 - q_0) - \frac{h}{8}(q_1 - q_3 + p_2 - p_4) - \frac{h^2}{4}p_0 q_0, \\
\alpha_7 &= 1 - \frac{h}{2}(p_0 + q_0) + \frac{h}{8}(q_1 - q_3 + p_2 - p_4) + \frac{h^2}{4}p_0 q_0, \\
\alpha_8 &= 1 + \frac{h}{2}(p_0 - q_0) - \frac{h}{8}(q_1 - q_3 + p_2 - p_4) - \frac{h^2}{4}p_0 q_0.
\end{aligned}
$$

When $p(x, y)$ and $q(x, y)$ are set to be some constants, Eq. (3.1) is called the constant coefficient convection-diffusion equation and the generated 9-point coefficients are the same for every grid point. Moreover, when $p(x, y) = q(x, y) \equiv 0$, Eq. (3.1) is reduced to the 2D Poisson equation, and Eq. (3.16) is reduced to the well-known Mehrstellen formula [24].

Extrapolation and operator based interpolation for large Re**.** In previous chapter, I proposed an operator based interpolation scheme combined with Richardson extrapolation technique to achieve sixth order accurate solution on the fine grid. The numerical results show that our method is very efficient and accurate for the 2D Poisson equation. I point out that the Poisson equation is a special case of the convection diffusion equation with $Re = 0$, which means that the order of accuracy of the computed solution for the Poisson equation will not be affected by the Reynolds number. In practice, when Re is very small, I just assume that I obtain the exact fourth

order accurate solutions from the FOC scheme and apply the fourth order Richardson extrapolation. For the convection diffusion equation with high Reynolds numbers, I need to consider the effect of the Reynolds number on the order of accuracy of the computed solutions.

Since the exact order of solution accuracy is related to the Reynolds number (less than 4 with large Re), I assume the order of solution accuracy is m, which is easy to be computed by Eq. (3.15). The Richardson extrapolation formula that I use can be written in the form of

$$\tilde{u}_{i,j}^{2h} = \frac{(2^m u_{2i,2j}^h - u_{i,j}^{2h})}{2^m - 1}. \tag{3.17}$$

Then I apply Algorithm 4.2.2 to approximate the $\tilde{m}_{th}$ $(m < \tilde{m} \leq 6)$ order solutions on fine grid.

3.3 Special Solution Strategies

Similar to the 2D Poisson equation, I use the multiscale multigrid method to solve the linear systems arising from discretization of the 2D convection-diffusion equation. I point out that using the point Gauss-Seidel relaxation in a standard multigrid method is efficient for solving elliptic problems like the Poisson equation and the convection diffusion equation with small Re [9], but the convergence will be slow with high Re. Several acceleration schemes have been developed to speed up the multigrid method, like the minimal residual smoothing method (MRS). Unfortunately, MRS combined with point relaxation in multigrid method still cannot achieve the grid independence for some high Reynolds number problems and it still needs more than 1000 iterations to converge when the magnitude of the Reynolds number is higher than 10^5 [71].

In order to achieve better efficiency and robustness, I will use alternating (X-Y) line Gauss-Seidel relaxation in our multigrid method. The X-Y line Gauss-Seidel relaxation in lexicographic order performs one sweep of line Gauss-Seidel relaxation

along the x-coordinate direction first, then another sweep of the line Gauss-Seidel relaxation along the y-coordinate direction. However, it was shown in [75, 79] that merely using the X-Y line Gauss-Seidel relaxation in a standard multigrid method does not provide fast convergence for convection dominated problems with high Reynolds number. The reason is that the coarse grid solution may not provide a meaningful correction to the fine grid computed solution with a small amount of artificial viscosity. One simple and efficient approach to fix this problem is to properly scale the residual before it is projected to the coarse grid [74].

3.3.1 Residual scaling technique

There were several research works using residual scaling techniques in [10, 74]. The theoretical basis for residual scaling is not clear. One possible reason is based on the fact that the FOC scheme add artificial viscosity to the discretized equations [74]. The amount of artificial viscosity is proportional to the meshsize h. Multigrid method used different h at different level of grids with same discretization scheme. For solving the convection-diffusion equations with very high Reynolds number, the solutions obtained from different levels do not have the same scale. So, the regular coarse grid solution in multigrid may not be correct to improve the fine grid solutions. One simple approach is to scale the residual explicitly before it is projected to the coarse grid, or after the coarse grid correction is interpolated to the fine grid. These two strategies are called prescaling and postscaling techniques, which are mathematically equivalent. In this dissertation, I use prescaling technique.

The residual scaling procedure at a grid point (x_i, y_j) can be written in the form of

$$\tilde{r}(x_i, y_j) = \beta r(x_i, y_j), \tag{3.18}$$

where β is the scaling factor, $r(x_i, y_j)$ is the residual from the fine grid and $\tilde{r}(x_i, y_j)$ is the residual after scaling. Let's assume the scaling factor β has been determined,

the multigrid method V-cycle with prescaling accelerating technique in our multiscale multigrid method can be modified to the following algorithm:

Algorithm 5 k_{th} iteration of $MGV - cycle(A_h, u_h, b_h)$ with residual prescaling technique

1: If Ω_h is the coarsest level grid, directly solve $u_h^{(k)} = (A_h)^{-1} b_h$, else goto step 2.
2: Relax ν_1 times on $A_h u_h = b_h$ with initial guess $u_h^{(k)}$.
3: Compute $r_h^{(k)} = b_h - A_h u_h^{(k)}$.
4: Scale $r_h^{(k)} = \beta r_h^{(k)}$.
5: Restrict residual from Ω_h to Ω_{2h} as $b_{2h}^{(k)} = r_{2h}^{(k)} = I_h^{2h} r_h^{(k)}$.
6: Solve $u_{2h}^{(k)} = MGV - cycle(A_{2h}, u_{2h}^{(k)}, b_{2h}^{(k)})$ with initial guess $u_{2h}^{(k)} = 0$.
7: Correct $u_h^{(k*)} = u_h^{(k)} + I_{2h}^h u_{2h}^{(k)}$.
8: Relax ν_2 times on $A_h u_h = b_h$ with initial guess $u_h^{(k*)}$.

In many practical applications, the optimal residual scaling factor is determined by the absolute values of the convection coefficients $|p(x, y)|$ and $|q(x, y)|$ at the reference grid point and the scaling factor is a function of the grid point (x_i, y_j). In [72], Zhang suggested that the optimal residual scaling factor lie in an interval if $|p(x, y)| > 10^3$ or $|q(x, y)| > 10^3$. For the convection-diffusion equations with stagnation points, the residual should be set to zero at these stagnation points. That's because the feature of the stagnation points need to be represented on the coarse grids, otherwise the multigrid method may diverge [10].

The detail of how to choose an optimal residual scaling factor with high Reynolds number can be found in [75, 79]. In our experiment, I tested several scaling factors and only list the numerical results from the best one in Section 3.4.

3.4 Numerical Results

I tested our sixth order compact scheme (SOC) and compared the results with the standard fourth order compact difference scheme (FOC) and Sun-Zhang's sixth order method (REC) [59].

For all test problems, I first compared the computed solution accuracy and the

CPU cost for different strategies with some fixed convection coefficients on different meshsizes. Then, I tested the effect of Re on the computed solution accuracy and the CPU cost for different strategies with a fixed meshsize. The graphic comparison of the numerical data will also be provided for each test case.

3.4.1 Test problem 1

We first considered a case in which the convection coefficients $p(x, y)$ and $q(x, y)$ are polynomials in x and y. The test case is

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

and the domain $\Omega = (0, 1) \times (0, 1)$. The boundary values of the solution are assumed to be known and the initial guess is $u(x, y) = 0$. Here, I only use one constant P to scale both variable coefficients $p(x, y)$ and $q(x, y)$ because the magnitudes of these two functions vary differently in the domain Ω.

Table 3.1 shows the number of iterations, maximum errors and the order of solution accuracy for different solution strategies with different meshsizes. For $P = 10$, the cell Reynolds number is small, we can see that the order of solution accuracy for the FOC scheme is nearly 4 as I expected. The numerical results illustrate that the SOC scheme solved the problem with better accuracy than the FOC scheme did, and the order of solution accuracy was close to 6. The results also indicate that the computed solutions from the SOC scheme and the Sun-Zhang's REC method were comparable, but the SOC scheme took much less CPU time. The number of iterations for the REC method increased very quickly when the mesh became finer. In addition, when $n > 64$, it did not converge within the maximum number of iterations (5000) I set. On the other hand, by using the X-Y line relaxation in the multigrid method

combined with residual scaling technique, the convergence rate of the FOC scheme and the SOC scheme was independent of the grid size.

Table 3.1: Test Problem 1: Comparison of CPU cost and solution accuracy with different meshsizes and fixed P values.

		$P = 10$				$P = 1000$			
n	strategy	# iter	CPU	error	order	# iter	CPU	error	order
16	REC	132	0.011	2.41e-6	5.66	58	0.005	4.20e-3	3.17
	FOC	6	0.003	3.45e-5	4.06	16	0.004	6.17e-3	2.85
	SOC	(6,6),18	0.004	2.41e-6	5.66	(13,14),96	0.004	4.19e-3	3.17
32	REC	518	0.173	4.36e-8	5.79	47	0.016	1.36e-4	4.95
	FOC	7	0.008	2.13e-6	4.02	19	0.013	4.15e-4	3.89
	SOC	(6,6),17	0.007	4.36e-8	5.79	(14,17),64	0.017	1.36e-4	4.95
64	REC	2064	2.845	7.47e-10	5.86	44	0.081	2.97e-6	5.51
	FOC	7	0.025	1.33e-7	4.00	19	0.055	2.57e-5	4.01
	SOC	(6,7),15	0.032	7.36e-10	5.89	(17,17),35	0.076	2.97e-6	5.51
128	REC	not converge	–	–	–	132	1.150	5.71e-8	5.70
	FOC	7	0.105	8.29e-9	4.00	17	0.358	1.60e-6	4.01
	SOC	(7,7),13	0.124	1.25e-11	5.87	(17,14),21	0.457	5.71e-8	5.70
256	REC	not converge	–	–	–	490	20.591	9.98e-10	5.83
	FOC	7	0.612	5.18e-10	4.00	15	1.146	1.00e-7	4.00
	SOC	(7,7),15	1.190	1.97e-13	5.98	(14,13),18	1.409	9.95e-10	5.84

When the magnitude of the convection coefficients increased, i.e., when $P = 1000$, we found that the order of solution accuracy from the FOC scheme was reduced as I expected, especially for $n = 16$. It is clear that the SOC scheme still increased the solution accuracy when Re increased. Since Re is a function of the meshsize h, convergence improved when h was refined. We can see that the number of V-Cycles for both the FOC scheme and the SOC scheme decreased when n increased and the REC method converged for all the meshsizes I tested. Fig. 3.1 gives a comparison of the CPU cost and the maximum errors with different meshsizes.

Table 3.2 contains the results with various Re with a fixed meshsize $h = 1/64$. I compared the CPU cost and the maximum errors for the FOC scheme and the SOC scheme. I note that the magnitude of the Reynolds number affected the solution accuracy and the convergence rate for the FOC scheme and the SOC scheme in-

Table 3.2: Test Problem 1: Comparison of CPU cost and solution accuracy with different P values for a fixed meshsize.

n=64		FOC			SOC		
P	Re	# iter	CPU	error	# iter	CPU	error
0	0.0	7	0.027	4.58e-12	(6,7),2	0.028	1.02e-13
1	7.8125e-3	7	0.027	1.01e-8	(6,7),12	0.031	6.42e-11
10	7.8125e-2	7	0.025	1.33e-7	(6,7),15	0.032	7.36e-10
10^2	7.8125e-1	9	0.038	2.11e-6	(8,8),19	0.041	3.99e-8
10^3	7.8125e0	19	0.055	2.57e-5	(57,56),35	0.055	2.97e-6
10^4	7.8125e1	73	0.186	2.63e-4	(32,40),526	0.293	8.87e-5
10^5	7.8125e2	44	0.115	1.03e-3	(32,41),1709	0.572	1.62e-4
10^6	7.8125e3	45	0.119	1.13e-3	(32,41),1803	0.595	1.52e-4
10^7	7.8125e4	44	0.123	1.13e-3	(32,41),1810	0.592	1.52e-4
10^8	7.8125e5	44	0.121	1.13e-3	(32,41),1811	0.603	1.52e-4
10^9	7.8125e6	44	0.119	1.13e-3	(32,41),1811	0.587	1.52e-4
10^{10}	7.8125e7	44	0.124	1.13e-3	(32,41),1811	0.598	1.52e-4

versely. When Re was small ($Re \leq 1$), these methods converged rapidly and yielded reasonably accurate solutions. When Re increased, the solution accuracy and the convergence rate severely deteriorated.

We note that there was only a little change for the accuracy and the number of iterations for the FOC and SOC schemes when $P > 10^5$. However, the SOC scheme still yielded better accuracy than the FOC scheme, but the improvement degraded when Re increased, as I expected. The ratio of the fourth order error to the sixth order error with different P is shown in Fig. 3.2. We can see that the ratio is close to a certain constant when $P \to +\infty$.

3.4.2 Test problem 2

I chose the coefficients as multiples of the trigonometric functions

$$\begin{cases} u(x,y) &= \cos(4x+6y), \\ p(x,y) &= P\sin(\pi x), \\ q(x,y) &= P\cos(\pi y), \end{cases}$$

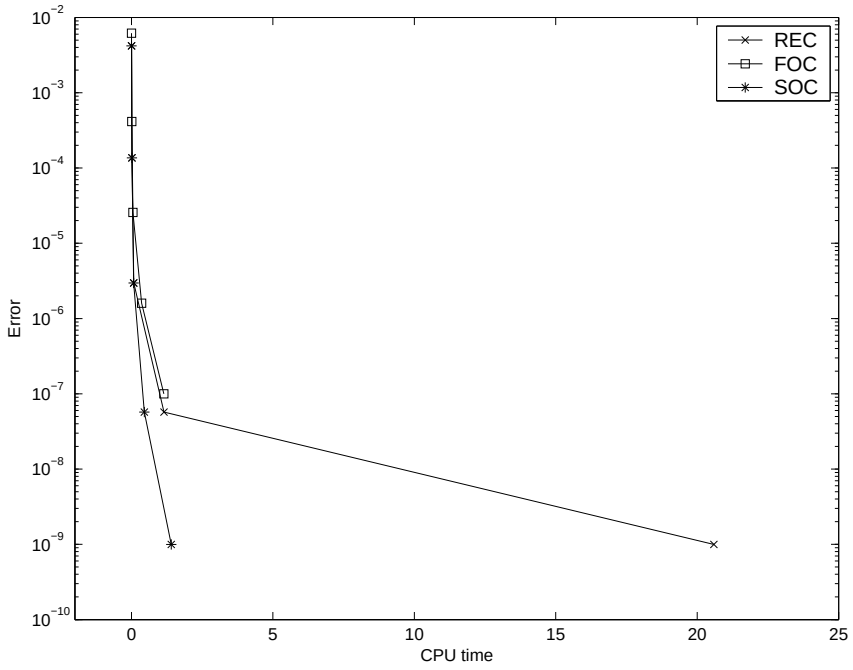


Figure 3.1: Comparison of the maximum errors and the CPU costs for the Problem 1 ($P = 1000$). Each symbol with increasing CPU time corresponds to an increasing fine grid: 16, 32, 64, 128, and 256 intervals.

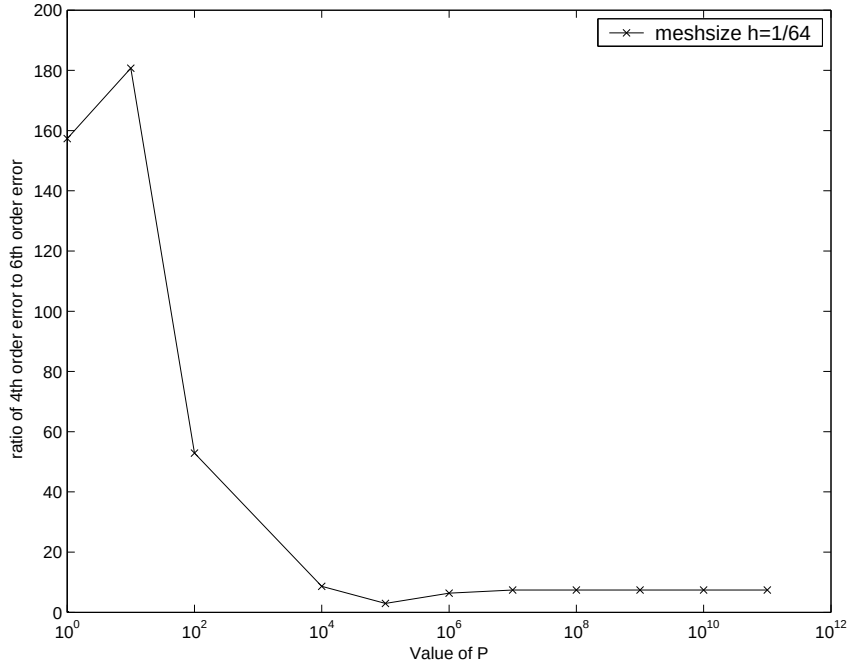


Figure 3.2: Ratio of the fourth order error to the sixth order error for the Problem 1.

where $\Omega = (0,1) \times (0,1)$. The test conditions were set to be the same as for the test Problem 1.

Table 3.3: Test Problem 2: Comparison of CPU cost and solution accuracy with different meshsizes and fixed P values.

		$P = 10$				$P = 1000$			
n	strategy	# iter	CPU	error	order	# iter	CPU	error	order
16	REC	94	0.007	2.30e-5	5.44	331	0.025	5.69e-3	2.02
	FOC	9	0.005	1.36e-4	4.19	24	0.008	6.25e-3	2.01
	SOC	(8,8),24	0.003	2.30e-5	5.44	(32,23),29	0.006	5.69e-3	2.02
32	REC	364	0.121	4.21e-7	5.77	294	0.098	5.38e-4	3.40
	FOC	9	0.012	7.97e-6	4.09	60	0.041	7.65e-4	3.03
	SOC	(8,8),24	0.008	4.21e-7	5.77	(23,51),31	0.036	5.38e-4	3.40
64	REC	1445	2.079	6.95e-9	5.92	224	0.305	1.90e-5	4.82
	FOC	9	0.041	4.82e-7	4.05	70	0.256	5.63e-5	3.76
	SOC	(8,8),23	0.043	6.95e-9	5.92	(51,63),30	0.197	1.90e-5	4.82
128	REC	not converge	–	–	–	130	1.532	3.36e-7	5.82
	FOC	9	0.261	2.96e-8	4.03	67	1.491	3.57e-6	3.97
	SOC	(8,8),21	0.241	1.11e-10	5.96	(63,63),28	1.587	3.36e-7	5.82
256	REC	not converge	–	–	–	419	17.708	6.54e-9	5.68
	FOC	9	1.209	1.84e-9	4.01	64	6.821	2.22e-7	4.01
	SOC	(8,9),19	1.594	1.74e-12	6.00	(64,52),26	4.774	6.54e-9	5.68

The numerical data with comparison are shown in Table 3.3, Table 3.4, Fig. 3.3 and Fig. 3.4. For $P = 10$, the SOC scheme achieved the sixth order solution accuracy and the convergence rate of our multiscale multigrid method was independent of the grid size. Once again, our sixth order method solved the problem with the same accuracy as Sun-Zhang's REC method did, but took much less CPU time. In addition, the REC method still could not converge when $n > 64$.

For $P = 1000$, it seems that the magnitude of Re affected the order of accuracy more than Problem 1 when n was smaller than 64. When n increased, the number of iterations for the FOC scheme and the REC method decreased, which once again showed that the magnitude of the Reynolds number affected the convergence rate inversely.

Similar to Problem 1, Table 3.4 compares results for different Re with a fixed meshsize. We note that the SOC scheme yielded more accurate solutions compared

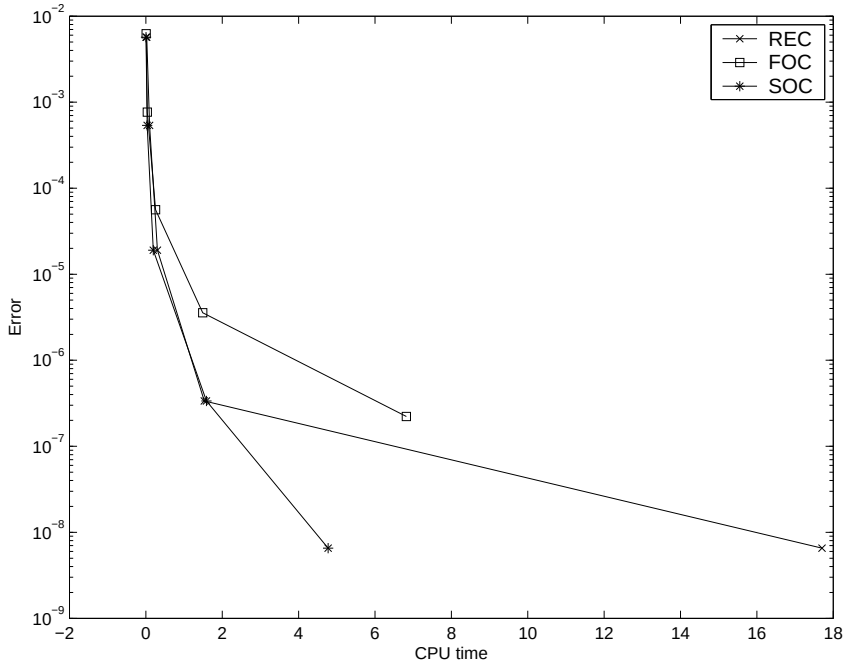


Figure 3.3: Comparison of the maximum errors and the CPU costs for the Problem 2 ($P = 1000$). Each symbol with increasing CPU time corresponds to an increasing fine grid: 16, 32, 64, 128, and 256 intervals.

to the FOC scheme and it also costed less CPU time than the FOC scheme did when $P \geq 10^5$. And it once again showed the convergence rate and the computed accuracy approached some limits and did not deteriorate any more when Re was beyond some threshold.

3.5 Concluding Remarks

I extended the idea of integrating high accuracy discretization into the multigrid method from [66] to solve the 2D convection diffusion equation. In order to compute highly accurate solutions for the problems with high Reynolds numbers, I modified the operator based interpolation scheme to use the correct order of accuracy from the FOC schemes to perform the Richardson extrapolation. I also used the X-Y line relaxation combined with residual scaling technique in our multiscale multigrid method to achieve the grid independent convergence.

Table 3.4: Test Problem 2: Comparison of CPU cost and solution accuracy with different P values for a fixed meshsize.

n=64		FOC			SOC		
P	Re	# iter	CPU	error	# iter	CPU	error
0	0.0	9	0.040	2.93e-9	(8,8),16	0.039	1.59e-10
1	7.8125e-3	9	0.041	3.81e-8	(8,8),19	0.040	4.72e-10
10	7.8125e-2	9	0.041	4.82e-7	(8,8),23	0.043	6.95e-9
10^2	7.8125e-1	14	0.055	5.17e-6	(15,14),25	0.065	4.34e-7
10^3	7.8125e0	70	0.256	5.63e-5	(51,63),30	0.197	1.90e-5
10^4	7.8125e1	108	0.337	3.45e-4	(76,93),33	0.358	2.03e-4
10^5	7.8125e2	109	0.367	5.01e-4	(75,94),34	0.363	1.58e-4
10^6	7.8125e3	113	0.381	4.81e-4	(75,92),34	0.352	1.48e-4
10^7	7.8125e4	113	0.372	4.79e-4	(75,92),34	0.347	1.49e-4
10^8	7.8125e5	113	0.369	4.79e-4	(75,92),34	0.356	1.49e-4
10^9	7.8125e6	113	0.376	4.79e-4	(75,92),34	0.349	1.49e-4
10^{10}	7.8125e7	113	0.378	4.79e-4	(75,92),34	0.361	1.49e-4

The test results showed that the sixth order compact scheme is efficient, robust and accurate. It computed more accurate solutions than the FOC scheme, and the CPU cost was comparable. For some high Reynolds number cases, the SOC scheme even took less computation time than the FOC scheme did.

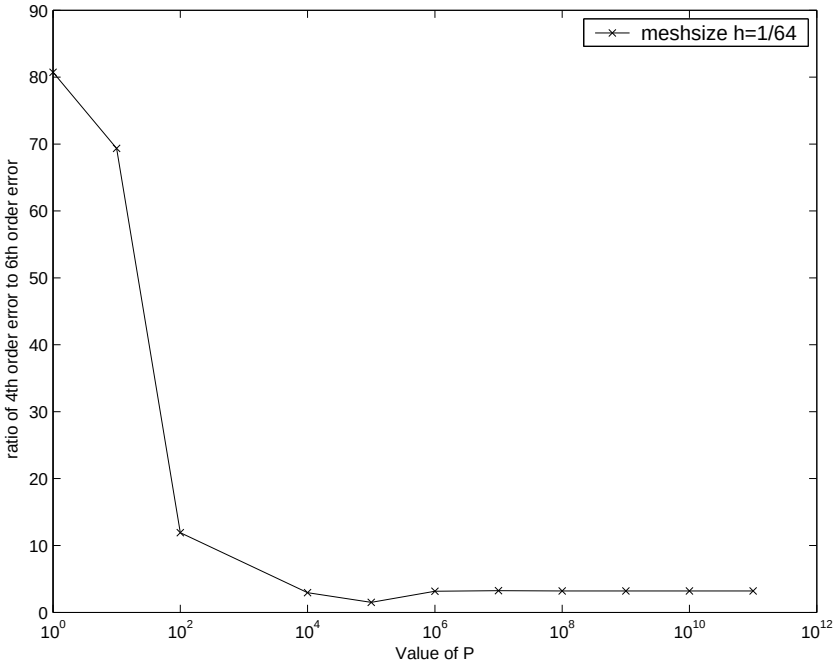


Figure 3.4: Ratio of the fourth order error to the sixth order error for the Problem 2.

Chapter 4

Sixth Order Solution for 3D Convection Diffusion Equation

In many practical CSEI applications, the numerical simulation of three dimensional (3D) elliptic partial differential equations tends to be computationally intensive due to its huge requirements on the memory and CPU time to compute solutions with desired accuracy. The reason is that the size of resulting linear system for 3D problems is usually so large that even super-computer may not be able to handle these computing. One simple example is that solving a 3D equation with $N_x = N_y = N_z = 100$ will generate one million unknowns for the discrete linear systems. Since the traditional numerical methods have low accuracy and need extremely fine discretization, I want to develop a high-order discretization method combined with powerful linear system solver to compute approximate solutions with high accuracy using coarser discretization.

In previous two chapters, I have designed an efficient sixth order compact finite difference schemes combined with Richardson extrapolation and operator based interpolation scheme. The sixth order scheme has good numerical stability and provides high accuracy approximations. In this chapter, I will extend the sixth order discretization scheme from two dimensions to three dimensions. For solving the discrete linear system with grid independent convergence, I introduce the plane relaxation technique in our multiscale multigrid method.

4.1 Introduction

I consider the three dimensional (3D) convection diffusion equation

$$u_{xx} + u_{yy} + u_{zz} + p(x,y,z)u_x + q(x,y,z)u_y + r(x,y,z)u_z = f(x,y,z), \tag{4.1}$$

for a specified forcing function f in a continuous domain Ω of a 3D space with suitable boundary conditions prescribed on $\partial\Omega$. Here Ω is assumed to be comprised of a union of rectangular solids. Functions p, q, r, f, and u are assumed to be continuously differentiable and have the required partial derivatives on Ω. The cell Reynolds number for 3D problems can be computed as

$$Re = \max(\sup_{(x,y,z)\in\Omega} |p(x,y,z)|, \sup_{(x,y,z)\in\Omega} |q(x,y,z)|, \sup_{(x,y,z)\in\Omega} |p(x,y,z)|)h/2. \tag{4.2}$$

Like 2D problems, traditional numerical discretization schemes for the 3D convection-diffusion equations usually employ centered difference for the second order diffusion terms and upwind difference for the first order convection terms. Once again, it has been proved that high order difference schemes can achieve good numerical stability and high accuracy approximations. For the fourth order finite difference schemes for 3D problems, there are several strategies to derive [3, 26, 55, 75]. These approximations fall into two categories. The first one is introduced by Gupta, etc. [26] using truncated Taylor series expansions. The finite difference formulas are obtained by linear combination over a set of neighboring points surrounding the given mesh point. Discretization using this strategy is straightforward but it becomes very difficult in higher dimensions. The other one was developed by Spotz and Carey [55]. They obtained high order discretization for particular equations by employing central difference scheme repeatedly until the desired solution accuracy was reached.

The sixth order compact difference schemes for 3D convection-diffusion equations

with variable coefficients are extremely difficult to develop due to the need for extensive algebraic manipulations. As far as I know, there is no sixth order compact scheme in a single scale. I provided a two scale grid method to approximate the sixth order solutions [68].

4.2 Finite Approximation for 3D Problems

The basic idea is the same as the 2D problem, but the 3D operator based interpolation scheme combined with the Richardson extrapolation technique is different from the one for 2D problem.

4.2.1 Fourth order discretization

I assume that discretization is done on a cubic cell with meshsize h. I use u_0 to denote the approximate value of $u(x, y, z)$ at an internal mesh point (i, j, k). For each grid point, there are 26 neighboring grid points in the cubic as in Fig. 4.1.

Based on the truncated Taylor series expansions there are two computational stencils available: 19-points computational stencil and 15-points computational stencil. These two stencils use different groups of grid points from the 27 grid points in a cubic. Both Gupta and Zhang have proposed these two computational stencils to discretize the 3D convection-diffusion equations [26, 75, 78].

19-point scheme vs. 15-point scheme. For convenience I divide the grid points by their indexes in Fig. 4.1 into three groups: $group\ A = \{0-6\}$, $group\ B = \{7-18\}$, $group\ C = \{19-26\}$. The 19-point compact scheme utilizes the grid points in groups A and B. The 15-point compact scheme utilizes the grid points in groups A and C.

The 15-point compact scheme can be derived by considering cross derivatives of the same order together and by utilizing their symmetry relation. This is different from the strategies used for deriving 19-point scheme, in which the cross derivatives

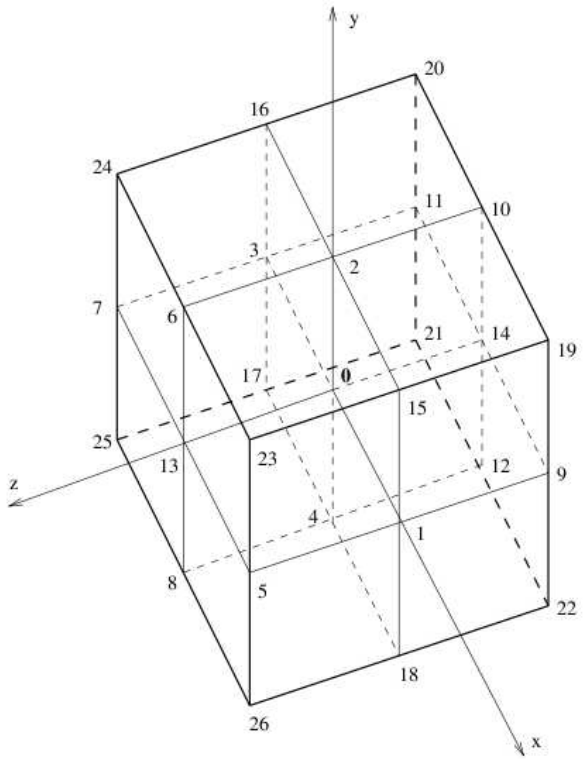


Figure 4.1: Labeling of the 27 grid points in a unit cube.

are approximated individually. Considering the memory cost, the 15-point scheme is more attractive than the 19-point one because it requires less grid points. However, the truncation error of 19-point scheme is smaller than that of 15-point scheme.

From [78] I know that the truncation error of the 19-point compact scheme is of the form

$$\begin{aligned}\tau_{19} =& h^4(-\frac{1}{120}(p\frac{\partial^5 u}{\partial x^5}+q\frac{\partial^5 u}{\partial y^5}+r\frac{\partial^5 u}{\partial z^5})-\frac{1}{144}(p^2\frac{\partial^4 u}{\partial x^4}+q^2\frac{\partial^4 u}{\partial y^4}+r^2\frac{\partial^4 u}{\partial z^4})\\ &-\frac{1}{36}(pq\frac{\partial^4 u}{\partial y^3\partial x}+pq\frac{\partial^4 u}{\partial y\partial x^3}+qr\frac{\partial^4 u}{\partial z\partial y^3}+qr\frac{\partial^4 u}{\partial z^3\partial y}+pr\frac{\partial^4 u}{\partial z^3\partial x}+pr\frac{\partial^4 u}{\partial z\partial x^3})\\ &-\frac{1}{72}(p\frac{\partial^5 u}{\partial y^4\partial x}+q\frac{\partial^5 u}{\partial y\partial x^4}+r\frac{\partial^5 u}{\partial z\partial x^4}+p\frac{\partial^5 u}{\partial z^4\partial x}+q\frac{\partial^5 u}{\partial z^4\partial y}+r\frac{\partial^5 u}{\partial z\partial y^4})\\ &-\frac{1}{36}(p\frac{\partial^5 u}{\partial y^2\partial x^3}+q\frac{\partial^5 u}{\partial z^2\partial y^3}+r\frac{\partial^5 u}{\partial z^3\partial y^2}+p\frac{\partial^5 u}{\partial z^2\partial x^3}+q\frac{\partial^5 u}{\partial y^3\partial x^2}+r\frac{\partial^5 u}{\partial z^3\partial x^2})\\ &-\frac{1}{72}(\frac{\partial^6 u}{\partial y^4\partial x^2}+\frac{\partial^6 u}{\partial y^2\partial x^4}+\frac{\partial^6 u}{\partial z^2\partial x^4}+\frac{\partial^6 u}{\partial z^4\partial x^2}+\frac{\partial^6 u}{\partial z^4\partial y^2}+\frac{\partial^6 u}{\partial z^2\partial y^4})\\ &-\frac{1}{360}(\frac{\partial^6 u}{\partial x^6}+\frac{\partial^6 u}{\partial y^6}+\frac{\partial^6 u}{\partial z^6})),\end{aligned}$$

and the truncation error of the 15-point compact scheme is in the form of

$$\begin{aligned}\tau_{15} =& \tau_{19}-\frac{1}{12}(pq\frac{\partial^4 u}{\partial z^2\partial y\partial x}+pr\frac{\partial^4 u}{\partial z\partial y^2\partial x}+qr\frac{\partial^4 u}{\partial z\partial y\partial x^2}\\ &+p\frac{\partial^5 u}{\partial z^2\partial y^2\partial x}+q\frac{\partial^5 u}{\partial z^2\partial y\partial x^2}+r\frac{\partial^5 u}{\partial z\partial y^2\partial x^2}+\frac{\partial^6 u}{\partial z^2\partial y^2\partial x^2}).\end{aligned}$$

From above two formulas, I note that τ_{15} contains all terms of τ_{19} and some cross derivatives with respect to all three variables. The largest coefficient factor for τ_{15} is almost three times that of τ_{19}. By using Eq. (4.2), I can rewrite these two truncations errors as $\tau_{19} = O(\frac{1}{36}Re^2)$ and $\tau_{15} = O(\frac{1}{12}Re^2)$. When Re becomes large, the magnitude of the convection coefficient affect the order of accuracy inversely and the 15-point scheme becomes worse than the 19-point scheme.

I want to point out here that if I set one of these three variables to be constant, all the cross derivatives become zero. In that case, the truncation errors of both 19-point

scheme and 15-point scheme are equal.

19-point compact finite difference scheme. I choose 19-point scheme because it will give us more accurate solutions for 3D convection-diffusion equations with large cell Reynolds numbers. The involving immediate 18 neighboring points are denoted by u_l, $l = 1, 2, ..., 18$, as in Fig. 4.2. The 8 corner points, which are the white colored points in Fig. 4.2, 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.

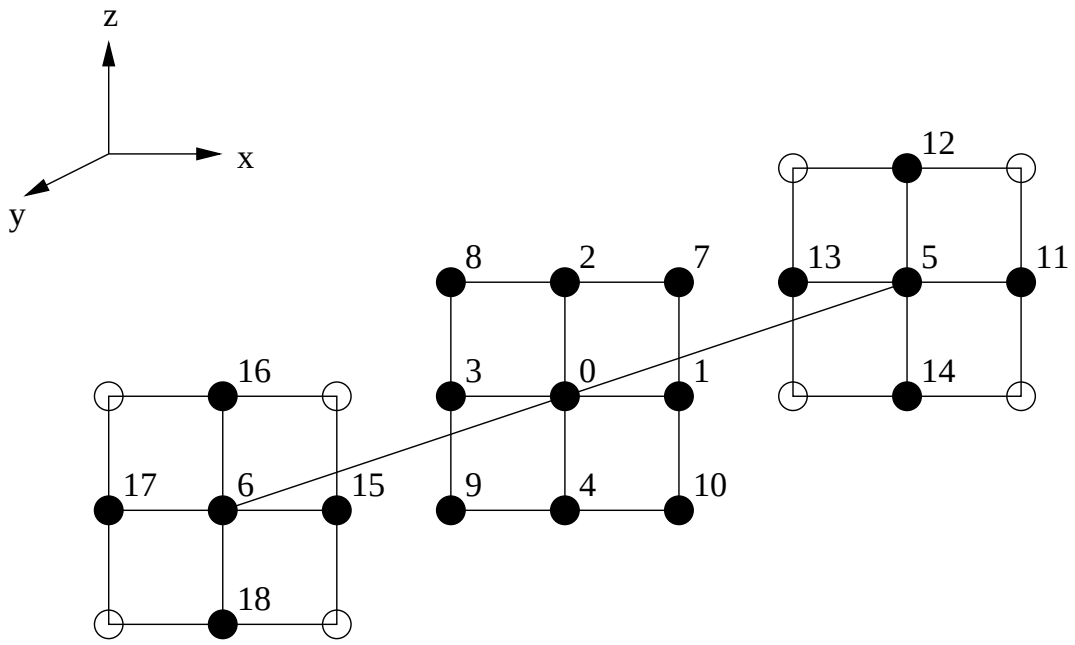


Figure 4.2: 19 point computational stencil.

I used Zhang's explicit fourth order compact scheme for Eq. (4.1), which was derived from the general implicit formula by Ananthakrishnaiah et al. [3]. It yields the following 19-point formula

$$\sum_{l=0}^{18} \alpha_l u_l = F_0, \tag{4.3}$$

where the coefficients α_l and the right hand side F_0 are given by

$$
\begin{aligned}
\alpha_0 = & -[24 + h^2(p_0^2 + q_0^2 + r_0^2) + h(p_1 - p_3 + q_2 - q_4 + r_5 - r_6)], \\
\alpha_1 = & \, 2 - \frac{h}{4}(2p_0 - 3p_1 - p_2 + p_3 - p_4 - p_5 - p_6) \\
& + \frac{h^2}{8}[4p_0^2 + p_0(p_1 - p_3) + q_0(p_2 - p_4) + r_0(p_5 - p_6)], \\
\alpha_2 = & \, 2 - \frac{h}{4}(2q_0 - q_1 - 3q_2 - q_3 + q_4 - q_5 - q_6) \\
& + \frac{h^2}{8}[4q_0^2 + p_0(q_1 - q_3) + q_0(q_2 - q_4) + r_0(q_5 - q_6)], \\
\alpha_3 = & \, 2 + \frac{h}{4}(2p_0 + p_1 - p_2 - 3p_3 - p_4 - p_5 - p_6) \\
& + \frac{h^2}{8}[4p_0^2 - p_0(p_1 - p_3) - q_0(p_2 - p_4) - r_0(p_5 - p_6)], \\
\alpha_4 = & \, 2 + \frac{h}{4}(2q_0 - q_1 + q_2 - q_3 - 3q_4 - q_5 - q_6) \\
& + \frac{h^2}{8}[4q_0^2 - p_0(q_1 - q_3) - q_0(q_2 - q_4) - r_0(q_5 - q_6)], \\
\alpha_5 = & \, 2 - \frac{h}{4}(2r_0 - r_1 - r_2 - r_3 - r_4 - 3r_5 + r_6) \\
& + \frac{h^2}{8}[4r_0^2 + p_0(r_1 - r_3) + q_0(r_2 - r_4) + r_0(r_5 - r_6)], \\
\alpha_6 = & \, 2 + \frac{h}{4}(2r_0 - r_1 - r_2 - r_3 - r_4 + r_5 - 3r_6) \\
& + \frac{h^2}{8}[4r_0^2 - p_0(r_1 - r_3) - q_0(r_2 - r_4) - r_0(r_5 - r_6)], \\
\alpha_7 = & \, 1 + \frac{h}{2}(p_0 + q_0) + \frac{h}{8}(p_2 - p_4 + q_1 - q_3) + \frac{h^2}{4}p_0q_0, \\
\alpha_8 = & \, 1 - \frac{h}{2}(p_0 - q_0) - \frac{h}{8}(p_2 - p_4 + q_1 - q_3) - \frac{h^2}{4}p_0q_0, \\
\alpha_9 = & \, 1 - \frac{h}{2}(p_0 + q_0) + \frac{h}{8}(p_2 - p_4 + q_1 - q_3) + \frac{h^2}{4}p_0q_0, \\
\alpha_{10} = & \, 1 + \frac{h}{2}(p_0 - q_0) - \frac{h}{8}(p_2 - p_4 + q_1 - q_3) - \frac{h^2}{4}p_0q_0,
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{11} &= 1 + \frac{h}{2}(p_0 + r_0) + \frac{h}{8}(p_5 - p_6 + r_1 - r_3) + \frac{h^2}{4} p_0 r_0, \\
\alpha_{12} &= 1 + \frac{h}{2}(q_0 + r_0) + \frac{h}{8}(q_5 - q_6 + r_2 - r_4) + \frac{h^2}{4} q_0 r_0, \\
\alpha_{13} &= 1 - \frac{h}{2}(p_0 - r_0) - \frac{h}{8}(p_5 - p_6 + r_1 - r_3) - \frac{h^2}{4} p_0 r_0, \\
\alpha_{14} &= 1 - \frac{h}{2}(q_0 - r_0) - \frac{h}{8}(q_5 - q_6 + r_2 - r_4) - \frac{h^2}{4} q_0 r_0, \\
\alpha_{15} &= 1 + \frac{h}{2}(p_0 - r_0) - \frac{h}{8}(p_5 - p_6 + r_1 - r_3) - \frac{h^2}{4} p_0 r_0, \\
\alpha_{16} &= 1 + \frac{h}{2}(q_0 - r_0) - \frac{h}{8}(q_5 - q_6 + r_2 - r_4) - \frac{h^2}{4} q_0 r_0, \\
\alpha_{17} &= 1 - \frac{h}{2}(p_0 + r_0) + \frac{h}{8}(p_5 - p_6 + r_1 - r_3) + \frac{h^2}{4} p_0 r_0, \\
\alpha_{18} &= 1 - \frac{h}{2}(q_0 + r_0) + \frac{h}{8}(q_5 - q_6 + r_2 - r_4) + \frac{h^2}{4} q_0 r_0, \\
F_0 &= \frac{h^2}{2}(6f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6) \\
&\quad + \frac{h^3}{4}[p_0(f_1 - f_3) + q_0(f_2 - f_4) + r_0(f_5 - f_6)].
\end{aligned}
$$

If I set the convection coefficients $p = q = r \equiv 0$, Eq. (4.1) will be reduced to 3D Poisson equation, which has also been studied by several authors like Kwon et al. [31], Spotz and Carey [56]. Eq. (4.3) can be used for every grid point, no special formulas are needed for approximations at grid points near the boundaries.

By using Eq. (4.3), for every discrete grid point, I obtain a system of linear equations

$$Au = b. \tag{4.4}$$

Like 2D problems, the coefficient matrix A is very large and sparse. It is nonsymmetric and nonpositive definite if the convection coefficients are nonzero. When the convection diffusion equation is convection dominated, the matrix A loses its diagonal dominance. Even without the diagonal dominance, our previous work showed that there is no stability difficulty with the FOC scheme for both the 2D and 3D

convection diffusion equations, and the multigrid method converges by using efficient smoothers and residual scaling techniques [27, 67, 72].

4.2.2 3D operator based interpolation

For 2D problems, I proposed an operator based interpolation scheme combined with extrapolation technique to approximate the sixth order accurate fine grid solution [66, 67]. The numerical results show that our interpolation scheme is very efficient and accurate for 2D problems.

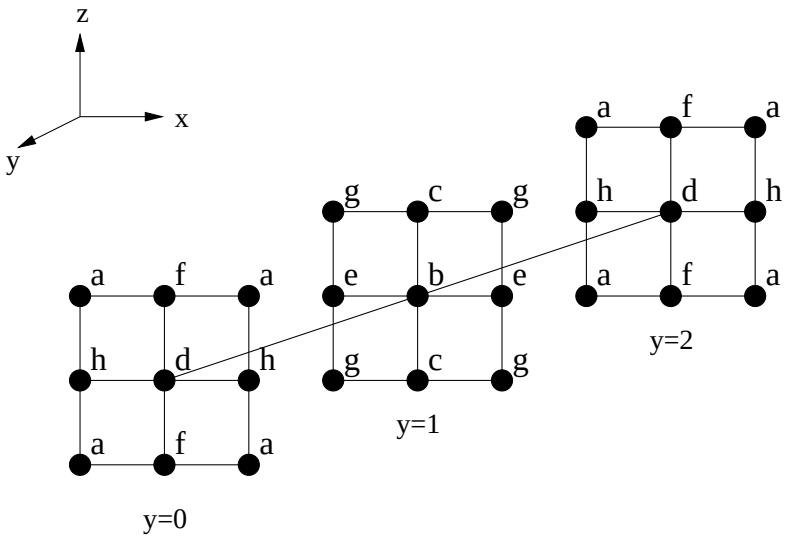
The interpolation scheme for 2D problems is an iterative procedure combined with the Richardson extrapolation technique, which updates the solutions of grid points by groups in each iteration. For the 3D convection diffusion equation, the basic idea behind the interpolation scheme is almost the same as that for the 2D problems, but it needs more complicated grouping strategy. Like in Fig. 4.3, I divide the fine grid points into eight different groups by their *odd* or *even* indexing in the x, y and z-coordinate directions. Group a contains the $(even, even, even)$ grid points on the Ω_h grid, which are the corresponding grid points on the Ω_{2h} coarse grid.

Similar to 2D convection-diffusion equation, by solving the system of linear equations arising from the FOC scheme, we can get the fourth order solutions $u^h_{i,j,k}$ and $u^{2h}_{i,j,k}$ on Ω_h and Ω_{2h}, respectively. For the sixth order accurate solutions, it is easy to approximate the solution for the coarse grid Ω_{2h} by using the Richardson extrapolation like

$$\tilde{u}^{2h}_{i,j,k} = \frac{2^m u^h_{2i,2j,2k} - u^{2h}_{i,j,k}}{2^m - 1}. \tag{4.5}$$

Here, m is the order of the computed solution accuracy from the FOC scheme. From [67], we know that the magnitude of the convection coefficients will affect the value of m and m is computed as I introduced in Chapter 3.

For the sixth order fine grid solution, I directly interpolate the sixth order coarse grid solution $\tilde{u}^{2h}_{i,j,k}$ to the corresponding grid points in group a. Then I use an iterative



a : (even,even,even)
e : (even,odd,odd)
b : (odd,odd,odd)
f : (odd,even,even)
c : (odd,odd,even)
g : (even,odd,even)
d : (odd,even,odd)
h : (even,even,odd)

Figure 4.3: Group information of 3D grid points in a unit cube.

mesh refinement interpolation technique to approximate sixth order solutions for the fine grid points in other groups. The details of one iteration (from step n to step $n+1$) is outlined in Algorithm 4.2.2.

Algorithm 4.2.2: Sixth order operator based interpolation scheme for 3D convection diffusion equation.

1: Let $u^h_{old} = \tilde{u}^{h,n}$.

2: If $n = 0$, goto Step 3, else goto Step 4.

3: *Update every grid point in group a on* Ω_h.

From $u^{2h}_{i,j,k} \in \Omega^4_{2h}$ and $u^h_{2i,2j,2k} \in \Omega^4_h$, I first compute $\tilde{u}^{2h,n+1}_{i,j,k} \in \Omega^6_{2h}$ by Eq. (4.5). Then I use direct interpolation to obtain $\tilde{u}^{h,n+1}_{2i,2j,2k} \in \Omega^6_h$.

4: *Update every grid point in groups c, d, and e on* Ω_h.

For each $(odd, odd, even)$ grid point (x_i, y_j, z_k) in group c, the updated solution is approximated from Eq. (4.3) as

$$\begin{aligned}
\tilde{u}^{h,n+1}_{i,j,k} = [&F_{i,j,k} - A_{i,j,k}(1)\tilde{u}^{h,n}_{i+1,j,k} - A_{i,j,k}(2)\tilde{u}^{h,n}_{i,j,k+1} - A_{i,j,k}(3)\tilde{u}^{h,n}_{i-1,j,k} \\
&- A_{i,j,k}(4)\tilde{u}^{h,n}_{i,j,k-1} - A_{i,j,k}(5)\tilde{u}^{h,n}_{i,j-1,k} - A_{i,j,k}(6)\tilde{u}^{h,n}_{i,j+1,k} - A_{i,j,k}(7)\tilde{u}^{h,n}_{i+1,j,k+1} \\
&- A_{i,j,k}(8)\tilde{u}^{h,n}_{i-1,j,k+1} - A_{i,j,k}(9)\tilde{u}^{h,n}_{i-1,j,k-1} - A_{i,j,k}(10)\tilde{u}^{h,n}_{i+1,j,k-1} \\
&- A_{i,j,k}(11)\tilde{u}^{h,n+1}_{i+1,j-1,k} - A_{i,j,k}(12)\tilde{u}^{h,n}_{i,j-1,k+1} - A_{i,j,k}(13)\tilde{u}^{h,n+1}_{i-1,j-1,k} \\
&- A_{i,j,k}(14)\tilde{u}^{h,n}_{i,j-1,k-1} - A_{i,j,k}(15)\tilde{u}^{h,n+1}_{i+1,j+1,k} - A_{i,j,k}(16)\tilde{u}^{h,n}_{i,j+1,k+1} \\
&- A_{i,j,k}(17)\tilde{u}^{h,n+1}_{i-1,j+1,k} - A_{i,j,k}(18)\tilde{u}^{h,n}_{i,j+1,k-1}]/A_{i,j,k}(0).
\end{aligned}$$

Here, $F_{i,j,k}$ represents the right-hand side part of Eq. (4.3). The sixth order solutions for grid points in groups d and e are approximated similar to like those in group c. Each grid point in these three groups has 4 neighboring fine grid points in group a.

5: *Update every grid point in groups f, g, and h on* Ω_h.

For each $(odd, even, even)$ grid point (x_i, y_j, z_k) in group f the updated solution is computed as

$$\begin{aligned}\tilde{u}^{h,n+1}_{i,j,k} = [&F_{i,j,k} - A_{i,j,k}(1)\tilde{u}^{h,n+1}_{i+1,j,k} - A_{i,j,k}(2)\tilde{u}^{h,n+1}_{i,j,k+1} - A_{i,j,k}(3)\tilde{u}^{h,n+1}_{i-1,j,k} \\ &- A_{i,j,k}(4)\tilde{u}^{h,n+1}_{i,j,k-1} - A_{i,j,k}(5)\tilde{u}^{h,n+1}_{i,j-1,k} - A_{i,j,k}(6)\tilde{u}^{h,n+1}_{i,j+1,k} - A_{i,j,k}(7)\tilde{u}^{h,n}_{i+1,j,k+1} \\ &- A_{i,j,k}(8)\tilde{u}^{h,n}_{i-1,j,k+1} - A_{i,j,k}(9)\tilde{u}^{h,n}_{i-1,j,k-1} - A_{i,j,k}(10)\tilde{u}^{h,n}_{i+1,j,k-1} \\ &- A_{i,j,k}(11)\tilde{u}^{h,n}_{i+1,j-1,k} - A_{i,j,k}(12)\tilde{u}^{h,n}_{i,j-1,k+1} - A_{i,j,k}(13)\tilde{u}^{h,n}_{i-1,j-1,k} \\ &- A_{i,j,k}(14)\tilde{u}^{h,n}_{i,j-1,k-1} - A_{i,j,k}(15)\tilde{u}^{h,n}_{i+1,j+1,k} - A_{i,j,k}(16)\tilde{u}^{h,n}_{i,j+1,k+1} \\ &- A_{i,j,k}(17)\tilde{u}^{h,n}_{i-1,j+1,k} - A_{i,j,k}(18)\tilde{u}^{h,n}_{i,j+1,k-1}]/A_{i,j,k}(0).\end{aligned}$$

The sixth order solutions for grid points in groups g and h are approximated similarly like those in group f. Each grid point in these three groups has 2 neighboring fine grid points in group a.

6: *Update every grid point in group b on* Ω_h.

For each (odd, odd, odd) grid point (x_i, y_j, z_k) in group b the updated solution is computed as

$$\begin{aligned}\tilde{u}^{h,n+1}_{i,j,k} = [&F_{i,j,k} - A_{i,j,k}(1)\tilde{u}^{h,n+1}_{i+1,j,k} - A_{i,j,k}(2)\tilde{u}^{h,n+1}_{i,j,k+1} - A_{i,j,k}(3)\tilde{u}^{h,n+1}_{i-1,j,k} \\ &- A_{i,j,k}(4)\tilde{u}^{h,n+1}_{i,j,k-1} - A_{i,j,k}(5)\tilde{u}^{h,n+1}_{i,j-1,k} - A_{i,j,k}(6)\tilde{u}^{h,n+1}_{i,j+1,k} - A_{i,j,k}(7)\tilde{u}^{h,n+1}_{i+1,j,k+1} \\ &- A_{i,j,k}(8)\tilde{u}^{h,n+1}_{i-1,j,k+1} - A_{i,j,k}(9)\tilde{u}^{h,n+1}_{i-1,j,k-1} - A_{i,j,k}(10)\tilde{u}^{h,n+1}_{i+1,j,k-1} \\ &- A_{i,j,k}(11)\tilde{u}^{h,n+1}_{i+1,j-1,k} - A_{i,j,k}(12)\tilde{u}^{h,n+1}_{i,j-1,k+1} - A_{i,j,k}(13)\tilde{u}^{h,n+1}_{i-1,j-1,k} \\ &- A_{i,j,k}(14)\tilde{u}^{h,n+1}_{i,j-1,k-1} - A_{i,j,k}(15)\tilde{u}^{h,n+1}_{i+1,j+1,k} - A_{i,j,k}(16)\tilde{u}^{h,n+1}_{i,j+1,k+1} \\ &- A_{i,j,k}(17)\tilde{u}^{h,n+1}_{i-1,j+1,k} - A_{i,j,k}(18)\tilde{u}^{h,n+1}_{i,j+1,k-1}]/A_{i,j,k}(0).\end{aligned}$$

Each grid point in group b has no neighboring fine grid points in group a.

7: Compute the 2-norm $R = ||\tilde{u}^{h,n+1} - u^h_{old}||_2$. If R is bigger than a certain tolerance (10^{-10} in this dissertation), go back to Step 1.

In Algorithm 4.2.2, $A_{i,j,k}(l), l = 0, 1, ..., 18$, are the pre-computed coefficients for grid point (x_i, y_j, z_k). Ω_h^4 and Ω_{2h}^4 denote the fourth order accurate solution space from the FOC schemes, Ω_h^6 and Ω_{2h}^6 are the improved sixth order accurate solution space. $\tilde{u}^{h,n}$ is the approximate solution for the fine grid after the n iterations.

I update the fine grid points group by group based on the number of their neighboring grid points with sixth order solution (group a) from Step 2. Grid points in groups c, d and e have more qualified neighbors than those in other groups, so I update these three groups first. The iteration will continue until the 2-norm R of the correction vector is reduced below a certain tolerance.

4.3 Special Solution Strategies

For solving the discretized 3D convection-diffusion equation, multigrid techniques have been used extensively [27, 68, 78]. However, standard multigrid algorithms fail to achieve optimal grid independent convergence rates in solving non-elliptic problems. I used multiscale multigrid combined with plane relaxation, as illustrated in Fig. 4.4, to handle the non-elliptic problems.

4.3.1 Plane relaxation in multigrid method

Standard multigrid method with point Gauss-Seidel relaxation is known to be highly efficient in solving systems of elliptic partial differential equations, but it fails to achieve optimal grid independent convergence rate for some convection diffusion equations like the convection dominated problems with high Reynolds number and the Poisson equation that has anisotropic discrete operators [27, 35, 36, 72]. Like the line relaxation can handle the anisotropic and convection-dominated problems, plane relaxation is an efficient approach which can eliminate all high frequency errors in the presence of strong anisotropies for 3D problems. Other authors have also used plane implicit methods as multigrid smoothers with approximated solutions for the planes

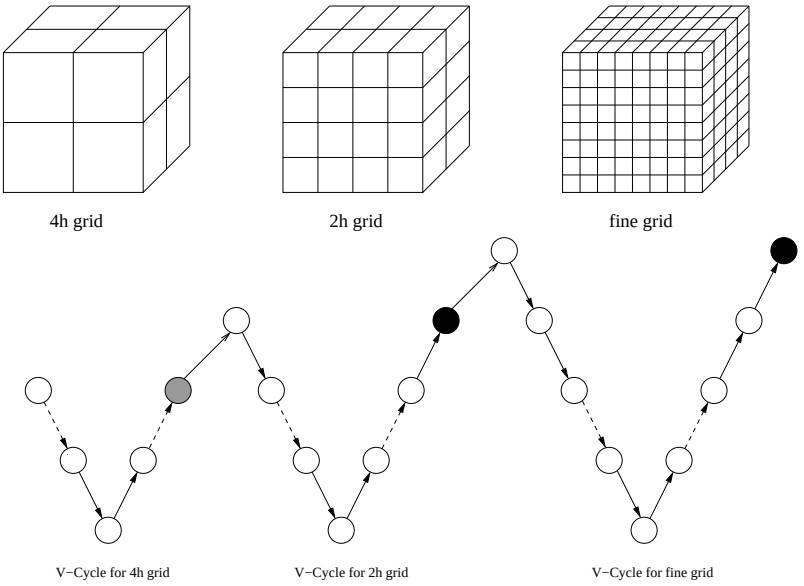


Figure 4.4: Representation of multiscale multigrid method for 3D convection diffusion equation.

[42, 62].

The alternating $(x-y-z)$ plane Gauss-Seidel relaxation in lexicographic order performs one sweep of (y,z)-plane Gauss-Seidel relaxation along the x-coordinate direction first, followed by one sweep of (x,z)-plane Gauss-Seidel relaxation and (x,y)-plane Gauss-Seidel relaxation along the y-coordinate direction and z-coordinate direction, respectively. For each direction, I divide the 3D problem into N 2D sub-problems as in Fig. 4.5, where N is the number of intervals along that direction. Let us first consider the (x,z)-plane 2D sub-problem, its nine point computational stencil can be generated from the Eq. (4.3) as

$$\begin{aligned} &A_0\tilde{u}_0 + A_1\tilde{u}_1 + A_2\tilde{u}_2 + A_3\tilde{u}_3 + A_4\tilde{u}_4 + A_5\tilde{u}_5 + A_6\tilde{u}_6 + A_7\tilde{u}_7 + A_8\tilde{u}_8 \\ =&F_0 - \alpha_5 u_5 - \alpha_6 u_6 - \alpha_{11} u_{11} - \alpha_{12} u_{12} - \alpha_{13} u_{13} - \alpha_{14} u_{14} - \alpha_{15} u_{15} \\ &- \alpha_{16} u_{16} - \alpha_{17} u_{17} - \alpha_{18} u_{18}, \end{aligned} \tag{4.6}$$

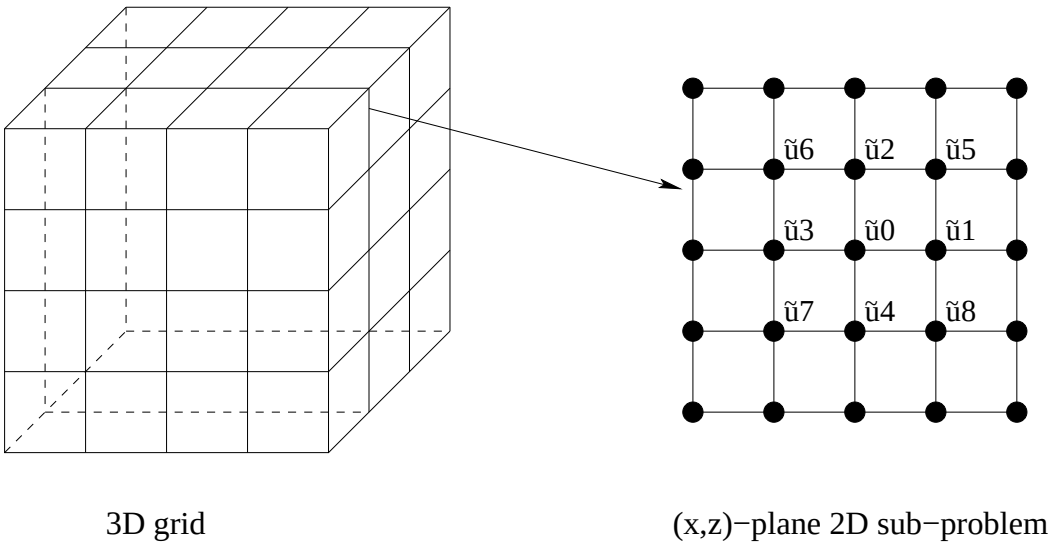


Figure 4.5: 2D sub-problem in plane relaxation.

where the coefficients A_l and the 2D solution $\tilde{u}_l$ $(l = 0, 1, ..., 8)$ are set as

$$A_l = \begin{cases} \alpha_l & 0 \le l \le 4, \\ \alpha_{l+2} & 5 \le l \le 8, \end{cases} \qquad \tilde{u}_l = \begin{cases} u_l & 0 \le l \le 4, \\ u_{l+2} & 5 \le l \le 8. \end{cases}$$

The nine point computational stencil for the (x, y) and (y, z)-planes can be generated like Eq. (4.6) similarly, but the index of the 3D grid points for those corresponding 2D grid points are different.

I note that the grids visited in the 2D planes are different from the grids used for the 3D multigrid. Therefore, the coefficient matrices for solving the planes does not correspond to the coefficient matrices for 3D grid hierarchy. If I choose to pre-compute all the coefficient matrices for every 2D sub-problems, the memory requirements would be significantly increased. According to [35], to pre-compute all the 2D coefficient matrices for plane smoothers, the memory cost is 52% higher than that of the point relaxation scheme.

Due to the improvement of supercomputer, in many large simulation and modeling applications, the solution of PDEs is moving from time-critical to accuracy-critical

[35]. In these applications, memory usage is becoming the most important limiting factor to solve large problems. Therefore, the inner 2D coefficient matrix is computed each time the plane is visited.

Inner 2D solver. The plane relaxation is considered in the multigrid literature to have poor numerical and parallel properties because it needs to solve a large number of 2D sub-problems. However, it is shown in [35] that an exact solution of the 2D sub-problems for planes is not needed and that one multigrid cycle is sufficient if I use multigrid method as the inner 2D solver. This behavior has also been reported by other researchers in [42, 62].

I would like to mention here that if I use the geometric multigrid method as the inner 2D solver to solve the 3D convection diffusion equation with variable coefficients, for some 2D planes, we are not able to compute the full coefficient matrix for their coarse grids. For example in Fig. 4.6, which is a $5 \times 5 \times 5$ 3D grid. The 19 black color grid points are the coarse grid points that are needed for plane $y = 2$ to compute its nine point computational stencil by Eq. (4.6). In comparison, the 19 gray color grid points are the coarse grid points needed for plane $y = 1$. We note that, plane $y = 1$ needs some coarse grid points from plane $y = -1$, which is not in the 3D cube, to generate its coarse grid coefficient matrix for solving the 2D sub-problem. Due to these drawbacks, I only use the multigrid method as the inner 2D solver in plane relaxation for solving the 3D convection diffusion equation with constant coefficients. If I rewrite the Eq. (4.3) by using constant coefficients p, q, and r, the coefficients for each plane along the same direction are the same. For plane $y = 1$ in Fig. 4.6, I do not need the additional information from plane $y = -1$, because every plane along the y direction has the same coarse grid coefficients.

In order to keep the optimal convergence rate, for the inner 2D multigrid solver, we use alternating line relaxation smoother in the 2D V-Cycle because the 2D sub-

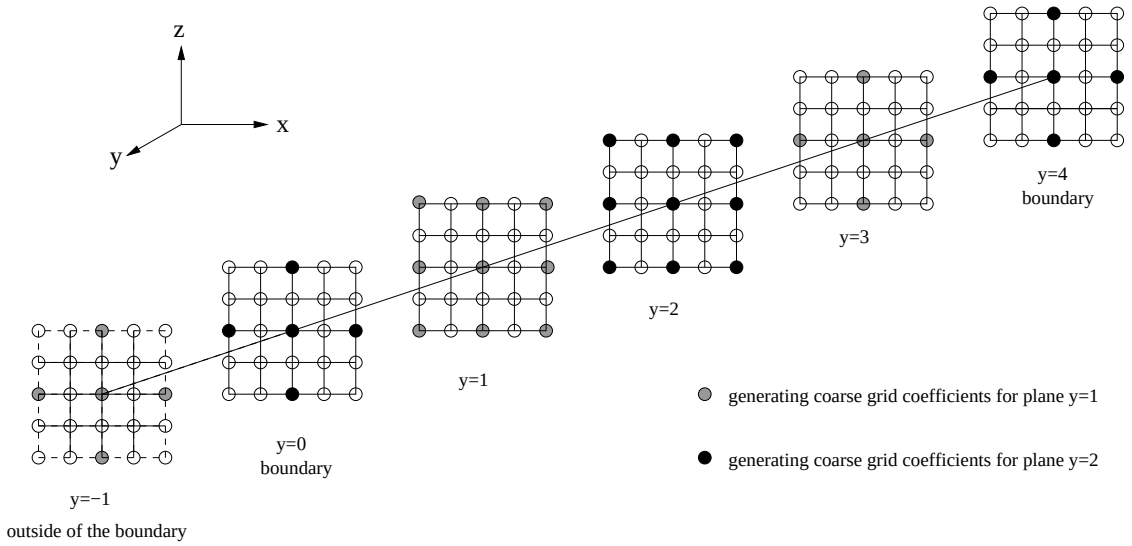


Figure 4.6: coarse grid of the 2D sub-problem in plane relaxation.

problem may still have the high convection operator or the anisotropy.

4.4 Numerical Results

The domain Ω for the two test problems I solved was chosen as the unit cube $(0,1)^3$. We tested two problems with both constant and variable coefficients. We chose Problem 1 with variable coefficients, so we could not use the plane relaxation with inner multigrid 2D solver. I verified that it was sufficient to use the standard point Gauss-Seidel relaxation smoother to solve the 3D convection diffusion equation efficiently with relatively small Reynolds number and its efficiency was degraded when the Reynolds number increased. For Problem 2 with constant coefficients, I tested the plane Gauss-Seidel relaxation smoother with some large Reynolds number and compared the results with the point relaxation smoother.

I used standard V(1,1) cycle in the multiscale multigrid method. The initial guess for the V-Cycle on the $4h$ grid was the zero vector. The stopping criteria for the operator based interpolation and the V-Cycle on $2h$ and h grids were 10^{-10}. The errors reported were the maximum absolute errors over the discrete grid of the finest

level. For the SOC method, the number of iterations contains three parts. They are the number of V-Cycles for Ω_{2h}, the number of V-Cycles for Ω_h, and the number of iterations for the iterative interpolation combined with the Richardson extrapolation.

In all tables, the column titles # iter refers to the number of iterations; CPU refers to the CPU cost in seconds; error refers to the maximum absolute error; order refers to the order of accuracy for the computed solution.

4.4.1 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 and simulates the Reynolds number in a flow simulation. The Dirichlet boundary conditions and the forcing term f are set to satisfy the exact solution.

I tested the first problem using point relaxation smoother for small to relatively large values of the Reynolds number ($Re \leq 10^5$). The numerical results are listed from Table 4.1 to Table 4.3.

Table 4.1 contains the numerical results for Problem 1 when $Re = 0$, which reduces it to the 3D Poisson equation. I noted that, with point relaxation smoother, both the SOC scheme and the FOC scheme solved the Problem 1 with optimal grid independent convergence rate. The order of the computed solution from the SOC scheme was close to 6 as I expected and was higher than that from the FOC scheme. In terms of computational cost with the same mesh size h, the FOC scheme was faster

Table 4.1: Maximum errors, CPU seconds and the number of iterations of the FOC and SOC schemes for Problem 1 with $Re = 0$.

	FOC Point				SOC Point			
h	# iter	CPU	error	order	# iter	CPU	error	order
1/8	11	0.004	2.04e-3	4.2	(8,11),33	0.005	1.55e-3	5.0
1/16	12	0.031	1.09e-4	4.1	(11,12),42	0.051	4.90e-5	5.4
1/32	12	0.281	6.29e-6	4.0	(12,12),44	0.617	1.15e-6	5.8
1/64	12	2.412	3.76e-7	4.0	(12,11),43	6.234	2.14e-8	5.8

Table 4.2: Maximum errors, CPU seconds and the number of iterations of the FOC and SOC schemes for Problem 1 with $Re = 10$.

	FOC Point				SOC Point			
h	# iter	CPU	error	order	# iter	CPU	error	order
1/8	12	0.004	2.55e-3	4.0	(9,12),35	0.006	1.95e-3	5.0
1/16	13	0.032	1.41e-4	4.1	(12,13),46	0.056	6.13e-5	5.5
1/32	13	0.291	8.18e-6	4.1	(13,12),47	0.637	1.40e-6	5.8
1/64	12	2.397	4.90e-7	4.1	(12,12),44	6.531	2.56e-8	5.7

because it only ran a standard multigrid V-Cycle and the SOC scheme needed to run the multiscale multigrid method and the operator based interpolation scheme. For the computed solution accuracy, the SOC scheme was more accurate than the FOC scheme for every meshsize I tested, this can be seen from Table 4.1.

When I chose $Re = 10$, it was clear from Table 4.2 that our SOC scheme still yielded a sixth order solution accuracy for small Reynolds number though there was a slight increase in the number of iterations as Re was increased and our multiscale multigrid method still kept the grid independent convergence rate. Similar behavior was observed for the FOC scheme.

The numerical results in Table 4.1 and Table 4.2 indicate that using point relaxation smoother was sufficient to solve the 3D convection diffusion equation with relatively small Reynolds numbers. However, the data in Table 4.3 shows that when the magnitude of the Reynolds number was large enough ($Re = 10^5$), the iterative convergence and the computed accuracy were severely degraded. The solution meth-

Table 4.3: Maximum errors, CPU seconds and the number of iterations of the FOC and SOC schemes for Problem 1 with $Re = 10^5$.

	FOC Point				SOC Point			
h	# iter	CPU	error	order	# iter	CPU	error	order
1/8	51	0.009	8.40e-2	2.0	(18,53),67	0.014	7.24e-2	2.7
1/16	158	0.262	1.97e-2	2.1	(53,149),139	0.232	1.18e-2	2.9
1/32	521	7.847	4.77e-3	2.0	(150,505),276	7.463	1.57e-3	3.0
1/64	not converge	–	–	–	not converge	–	–	–

ods for both the FOC and the SOC schemes did not obtain the grid independent convergence. They took hundreds of multigrid cycles to converge when $h \leq 1/32$ and did not converge within the maximum number of iterations (1000) I set when $h = 1/64$.

Table 4.4: Comparison of the number of iterations for different approximation strategy in operator based interpolation scheme for Problem 1.

	Re=10		$Re = 100$	
h	Grouping	Lexicographic	Grouping	Lexicographic
1/8	35	36	26	29
1/16	46	49	36	50
1/32	47	50	53	68
1/64	44	47	52	64

Grouping strategy. The operator based interpolation for the 3D convection-diffusion equation is an iterative method and the grid points are updated group by group as indicated in Algorithm 4.2.2. The reason that I update the grid points by their number of neighboring points is that it can reduce the number of iterations compared with the updating in lexicographical order. Supporting numerical results are shown in Table 4.4, Fig. 4.7 and 4.8.

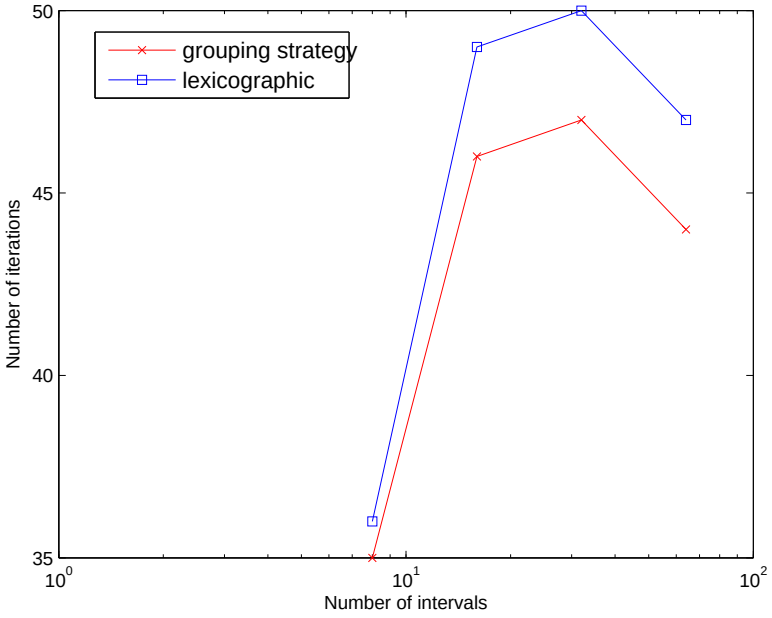


Figure 4.7: Comparison of the number of iterations of the operator based interpolation scheme with different updating strategies for Problem 1 ($Re = 10$).

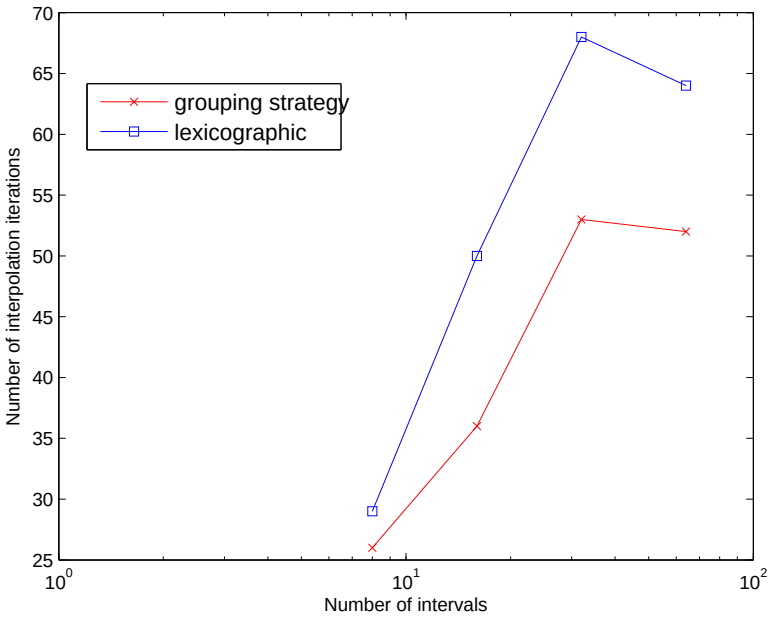


Figure 4.8: Comparison of the number of iterations of the operator based interpolation scheme with different updating strategies for Problem 1 ($Re = 100$).

4.4.2 Test problem 2

For test problem 2, I chose the constant coefficients with large Reynolds number as

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

I used both the point and the plane relaxation smoothers in the multiscale multigrid method to solve this problem.

I tested this problem with a very large Reynolds number ($Re = 10^7$) and the reported numerical results were listed in Table 4.5. I noted that the accuracy of the solutions computed by the multigrid method using plane relaxation smoother and point relaxation smoother was comparable. For the CPU cost with the same meshsize, the point relaxation method was faster than the plane relaxation method when $h \leq 1/16$, that was because the plane relaxation needed to run a large number of 2D V-Cycles. When the mesh became finer, the number of iterations of the point relaxation method increased very quickly and it did not converge when $h = 1/64$. On the other hand, the number of iterations of the plane relaxation method was almost stable with respect to the meshsize. For better understanding, I listed the graphic comparison of the number of iterations for different relaxation smoothers in Fig. 4.9. It was clear that the multigrid method using the plane relaxation smoother took much less iterations than that from point relaxation smoother and the grid independent convergence was kept when the mesh became finer.

Table 4.6 contains the test results with various Re and fixed meshsize $h = 1/32$. Computations were reported for the FOC and the SOC schemes using the plane relaxation smoother. I note that the magnitude of the Reynolds number (Re) affected the convergence and the computed solution accuracy of both schemes inversely. The SOC scheme yielded better solution accuracy than the FOC scheme with every Re

Table 4.5: Maximum errors, CPU seconds and the number of iterations of Problem 2 using plane relaxation and point relaxation smoothers with $Re = 10^7$.

		Point relaxation				Plane relaxation			
h	strategy	# iter	CPU	error	order	# iter	CPU	error	order
1/8	FOC	52	0.009	6.10e-2	2.1	8	0.017	6.10e-2	2.1
	SOC	(16,51),55	0.010	5.18e-2	2.7	(4,8),55	0.019	5.18e-2	2.7
1/16	FOC	164	0.262	1.41e-2	2.1	17	0.320	1.41e-2	2.1
	SOC	(51,164),74	0.294	7.96e-3	3.0	(10,16),74	0.389	7.96e-3	3.0
1/32	FOC	500	7.621	3.36e-3	2.0	26	4.852	3.36e-3	2.0
	SOC	(164,472),79	7.808	1.01e-3	3.3	(16,26),79	3.247	1.01e-3	3.3
1/64	FOC	not converge	–	–	–	46	70.073	8.21e-4	2.0
	SOC	not converge	–	–	–	(26,45),81	80.787	1.06e-4	3.2

value we tested. When $Re > 10^4$, there was only a little change for the solution accuracy and the number of iterations for both the FOC and the SOC schemes. I believed that the convergence rate and the number of iterations approached some limits and did not deteriorate any more when Re was beyond certain large values.

Table 4.6: Maximum errors, CPU seconds and the number of iterations of Problem 2 using plane smoother with different Re.

	FOC plane relaxation			SOC plane relaxation		
$Re(h = 1/32)$	# iter	CPU	error	# iter	CPU	error
0	6	1.062	6.29e-6	(6,6),45	1.482	1.15e-6
1	7	1.234	3.48e-5	(6,6),45	1.693	1.24e-6
10^2	12	2.109	4.84e-4	(10,12),48	2.925	2.87e-4
10^4	28	4.854	3.36e-3	(16,26),79	5.337	1.01e-3
10^5	26	4.518	3.36e-3	(16,26),79	5.349	1.01e-3
10^6	26	4.489	3.36e-3	(16,26),79	5.335	1.01e-3
10^7	26	4.455	3.36e-3	(16,26),79	5.367	1.01e-3
10^{10}	26	4.492	3.36e-3	(16,26),79	5.402	1.01e-3

4.5 Concluding Remarks

I extended the sixth order compact finite difference scheme for solving the 2D convection diffusion equations [68] to 3D problems. Our numerical results indicated that the SOC scheme is more accurate than the FOC scheme to produce solution.

The numerical results also indicated that our operator based interpolation using

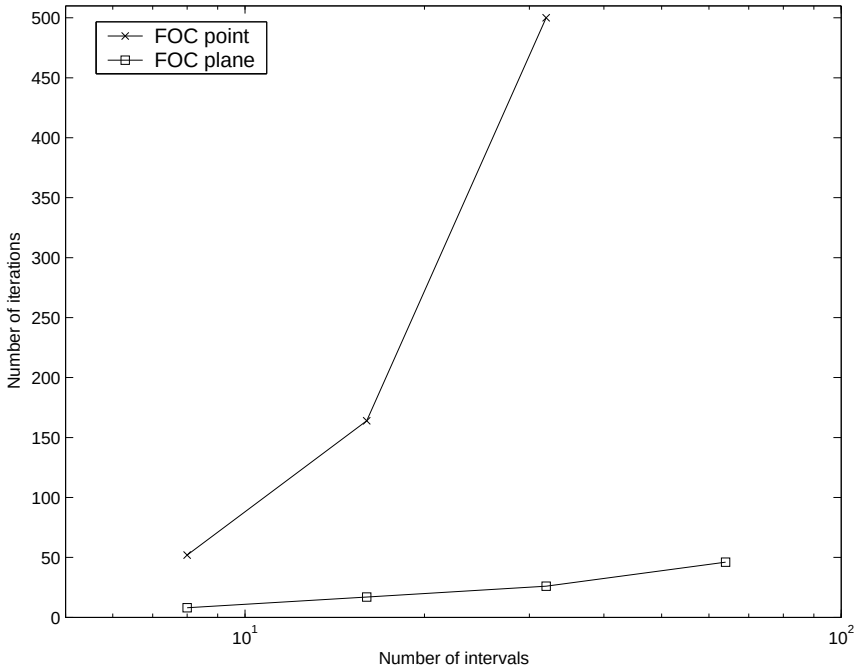


Figure 4.9: Comparison of the number of iterations of the FOC scheme for solving Problem 2 using plane relaxation and point relaxation($Re = 10^7$). Each symbol corresponds to an increasing fine grid: 8, 16, 32, and 64 intervals.

the grouping strategy takes fewer number of iterations than that from lexicographical order. In order to keep the grid independent convergence of our multiscale multigrid method for large Reynolds number problems, we used plane relaxation as our smoother in multigrid. The numerical results showed that the point relaxation method is efficient when Re is small and I recommend the user to use plane relaxation when the point relaxation cannot keep the good convergence rate.

Chapter 5

Multiple Coarse Grid Computation and Special W-cycle Multiscale Multigrid Method

From the above chapters, we developed an efficient computational framework to solve the PDEs like the Poisson equation and convection-diffusion equation with high accuracy and high efficiency. The computational framework is based on two methods: the fourth order multiscale multigrid method and an operator based interpolation scheme combined with extrapolation technique.

However, these two methods have their own disadvantages. The multiscale multigrid method that we used in previous chapters need to run three independent fourth order multigrid V-cycle procedures with different meshsize to achieve the multiscale grids approximation. We believe this method does not have optimal computational efficiency because the standard single V-cycle or W-cycle multigrid on Ω_h already has the multilevel grid hierarchy. We want to develop an algorithm like special relaxation or interpolation that can approximate the fourth order solutions on different level and run the extrapolation using standard V-cycle or W-cycle.

For the operator based interpolation technique, if the coefficient matrix A is not diagonally dominant like the convection-diffusion equation with very large cell Reynolds number, it may take a huge number of iterations to converge. In this chapter, I present another technique called the multiple coarse grid computation technique. This approach can be used to get the coarse grid solutions for every corresponding fine grid, which means the I can directly apply Richardson extrapolation for every grid points

and no operator based interpolation is needed.

5.1 Multiple Coarse Grid Computation

As I already mentioned in Chapter 1, one of our motivations is to build the efficient and scalable method for solving linear systems arising from higher order discretization scheme of PDEs that have the potential to be modified to work on the parallel computer. The operator based interpolation scheme is an iterative method which is not very easy to be changed to its parallel version.

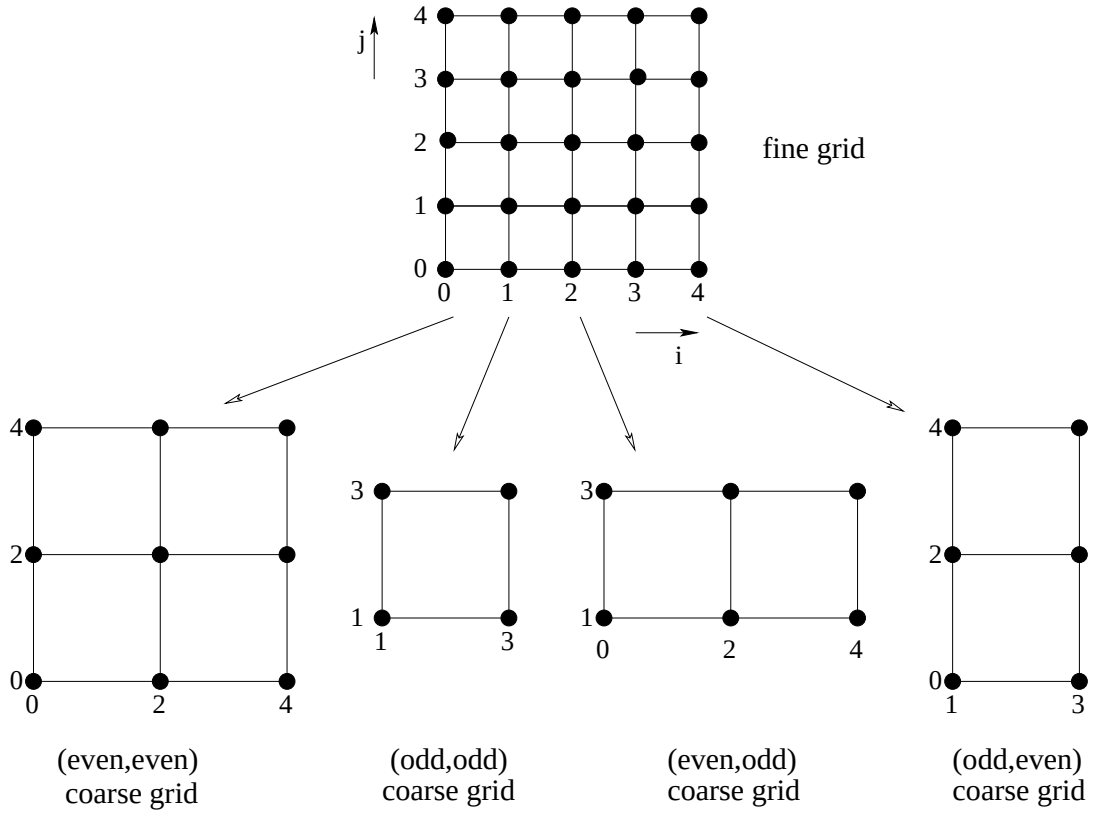


Figure 5.1: Illustration of the multiple coarse grid for a 5×5 fine grid.

The idea of using the multiple coarse grid is from the parallel superconvergent multigrid method. In addition to splitting the original grid and filtering residual vector to exploit parallelism, one can use the concurrent relaxation method on multiple grids [83]. The basic idea of multigrid superconvergent is to use multiple coarse grid

to generate better correction for the fine grid solution than that from the single coarse grids. The reason is that for general multigrid method of 2D problem, the residual of the fine grid is projected to only $(even, even)$ coarse grid. But I can also project the residual to other coarse grids. Therefore, a combination of error correction from all the coarse grid may make the fine grid converge faster than that from a single coarse grid. In general, for a d dimensional problem, the fine grid can be easily coarsened into 2^d coarse grids, like the 2D case in Fig. 5.1. If the computation work for each coarse grid can be loaded to a separate processor, it does not require more time than what is need for a single coarse grid.

The multiple coarse grid idea is used for parallel multigrid method to speed up the convergence [83], but it has never been used to increase the order of accuracy for the fine grid. The operator based interpolation scheme I developed has a drawback that it only has the fourth order coarse grid solution for the $(even, even)$ coarse grid. After the Richardson extrapolation for the $(even, even)$ grid, other grid points need to be iteratively approximated, but the cost is the loss of parallelism and the additional computation time. If we can generate four FOC solution $u^{2h}_{(e,e)}$, $u^{2h}_{(o,o)}$, $u^{2h}_{(e,o)}$, and $u^{2h}_{(o,e)}$ on coarse grid $\Omega^{(e,e)}_{2h}$, $\Omega^{(o,o)}_{2h}$, $\Omega^{(e,o)}_{2h}$, and $\Omega^{(o,e)}_{2h}$, respectively, then these four FOC solutions can cover every grid points in the fine grid Ω_h. The standard Richardson extrapolation can be used for every grid point on the fine grid. Although we need some additional work on the coarse grids, we can avoid the interpolation iteration on the fine grid to reduce the CPU cost.

5.1.1 1D multiple coarse grid

Let's start with 1D multiple coarse grid computation for easy understanding. For the fine grid Ω_h, we construct two coarse grids in such a way that all the even-numbered grid points belong to coarse grid Ω_{even} and all the odd-numbered grid points belong

to coarse grid Ω_{odd} as shown in Fig. 5.2. Then we have

$$\Omega_{even} = \{x_j | x_j \in \Omega_h \ and \ (j = even)\},$$

$$\Omega_{odd} = \{x_j | x_j \in \Omega_h \ and \ (j = odd)\}.$$

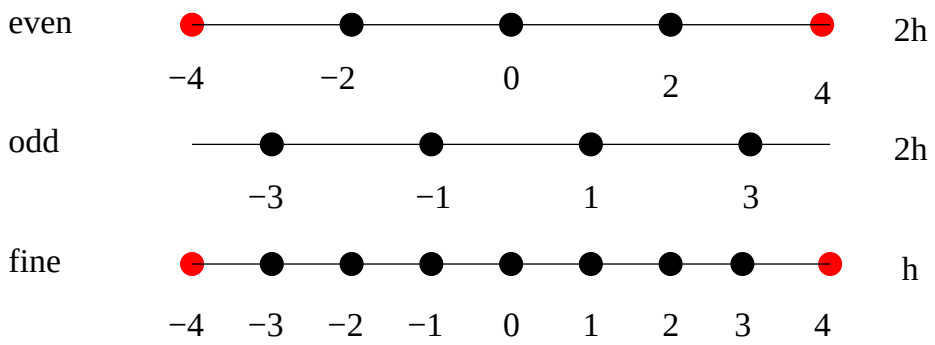


Figure 5.2: Illustration of the multiple coarse grid for 1D problem.

From Fig. 5.2, we note that the even indexed coarse grid is easy to be solved by double the mesh size from h to $2h$. However, the coarse grid Ω_{odd} only contains the *black* color grid points from fine grid but no *red* color boundary grid points. We could not develop the finite difference schemes for coarse grid Ω_{odd} if we only have the inner grid points. One possible approach is to add these red color boundaries to Ω_{odd} and develop special computational stencil for grid point u_{-3} and u_3 as shown in Fig. 5.3.

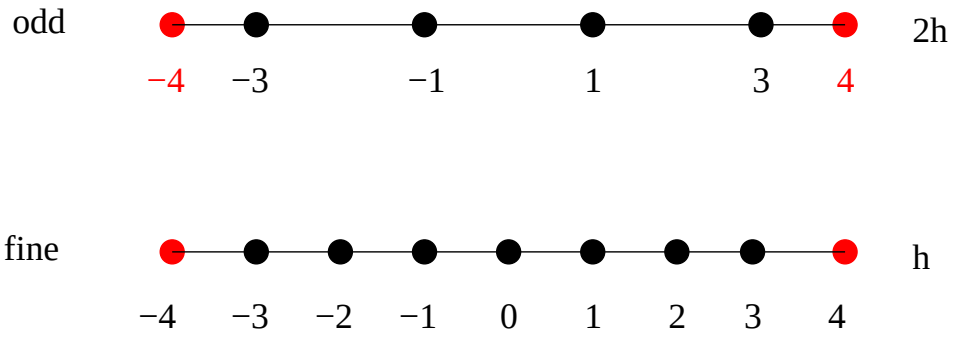


Figure 5.3: Ω_{odd} with two added red color boundary grid points.

For the 1D problem in Fig. 5.3, the computational stencil for the grid points near

the boundaries are different with other inner grid points. For the inner grid points like u_{-1} and u_1, their finite difference schemes are based on $2h$ meshsize. However, if we take grid point u_{-3} in Ω_{odd} as an example, its compact finite difference scheme needs the boundary grid point u_{-4} and inner grid point u_{-1}. Its meshsize between u_{-4} and u_{-1} are h and $2h$.

Lemma 5.1.1. *For coarse grid as shown in Fig. 5.3, the solution accuracy for central difference operator becomes first order.*

Proof. We denote x_j to be the near boundary grid points on left side and $u_j = u(x_j)$. By using Taylor series expansion we have

$$\begin{aligned}
u(x_j + h) &= u(x_j) + h(u_x)_j + h^2\frac{(u_{xx})_j}{2!} + h^3\frac{(u_{x^3})_j}{3!} + ...\\
u(x_j - h) &= u(x_j) - h(u_x)_j + h^2\frac{(u_{xx})_j}{2!} - h^3\frac{(u_{x^3})_j}{3!} + ...\\
u(x_j + 2h) &= u(x_j) + 2h(u_x)_j + 4h^2\frac{(u_{xx})_j}{2!} + 8h^3\frac{(u_{x^3})_j}{3!} + ...\\
u(x_j - 2h) &= u(x_j) - 2h(u_x)_j + 4h^2\frac{(u_{xx})_j}{2!} - 8h^3\frac{(u_{x^3})_j}{3!} + ...
\end{aligned}$$

To approximate the first derivative, then we apply

$$\begin{aligned}
&u(x_j + 2h) - 4u(x_j - h) + 3u(x_j)\\
=&6h(u_x)_j + O(h^3).
\end{aligned}$$

So,

$$\delta_x u_j = \frac{u(x_j + 2h) - 4u(x_j - h) + 3u(x_j)}{6h} + O(h^2).$$

For the second derivative, if we apply central difference operator, it will become

$$\begin{aligned}
&u(x_j + 2h) + 2u(x_j - h) - 3u(x_j)\\
=&3h^2(u_{xx})_j + O(h^3).
\end{aligned}$$

Then we have

$$\delta_{xx}u_j = \frac{u(x_j + 2h) + 2u(x_j - h) - 3u(x_j)}{3h^2} + O(h).$$

■

Since the second order central difference operator is degraded to first order, the FOC scheme which is based on the approximation for the second order terms will be degraded to second order for these near boundary grid points. In order to achieve fourth order solutions for every coarse grid points, we add two more grid points to the Ω_{odd} like the *blue* color grid points in Fig. 5.4.

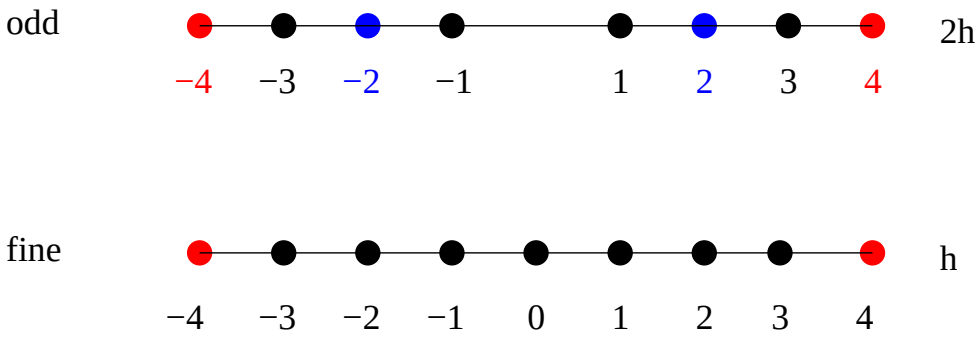


Figure 5.4: Ω_{odd} with two red color boundary and two blue color inner grid points .

By adding these four grid points, we can discretize every grid points in Ω_{odd} with fourth order accuracy using FOC scheme I introduced in Chapters 2 and 3. Let's assume the Ω_{odd} contains Nx grid points $u_{odd}(0), u_{odd}(1), \ \ldots \ , u_{odd}(Nx)$. Then the Ω_{even} will contains $Nx-3$ grid points and fine grid will contains $2Nx-7$ grid points. The grid points on Ω_{odd} are approximated as follows:

- For $j \in \{1, 2, Nx-2, Nx-1\}$, $u_{odd}(j)$ is approximated by three-point computational stencil from FOC scheme using grid points $u_{odd}(j-1)$ and $u_{odd}(j+1)$ with meshsize h. The truncation error is $O(h^4)$.

- For $j = 3$, $u_{odd}(j)$ is approximated by three-point computational stencil from FOC scheme using grid points $u_{odd}(j-2)$ and $u_{odd}(j+1)$ with meshsize $2h$. The truncation error is $O((2h)^4)$.

- For $j \in [4, Nx-4]$, $u_{odd}(j)$ is approximated by three-point computational stencil from FOC scheme using grid points $u_{odd}(j-1)$ and $u_{odd}(j+1)$ with meshsize $2h$. The truncation error is $O((2h)^4)$.

- For $j = Nx-3$, $u_{odd}(j)$ is approximated by three-point computational stencil from FOC scheme using grid points $u_{odd}(j-1)$ and $u_{odd}(j+2)$ with meshsize $2h$. The truncation error is $O((2h)^4)$.

By using above discretization and relaxation strategies, we can approximate fourth order solutions for every grid points on Ω_{odd}. After we get fourth order solutions for the fine grid and two coarse grids, each grid point on the fine grid has a corresponding grid point on either Ω_{even} or Ω_{odd}. Then we apply Richardson extrapolations for every fine grid points to approximate the sixth order solutions.

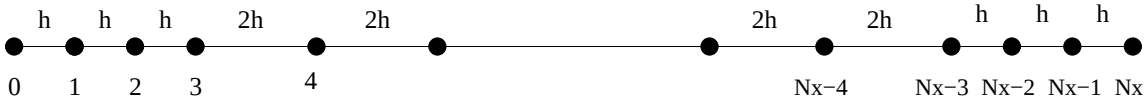


Figure 5.5: Representation of modified Ω_{odd} for 1D problem.

Numerical Results

Let's consider an example from Sun's previous work [59], the 1D convection-diffusion equation we tested is

$$\frac{\partial^2 u}{\partial x^2} - \frac{\partial u}{\partial x} - u = -\cos x - 2\sin x, \qquad 0 \le x \le \pi. \tag{5.1}$$

Eq. (5.1) has the Dirichlet boundary conditions as $u(0) = u(\pi) = 0$. The analytic solution for this problem is $u(x) = \sin x$.

We compared the truncated error and the order of accuracy by using multiple coarse grid computation technique and the Algorithm 2 I introduced in Chapter 2. The computational results are listed in Table 5.1 and Fig. 5.6.

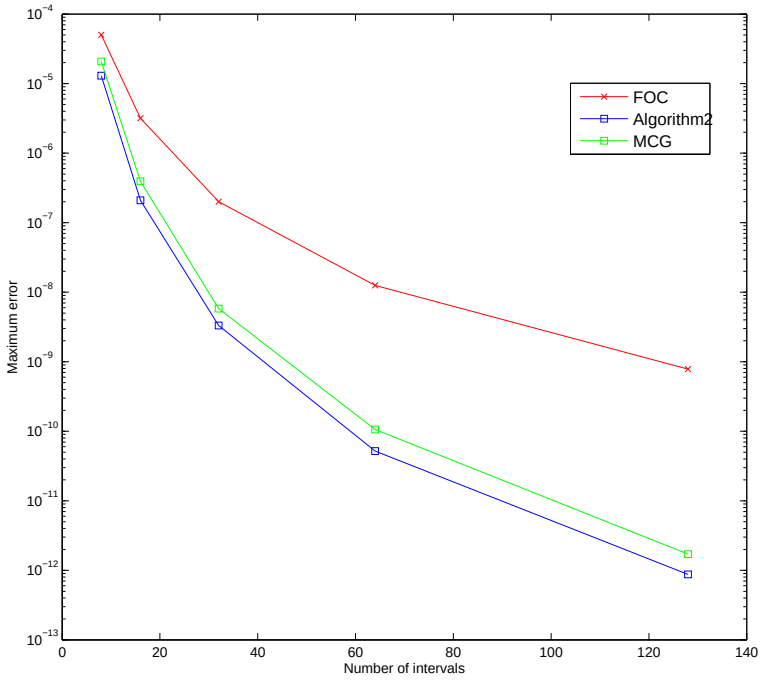


Figure 5.6: Comparison of maximum errors of FOC, Algorithm 2 and MCG methods.

From Table 5.1, we can see that the multiple coarse grid method (MCG) is more accurate than the fourth order scheme (FOC). Although the MCG method is not as accurate as the Algorithm 2 but it can achieve the sixth order solution accuracy when the number of intervals is bigger than 8. The reason why MCG is less accurate than Algorithm 2 is that there are two near boundary grid point using meshsize h to approximate instead of $2h$ in Ω_{odd}.

5.1.2 2D multiple coarse grid

For 2D problem, the approximation of four coarse grids is more difficult than 1D problem. I introduce two strategies to generate the multiple coarse grid solutions.

Table 5.1: Comparison of maximum errors and the order of accuracy by using the multiple coarse grid technique and Algorithm 2 for Eq. (5.1).

	FOC		Algorithm2		MCG	
h	Error	Order	Error	Order	Error	Order
$\pi/8$	5.02e-5	4.0	1.30e-5	5.9	2.08e-5	5.7
$\pi/16$	3.18e-6	4.0	2.10e-7	6.0	3.94e-7	6.1
$\pi/32$	2.00e-7	4.0	3.32e-9	6.0	5.81e-9	5.8
$\pi/64$	1.25e-8	4.1	5.20e-11	6.0	1.06e-10	6.0
$\pi/128$	7.83e-10	4.1	8.73e-13	6.0	1.71e-12	6.0

The first one is based on the idea of mesh-refinement technique by Hyman [29], which used grid rotation to approximate the fourth order solutions for four coarse grid using standard $(even, even)$ coarse grid as initial guess. It is very efficient to work on a single processor, but do not have much potential for parallelism. The other strategy is to let these four coarse grid solution generated on their own coarse grid, which is an extension from our 1D MCG method and can bring us more potential for parallelism.

Strategy 1. Suppose we already have the FOC fine grid solution u^h and coarse grid solution $u^{2h}_{(e,e)}$. For other three coarse grids, we only consider the inner grid points, as in Fig. 5.7. The black color grid points indicate the $(even, even)$ coarse grid solution $u^{2h}_{(e,e)}$, the numbers 1 and 2 mean the (odd, odd) and other coarse grid points, respectively. The grid points labeled with 1 will be interpolated before the grid points labeled with 2. We set $i, j = 0, 1, 2, ..., N$ on the evenly spaced $2N \times 2N$ fine grid. And we use converged u^h to project to every coarse grid point as our initial guess for the coarse grid solution, except the $(even, even)$ coarse grid.

The fine grid point (x_{2i+1}, y_{2j+1}) is also the (odd, odd) coarse grid. For the indexing on the coarse grid, we use the same indexing as in the fine grid for better understanding as shown in Fig. 5.1. By using Eq.(2.16) the interpolation for points

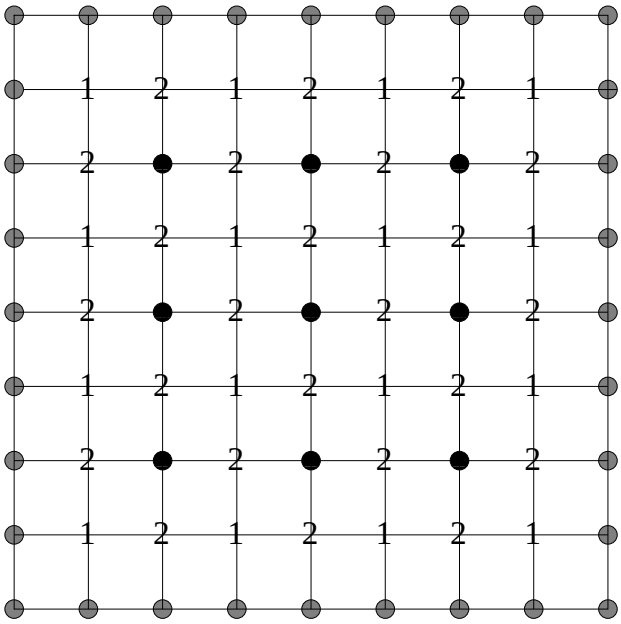


Figure 5.7: Illustration of how to generate multiple coarse grid solutions by using Strategy 1 for a 8×8 fine grid.

in group 1 is

$$
\begin{aligned}
& u^{2h}_{(o,o)}(2i+1, 2j+1) \\
& = \frac{1}{a}[F_{2i+1,2j+1} - b(u^{2h}_{(e,o)}(2i+2, 2j+1) + u^{2h}_{(e,o)}(2i, 2j+1) \\
& - c(u^{2h}_{(o,e)}(2i+1, 2j+2) + u^{2h}_{(o,e)}(2i+1, 2j)) - d(u^{2h}_{(e,e)}(2i+2, 2j+2) \\
& + u^{2h}_{(e,e)}(2i+2, 2j) + u^{2h}_{(e,e)}(2i, 2j+2) + u^{2h}_{(e,e)}(2i, 2j))].
\end{aligned}
$$

The interpolation for the $(even, odd)$ and $(odd, even)$ grid points are similar to the (odd, odd) grid points, it also uses its 8 fine grid neighbors. The procedure will continue until the correction vector below a certain tolerance.

Strategy 2. We project the fourth order fine grid solution u^h to the (odd, odd), $(even, odd)$, and $(odd, even)$ coarse grid as our initial guess. Let's first consider how to approximate the fourth order solution $u^{2h}_{(o,o)}$ on its own coarse grid.

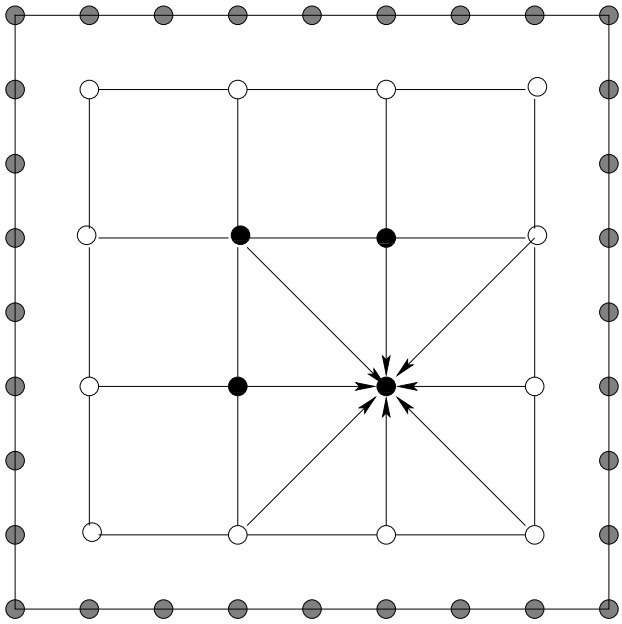


Figure 5.8: Illustration of how to generate (odd, odd) coarse grid solution by using Strategy 2

If the size of the fine grid is $(2N+1) \times (2N+1)$, then the size of $\Omega_{2h}^{(o,o)}$ is $N \times N$. A 8×8 example is shown in Fig. 5.8, where the gray colored points are the boundary grid points on the fine grid, the white colored grid points are the boundary of the (odd, odd) coarse grid and the black colored grid points are the inner grid points of the (odd, odd) coarse grid. In each relaxation step, we use all of its neighbor grid points from the (odd, odd) grid. But different with fine grid relaxation, we do not have the exact solution of boundary values of the (odd, odd) coarse grid. The most close value we have are their fourth order solution from u^h. So, if we just use the standard relaxation method for the inner grid points without updating the boundary values, we can not get enough accuracy. We need find a method to update the boundary values.

In each relaxation step, our treatment can be summarized as following: For the

inner coarse grid x_{2i+1}, y_{2j+1}

$$
\begin{aligned}
&u^{2h}_{(o,o)}(2i+1,2j+1)\\
&=\frac{1}{a}[F_{2i+1,2j+1}-b(u^{2h}_{(e,o)}(2i+3,2j+1)+u^{2h}_{(e,o)}(2i-1,2j+1)\\
&-c(u^{2h}_{(o,e)}(2i+1,2j+3)+u^{2h}_{(o,e)}(2i+1,2j-1))-d(u^{2h}_{(e,e)}(2i+3,2j+3)\\
&+u^{2h}_{(e,e)}(2i+3,2j-1)+u^{2h}_{(e,e)}(2i-1,2j+3)+u^{2h}_{(e,e)}(2i-1,2j-1))].
\end{aligned}
$$

For the boundary grid point, we can use one of its neighbors (who has all the 8 neighbors on the coarse grid) to use the 9 point stencil to do relaxation or we can use its 8 neighbors in the fine grid.

Numerical results

We consider a Poisson equation in the form of

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=-104\pi^2\sin(10\pi x)\cos(2\pi y),\qquad (x,y)\in\Omega=[0,1]\times[0,1],\qquad (5.2)$$

which has a zero boundary condition.

The analytic solution of Eq. (5.2) is:

$$u(x,y)=\sin(10\pi x)\cos(2\pi y).$$

Since the x direction changes more rapidly than y direction, we choose X-line relaxation method as our relaxation methods.

For the multiple coarse grid method, we only run the test case on a single processor. The numerical results are shown in Table 5.2 and Fig. 5.9. The "Multi-CG(S1)" stands for generating multiple coarse grid solution by using Strategy 1, and "Multi-CG(S2)" choose the Strategy 2. Since we use the same multiscale multigrid computation for these three methods to compute the fourth order solutions for both the

fine and $(even, even)$ coarse grids, in order to better compare these three methods, we only compare the CPU cost for the extrapolation and interpolation part which is listed in the "Extra-CPU" column of Table 5.2. And the "# Extra-iter" indicates the number of iteration for the operator interpolation or the generation of multiple coarse grid solution.

Table 5.2: Numerical comparison results for the multiple coarse gird method and MG-Six method for Problem 2.

n	strategy	# Extra-iter	Extra-CPU	error
32	MG-Six(line)	**21**	0.002	1.42e-3
	Multi-CG(S1)	**20**	0.000	1.42e-3
	Multi-CG(S2)	**143**	0.008	7.41e-4
64	MG-Six(line)	**29**	0.008	2.95e-5
	Multi-CG(S1)	**27**	0.005	2.95e-5
	Multi-CG(S2)	**190**	0.042	8.30e-6
128	MG-Six(line)	**32**	0.040	5.10e-7
	Multi-CG(S1)	**29**	0.031	5.10e-7
	Multi-CG(S2)	**155**	0.152	1.09e-7
256	MG-Six(line)	**31**	0.480	8.21e-9
	Multi-CG(S1)	**29**	0.368	8.22e-9
	Multi-CG(S2)	**187**	1.783	2.43e-9

From Table 5.2, we can see that Multi-CG(S1) can achieve the same order of accuracy as MG-Six method with less Extra-CPU time. For the Multi-CG(S2) method, since it needs to update the boundary value in each relaxation method, and the relaxation we need to compute the linear system for each coarse grid is the standard point Gauss-Seidel method, the number of iteration is obviously bigger than other two methods, but the accuracy is better than the other two methods. One possible improvement for this is to use multigrid method for every coarse grid like the fine grid, but it needs the extra work to deal with the boundary points. The reason is that only the $(even, even)$ grid points has the full boundary points. For other three coarse grids, standard multigrid method can not solve the problem, we need to add the artificial boundaries as we did for 1D problem and develop special relaxation

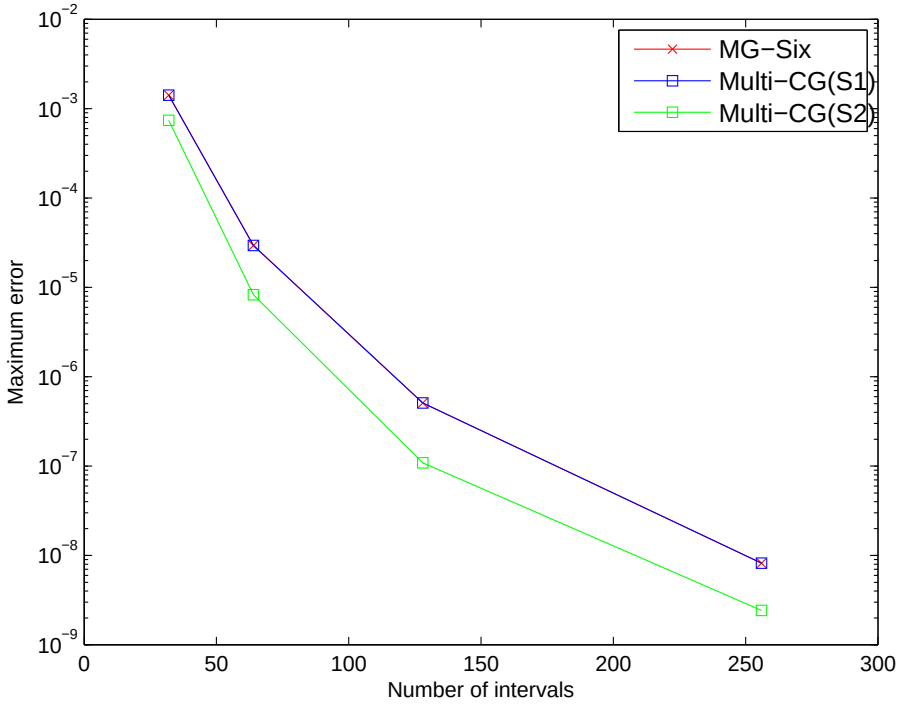


Figure 5.9: Comparison of maximum errors of MG-Six, Multi-CG(S1) and Multi-CG(S2).

method.

5.2 Special W-cycle Multiscale Multigrid

We want to develop a new multiscale multigrid method that can use the existing multilevel (different scale) grid hierarchy to approximate the fourth order solutions instead of running three independent V-cycle multigrid procedures as I introduced in previous chapters. Our method is based on the W-cycle multigrid grid structure as in Fig. 5.10.

5.2.1 Special design

Our aim is to use the multilevel grid property of multigrid method to compute the solutions on two different scale grids and increase the order of accuracy from fourth

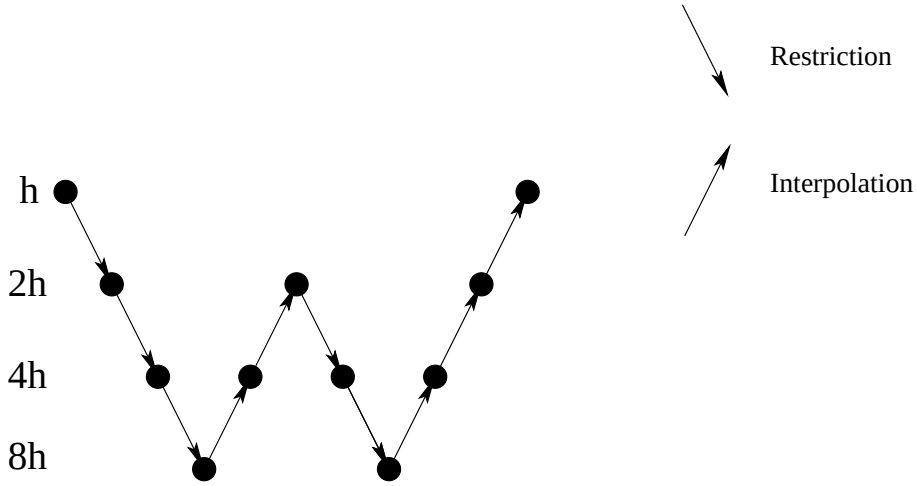


Figure 5.10: Standard W-cycle multigrid method.

order to sixth order by using extrapolation. Sun and Zhang has proposed an ADI method with Richardson extrapolation to approximate sixth order solutions [59]. Their method is the first method that combined the extrapolation procedure in each iteration step on two scale grids.

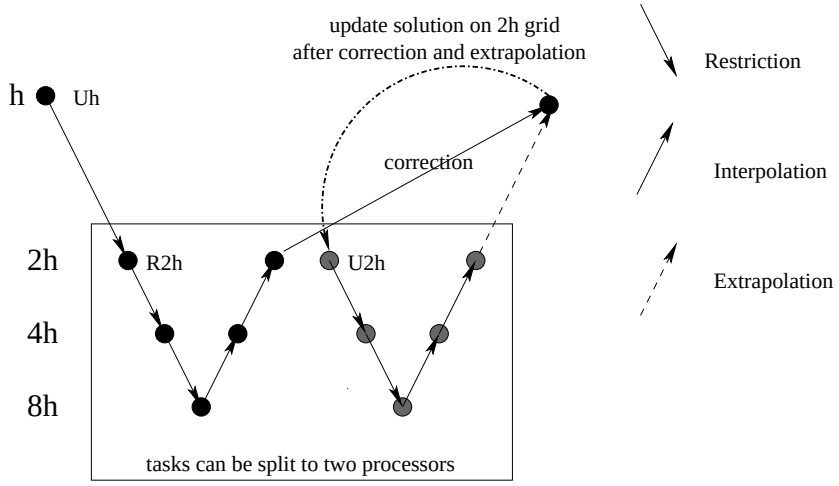


Figure 5.11: Special W-cycle multiscale multigrid method.

I modify the standard W-cycle multigrid method to approximate the sixth order solutions on Ω_{2h} and Ω_h at the same time. The procedure is listed in Fig. 5.11. We can see that the special W-cycle multiscale multigrid method has the similar grid

structure as the standard W-cycle in Fig. 5.10. I modify the right-hand side part of the standard W-cycle multigrid method to approximate the solution on Ω_{2h}, which are the *gray* colored points. Different from the standard W-cycle, there are two $2h$ grids sending results to the fine grid: one from *black* colored points to do the regular correction (interpolation) and the other one is from the *gray* colored points to do the Richardson extrapolation. The fine grid point will also send back two results to the Ω_{2h}: one is the regular residual (restriction) and the other one is the updated sixth order solutions for the coarse grid points. Except for these two additional communication between two scale grids, other computational costs are the same as the standard W-cycle.

The detail of every step of our multiscale W-cycle multigrid method is listed in Algorithm 6. Steps 3 and 4 are two independent $2h$ subproblem (V-cycle based) that can be parallelized using two different processors in each cycle of the algorithm. Steps 5 and 6 are the standard correction using bilinear interpolation. Step 7 is the Richardson extrapolation that can approximate sixth order solutions on $2h$ grid and the corresponding $(even, even)$ grid points on fine grid. We want to mention here that if we rewrite Step 4 to be the same as Step 3, Algorithm 6 will become the standard multigrid W-cycle method. If we remove Steps 4, 7 and 8, it will become the standard multigrid V-cycle method.

Special relaxation methods. We can choose regular relaxation methods as introduced in previous chapters like the point relaxation in lexicographic order and the alternating line relaxation. However, after each cycle only the $(even, even)$ grid points on the fine grid have the sixth order solutions. In order to approximate the sixth order solutions for other grid points on the fine grid as accurately as possible. We prefer to relax the grid points in the order by how many neighboring grid points that have sixth order accuracy, which is similar to our operator based interpolation

scheme in Chapter 2. We listed our point Gauss-Seidel relaxation in group order as follow:

- Update every (odd, odd) grid point.

  For each (odd, odd) point (i, j), the updated solution is

$$\tilde{u}_{i,j}^{h,k+1} = [F_{i,j} - A_{i,j}(1)\tilde{u}_{i+1,j}^{h,k} - A_{i,j}(2)\tilde{u}_{i,j+1}^{h,k} - A_{i,j}(3)\tilde{u}_{i-1,j}^{h,k} - A_{i,j}(4)\tilde{u}_{i,j-1}^{h,k} - A_{i,j}(5)\tilde{u}_{i+1,j+1}^{h,k+1} - A_{i,j}(8)\tilde{u}_{i+1,j-1}^{h,k+1} - A_{i,j}(6)\tilde{u}_{i-1,j+1}^{h,k+1} - A_{i,j}(7)\tilde{u}_{i-1,j-1}^{h,k+1}]/A_{i,j}(0).$$

  Here, each grid point has four neighboring grid points with sixth order accuracy.

- Update every $(odd, even)$ and $(even, odd)$ grid point.

  Here, each grid point has two neighboring grid points with sixth order accuracy.

- Update every $(even, even)$ grid point.

  Here, each grid point has no neighboring grid points with sixth order accuracy.

We want to mention here that for the two independent subproblems as in the rectangular domain in Fig. 5.11, we should use the same relaxation method. If we choose point relaxation for one of them and line relaxation for the other one, the solution accuracy is degraded because we need to keep the fourth order solutions on Ω_h and Ω_{2h} on the same level.

When the W-cycle converged, we should have a sixth order solutions on every $(even, even)$ fine grid point. We can also integrate the operator based interpolation in each fine grid relaxation after the extrapolation. But our experiment showed that it will increase the number of cycles to converge and it takes more computational cost than the post-interpolation (run operator based interpolation procedure after the W-cycle multiscale multigrid method) to approximate the sixth order solutions for other grids.

Algorithm 6 Special Multiscale W-cycle Algorithm

1: Relax ν_1 times on $A_h u_h = b_h$ with initial guess $u_h^{(k)}$

2: Compute $r_{2h}^{(k)} = I_h^{2h}(b_h - A_h u_h^{(k)})$

3:
- Relax ν_1 times on $A_{2h} u_{2h}^{(k)} = r_{2h}^{(k)}$ with initial guess 0
- Compute $r_{4h}^{(k)} = I_{2h}^{4h}(b_{2h} - A_{2h} u_{2h}^{(k)})$
- .....
- Solve $A_{Lh} u_{Lh}^{(k)} = r_{Lh}^{(k)}$
- .....
- Correct $u_{2h}^{(k*)} = u_{2h}^{(k)} + I_{4h}^{2h} u_{4h}^{(k+1)}$
- Relax ν_2 times on $A_{2h} u_{2h} = b_{2h}$ with initial guess $u_{2h}^{(k*)}$

4:
- Relax ν_1 times on $A_{2h} \tilde{u}_{2h} = b_{2h}$ with initial guess $\tilde{u}_{2h}^{(k)}$
- Compute $\tilde{r}_{4h}^{(k)} = I_{2h}^{4h}(b_{2h} - A_{2h} \tilde{u}_{2h}^{(k)})$
- Relax ν_1 times on $A_{4h} \tilde{u}_{4h}^{(k)} = r_{4h}^{(4k)}$ with initial guess 0
- .....
- Solve $A_{Lh} \tilde{u}_{Lh}^{(k)} = \tilde{r}_{Lh}^{(k)}$
- .....
- Correct $\tilde{u}_{2h}^{(k*)} = \tilde{u}_{2h}^{(k)} + I_{4h}^{2h} \tilde{u}_{4h}^{(k+1)}$
- Relax ν_2 times on $A_{2h} \tilde{u}_{2h} = b_{2h}$ with initial guess $\tilde{u}_{2h}^{(k*)}$

5: Correct $u_h^{(k*)} = u_h^{(k)} + I_{2h}^{h} u_{2h}^{(k+1)}$

6: Relax ν_2 times on $A_h u_h = b_h$ with initial guess $u_h^{(k*)}$

7: Do extrapolation for $\tilde{u}_{2h}$ and u_h as

$$u_h(i,j) = \frac{16 u_h(2i,2j) - \tilde{u}_{2h}(i,j)}{15}$$

8: Update $\tilde{u}_{2h}$ as $\tilde{u}_{2h}(i,j) = u_h(2i,2j)$

9: If the correction norm does not reach the convergence, go back to Step 1.

5.2.2 Numerical results

We consider a 2D Poisson equation in the form of

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -52\sin(4x)\cos(6y), \qquad (x, y) \in \Omega = [0, 1] \times [0, 1], \tag{5.3}$$

which has Dirichlet boundary condition.

The analytic solution of Eq. (5.2) is:

$$u(x, y) = \sin(4x)\cos(6y).$$

Table 5.3: Numerical comparison for standard multiscale multigrid method and W-cycle multiscale multigrid method

	Standard Multiscale Multigrid				W-cycle Multiscale Multigrid			
n	# iter	CPU	error	order	# iter	CPU	error	order
16	(15,15),**19**	0.001	6.29e-7	5.9	8, **19**	0.001	6.37e-7	5.9
32	(15,11),**18**	0.006	1.02e-8	6.0	8, **18**	0.005	9.91e-9	6.0
64	(15,11),**16**	0.029	1.59e-10	6.0	7, **16**	0.019	1.36e-10	6.2
128	(14,10),**14**	0.092	2.41e-12	6.0	11, **14**	0.080	1.05e-12	7.0

We tested the multiscale W-cycle multigrid method on a single processor and compare the numerical results with the standard multiscale multigrid method I introduced in Chapter 2. The numerical results are listed in Table 5.3 and Fig. 5.12. In Table 5.3, the "# iter" stands for the number of iterations. The standard multiscale multigrid method contains three parts of the iteration, number of V-cycles for Ω_{2h}, number of V-cycles for Ω_h and the number of operator based interpolation. For the W-cycle based multiscale multigrid method, we also used the operator based interpolation after the W-cycle is converged. Its iteration contains two parts, the number of W-cycle and the number of iterations for interpolation.

From Table 5.3 we can see that the accuracy of the solutions computed by the

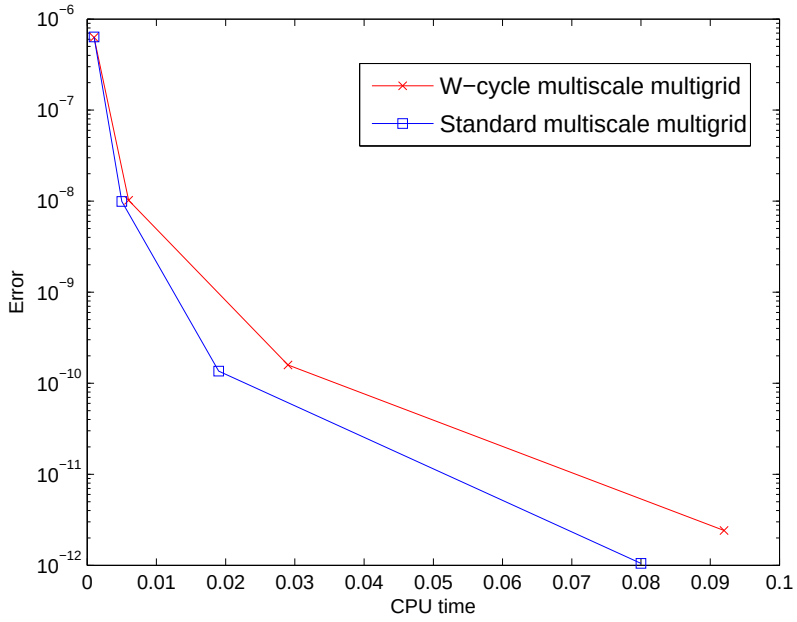


Figure 5.12: Comparison of CPU cost and maximum error for multiscale multigrid method and W-cycle multiscale multigrid method.

W-cycle multiscale multigrid method are more accurate than the standard multiscale multigrid method when $n > 16$. Both solutions can achieve the sixth order solution accuracy and for some finer meshes like $n = 128$, the order of accuracy by the W-cycle multiscale method can reach seventh order. In terms of computational cost (the CPU seconds) with the same mesh size h, the W-cycle multiscale multigrid method is faster than the standard multiscale multigrid method. For the number of iterations, the W-cycle multigrid method can converge within less than 10 cycles which is less than the standard multiscale multigrid method does.

In our numerical experiment, we only tested the W-cycle multiscale multigrid method on a single processor. However, as we mentioned before, these two independent V-cycle structure sub-problems which are in the rectangular domain in Fig. 5.11 can be loaded into two different processors. In addition, the mesh refinement post-interpolation procedure can also be paralleled. Our next research step is to build an

efficient parallel W-cycle multiscale multigrid method to solve some very large scale problems.

5.3 Concluding Remarks

I improved our computational framework by using multiple coarse grid computation technique and W-cycle multiscale multigrid method. The multiple coarse grid computation technique is efficient to increase the order of accuracy from fourth order to sixth order and provides us a good partition structure to convert the solver into a parallel version. The W-cycle multiscale multigrid integrates the Richardson extrapolation in each cycle of the multigrid computation, which can increase the solution accuracy and speed up the convergence.

Our test results showed that these two methods are efficient, robust and accurate. However, for some high Reynolds number cases, the point relaxation scheme may not be efficient to use in multigrid method. Since the line relaxation is difficult to perform by the order of how many neighboring grid points that are sixth order, semicoarsening technique is believed to be the right approach to solving such problems.

Chapter 6

Parallel Iterative Methods for Solving PDEs

From the previous chapters, I know that multigrid algorithms are highly efficient in solving systems of elliptic equations. However, in parallel setting, the sequential nature of the relaxation may degrade the efficiency of the algorithm. The aim of this chapter is to present, evaluate and analyze an efficient parallel multigrid solver for speeding up the computation of 2D/3D PDEs .

6.1 Introduction

Multigrid method are generally accepted as fast and efficient methods for solving PDEs, especially elliptic operators. For these kinds of problems, standard multigrid algorithms based on the iterative methods like Gauss-Seidel or damped Jacobi exhibit an optimal complexity, memory cost and good parallel efficiencies [34]. These characteristics have made multigrid method to be a commonly used solution method in many computational and engineering areas.

For the parallel strategies, generally, there are three main tracks: domain decomposition combined with multigrid, global multigrid partitioning, and concurrent multigrid methods.

The domain decomposition approach is a well known mathematical methodology for the numerical solution of elliptic boundary value problems. It can be used in the original mathematical model to reformulate the original PDE and in the discretization

stage to form parallelizable numerical schemes. It is particularly attractive in parallel computing, because it decomposes the finest grid into several blocks, and multigrid method can be carried out simultaneously in parallel inside each block. These kinds of strategies are widely used with finite element method since they are easier to implement and can be directly applied to general multi-block grids and irregular domain. However, domain decomposition based parallel algorithms are numerically different to the sequential version and may have a negative impact on the convergence rate [53]. It also requires a careful treatment of the connections between different blocks in order to achieve satisfactory convergence rates, which often involves domain overlapping.

For the grid partitioning strategy, the domain decomposition is applied on every grid level, not only on the finest grid. Therefore, for most of the multigrid methods, the parallel approaches based on the second strategy are algorithmically equivalent to the non-parallized one, so the parallel multigrid method using grid partitioning strategy has the same convergence rate of the sequential algorithm [34]. However, it may become very difficult to develop for some complicated applications. In addition, it implies more communication overheads since data exchange is required on every level.

The concurrent multigrid methods uses multiple coarse grids as I mentioned in Chapter 5 to correct fine grid solution. There will be no reduction in the number of grid points as the V-cycle goes down to the coarse levels. The correction on a level is computed as the averaged corrections from multiple grids. In addition, semicoarsening can also be conducted concurrently in different directions. This may result in robust algorithm to deal with strong and weak coupling and to have high level parallelism.

6.1.1 Parallelization by Grid Partition

In this dissertation, I will discuss more about the parallel multigrid method by using grid splitting strategy. As discussed in [83], a completed V-cycle in the multigrid method involves the relaxation procedure on different grids, and transfer of data between different scale of grids by using interpolation and prolongation. The parallelization process needs the finest level grid to be split into subgrids which can be assigned to different processors like in Fig. 6.1. If I use Jacobi type relaxation method, the relaxations can be easily carried out in parallel on different processors. If I choose Gauss-Seidel type or SOR type relaxations methods, the relaxation methods are generally carried out in parallel by using grid coloring schemes. For the red-black ordering, the red black Gauss-Seidel method is actually better than the standard (lexicographical) Gauss-Seidel method and the order of processing for different colors is not important in parallel processing. However, for multi-coloring schemes with more than two colors, the order of processing for different colors may affect the interprocessor communications.

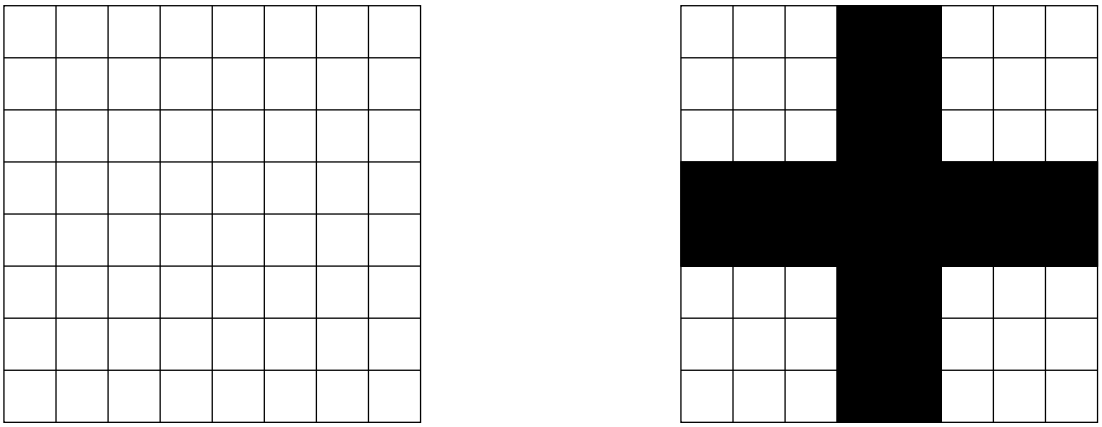


Figure 6.1: Finest level grid is split into four subgrids.

For example, a 9-point computational stencil is used to discretize a 2D PDE, the computational grid can be decoupled with minimum four colors as in Fig. 6.2. I assume that I have two processors $P1$ and $P2$ with assigned grid points. If I process

the relaxation in the order of $(black, red, green, yellow)$, then after both processors have computed the new values of black grid points, $P1$ needs to send the new results at the black grid points on its lower boundary to $P2$ for calculating the new values of red grid points on boundary of $P2$. Similarly, after the processing of red color grid points, $P2$ needs to send the new results of red grid points on the boundary to $P1$ for calculating the green grid points on boundary. For the green and yellow grid points, I will see that the same updating strategy will be used to calculate the new results for these two groups of grid points. Therefore, for two processors, after each sweep of relaxation they require four message passing with each message containing the new computed solutions on half of the boundary points.

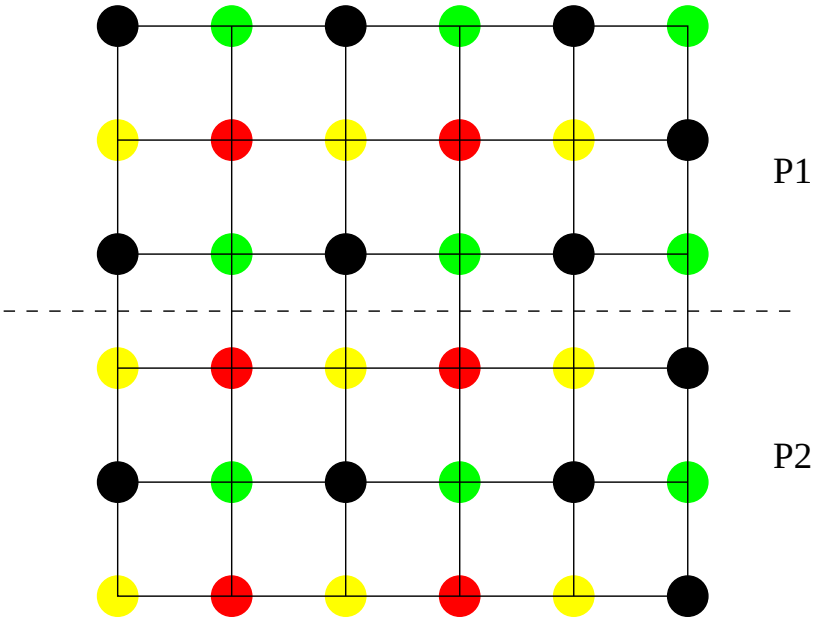


Figure 6.2: Illustration of decoupled four color grid for fourth order computational scheme of 2D problem.

If I process the relaxation in the order of $(black, green, yellow, red)$, then $P1$ does not need to send the new results at the black grid points on its boundary to $P2$ because the relaxation of green color grid points on $P2$ does not need the solutions at the black color grid points on $P1$. However, after the relaxation for green color

grid points, $P1$ needs to send the new calculated solution at green color grid points on its boundary to $P2$ because $P2$ needs these results to calculate the new solutions for red color grid points. By using the same updating method, there is another communication between $P1$ and $P2$ for yellow and red grid points. So, the total message passing between two processors for each relaxation is two. For the above two relaxation schemes, the total amount of data that are transfered between two processors are the same, but the second scheme is better than the first one because it needs fewer number of message passing.

6.2 Parallelization for 3D Convection Diffusion Equation using Multi-color Scheme

For the convection-diffusion equations, It is well known that the relaxation direction has a strong influence on the convergence of the lexicographic Gauss-Seidel type methods when the convection-diffusion equation represents strong convection in a particular direction. The direction of the relaxation should generally follow the convection direction, otherwise the convergence could be seriously deteriorated. When the convection direction is unknown, it is important to use robust methods whose convergence does not strongly depend on the convection direction.

The Jacobi iterative method for solving 3D convection-diffusion equation can be fully parallized but the drawback is that when it is used as a smoother in the multigrid method, it usually needs to be damped by a damping factor which is difficult to estimate for most practical problems. Even with a damping factor obtained by trail and error, the smoothing effect of the Jacobi relaxation is poor.

The lexicographical Gauss-Seidel relaxation has better smoothing effect than the Jacobi method. For the parallelization and vectorization benefit, I may reorder the grid points by dividing them into several colored groups so that the parallel relaxation can be carried out within each group. Considering the fourth order compact finite

difference schemes, in the 2D case, four colors are needed to decouple the 9-point compact scheme. In the 3D case, as I mentioned in Chapter 4, there are two computational schemes: 19-point scheme and 15-point scheme. For the 19-point scheme, four colors are sufficient to completely decouple the 3D grid points [27] as in Fig. 6.3. Note that updating a red point needs the values of 2 nearest and 4 next nearest grid point marked with each of the other three colors. For 19-point scheme, all grid points with the same color can be updated simultaneously on parallel computers.

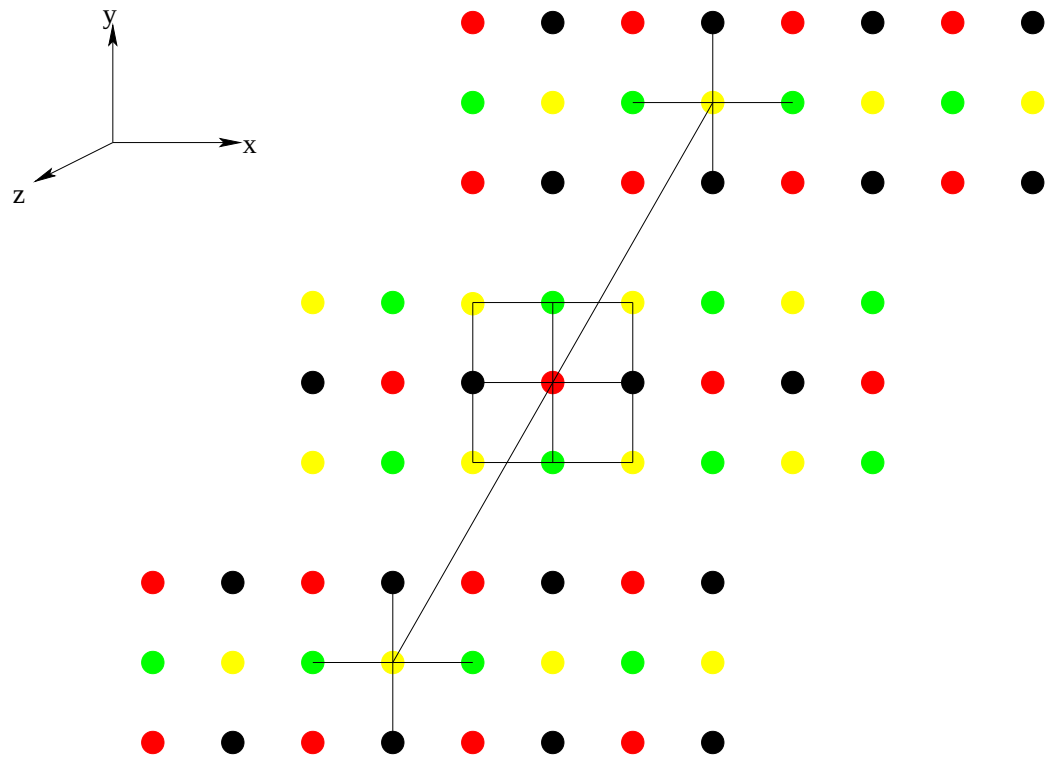


Figure 6.3: Illustration of decoupled four color grid for 19-point computational scheme of 3D problem.

For the 15-point scheme, it has been shown that the computational grid can be decoupled with only two colors [26] as in Fig. 6.4. I can see that each red color grid point is connected by 14 black color grid points. Therefore, two red-black relaxation sweeps will update the entire grid for 3D discretized grids, while the four color schemes needs four relaxation sweeps to update the whole grids.

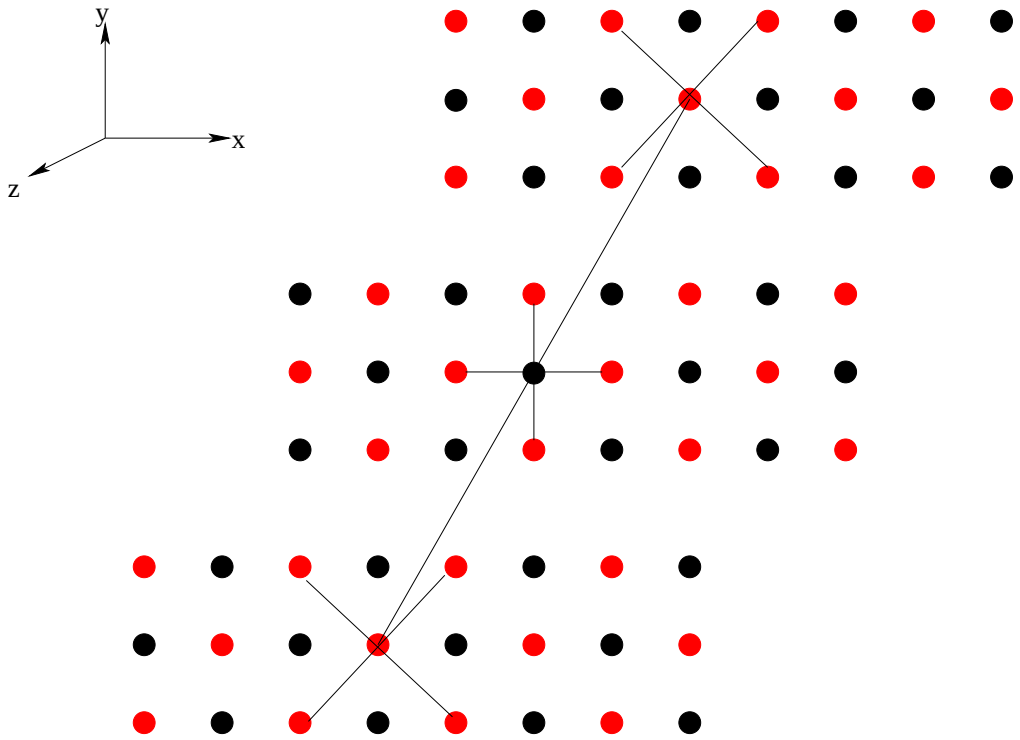


Figure 6.4: Illustration of decoupled two color grid for 15-point computational scheme of 3D problem.

In this dissertation, I prefer the four-color 19-point scheme because I have learned from chapter 4 that when Reynolds number becomes huge, the magnitude of the convection coefficient affects the order of accuracy inversely and the 15-point scheme is degraded worse than the 19-point scheme. In addition, the four-color Gauss-Seidel relaxation also leads to highly vectorizable and parallelizable solvers. Efficient vectorization is obtained since relaxation on grid points with the same color no longer contains vector feedback dependencies [7]. The computations are performed in a number of parallel operations equal to the number of independent colors. In addition to the gains in parallelization and vectorization, practical experience showed that better convergence and smoothing properties are usually obtained with multiple color ordering [7].

6.3 Numerical Results

All the codes were implemented and tested on the Lipscomb HPC Cluster at the University of Kentucky. Each node is a Intel Xeon X5650 (Westmere) with 2.66GHz and 36GB shared memory. The code is written in standard Fortran 77 and is run in double precision. Parallelization is achieved by using OpenMP parallel derivatives to the loops in relaxation and residual computation procedure.

I tested two 3D convection-diffusion equations using 19 point computational stencil with multiscale multigrid method. I first compare the numerical results from standard point Gauss-Seidel relaxation method and four-color Gauss-Seidel relaxation scheme. Then I compare the performance for single and multiple processors. The computation is terminated when the norm of correction vector is below the stopping criteria (10^{-10}). The domain Ω for the two test problems I solved was chosen as the unit cube $(0,1)^3$.

6.3.1 Test Problem 1

For the first test problem, I consider the following 3D convection-diffusion equation

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

This problem has variable coefficients and the constant Re represents the magnitude of the convection coefficients and simulates the Reynolds number in a flow simulation. The Dirichlet boundary conditions and the forcing term f are set to satisfy the exact solution.

I tested the first problem using multiscale multigrid method to achieve sixth order accuracy. Both the four color relaxation scheme and the standard point relaxation

scheme have been tested. The numerical results are listed in Table 6.1.

Table 6.1: Comparison of lexicographical Gauss-Seidel relaxation and four-color Gauss-Seidel relaxation.

	Lexicographical Gauss-Seidel		Four Color Gauss-Seidel	
n	iter	error	iter	error
16	(12,13),46	6.13e-5	(10,11),46	6.13e-5
32	(13,12),47	1.40e-6	(11,11),47	1.40e-6
64	(12,12),44	2.56e-8	(11,10),44	2.55e-8
128	(12,11),40	4.61e-10	(10,10),40	4.60e-10

I noted that both schemes can achieve the comparable accuracy of the computed solutions, but the four-color Gauss- Seidel relaxation scheme cost less CPU time for every case. For the number of iterations, I mentioned in previous section that better convergence and smoothing properties are usually obtained by multicolor relaxation scheme. The numerical results in Table 6.1 is a good support because for every n, the number of iterations by using four-color Gauss-Seidel is less than that from lexicographical Gauss-Seidel scheme. A graphic comparison is shown in Fig. 6.5.

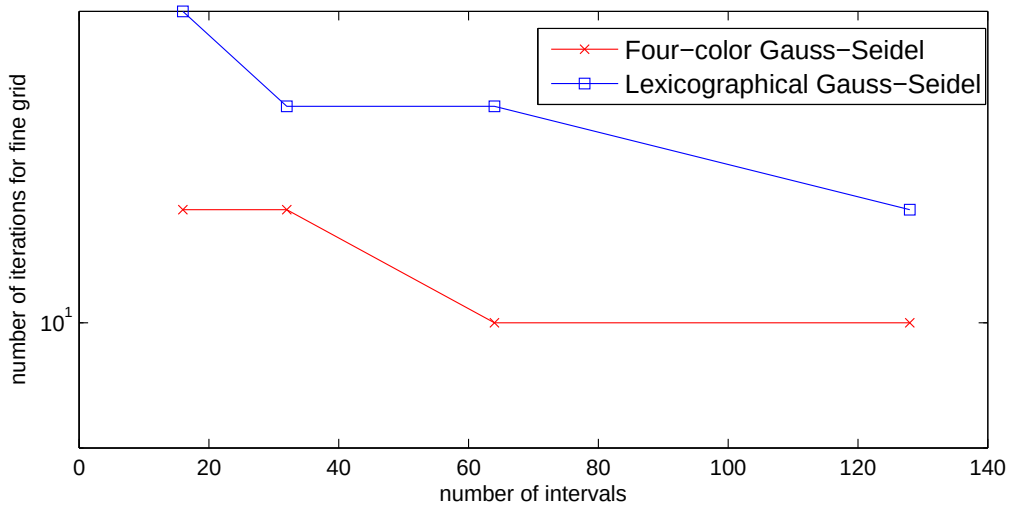


Figure 6.5: Comparison of number of iterations for lexicographical and four-color Gauss-Seidel schemes .

The CPU costs and the number of iterations with multiple processors is compared

in Table 6.2 with different meshsizes and a fixed $Re = 10$. I noted that from one processor (similar to serial computation) to eight processors, I keep the same convergence rate. The CPU costs decrease when the number of processors increases. However, the speedup for Problem 1 using different number of processors are not very impressive. When $n = 128$, the speedup with two processors is 1.27 and it is 1.91 when I choose eight processors. The speedup is lower than the results from Zhang's paper [78] because I use multiscale multigrid method which needs additional computation like the higher order interpolation between Ω_{2h} and Ω_h grids as well as the operator based interpolation procedure to approximate the sixth order solutions. These procedures have not been parallelized.

When the Re is increased to 100, the numerical results are listed in Table 6.3. I can see that the number of iterations is affected by the higher Reynolds number because it costs more iterations to converge. I point out that the speedup with 8 processors and $n = 128$ is 2.44 which is better than that from Table 6.2. For better understanding, I also provide the graphic showing the speedup of different number of processors as in Fig. 6.6 and Fig 6.7.

Table 6.2: Multiprocessors CPU costs and number of iterations for Problem 1 with different meshsizes and $Re = 10$.

	n=32		n=64		n=128	
	CPU	iter	CPU	iter	CPU	iter
1 processor	0.267	(13,12),47	2.237	(12,12),44	18.701	(12,11),40
2 processors	0.212	(13,12),47	1.723	(12,12),44	14.645	(12,11),40
3 processors	0.187	(13,12),47	1.369	(12,12),44	12.279	(12,11),40
4 processors	0.154	(13,12),47	1.263	(12,12),44	10.541	(12,11),40
8 processors	0.146	(13,12),47	1.104	(12,12),44	9.789	(12,11),40

Table 6.3: Multiprocessors CPU costs and number of iterations for Problem 1 with different meshsizes and $Re = 100$.

	n=32		n=64		n=128	
	CPU	iter	CPU	iter	CPU	iter
1 processor	0.513	(21,25),52	4.519	(25,25),52	36.571	(25,24),48
2 processors	0.316	(21,25),52	3.241	(25,25),52	27.139	(25,24),48
3 processors	0.274	(21,25),52	2.611	(25,25),52	19.889	(25,24),48
4 processors	0.229	(21,25),52	2.271	(25,25),52	16.430	(25,24),48
8 processors	0.216	(21,25),52	1.962	(25,25),52	15.101	(25,24),48

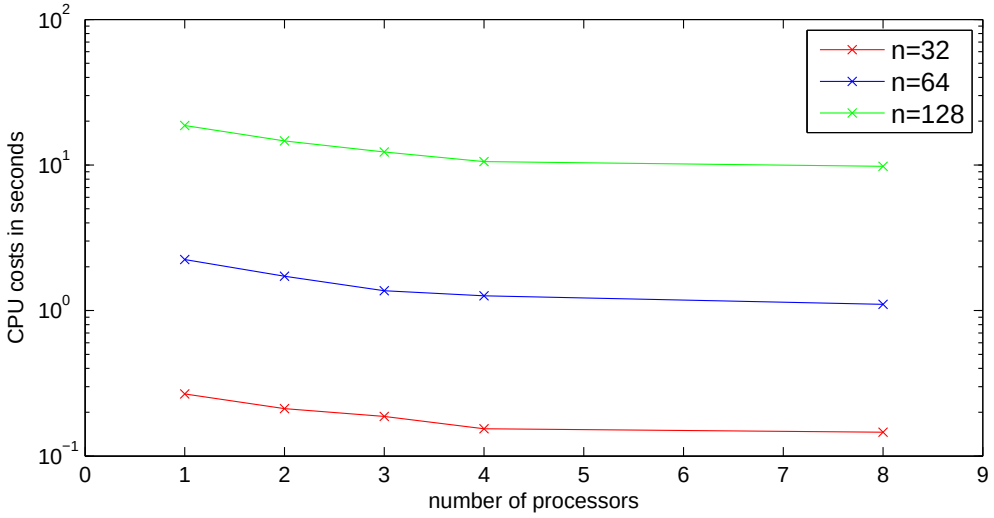


Figure 6.6: Comparison of CPU costs for different number of processors for Problem 1 with $Re = 10$.

6.3.2 Test Problem 2

For test problem 2, I chose the constant coefficient problems as

$$\begin{cases} u(x,y,z) &= x^2 + y^2 + z^2, \\ p(x,y,z) &= q(x,y,z) = r(x,y,z) = Re. \end{cases}$$

I first tested $Re = 10$ then I increase the Re to be 100. The numerical results are in Table 6.4, Table 6.5, Fig. 6.8 and Fig. 6.9.

Once again, the convergence rate is the same for different number of processors.

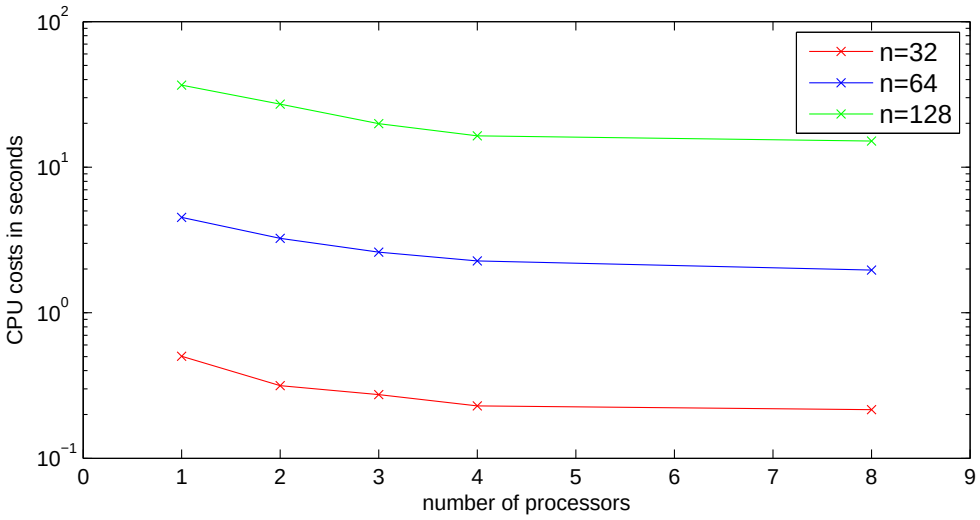


Figure 6.7: Comparison of CPU costs for different number of processors for Problem 1 with $Re = 100$.

When Re increases the number of iterations increased because the problems is convection dominated one may keep the grid independent convergence by using line relaxation and plane relaxation method. However, these relaxation methods need new techniques to achieve efficient parallelism.

Table 6.4: Multiprocessors CPU costs and number of iterations for Problem 2 with different meshsizes and $Re = 10$.

	n=32		n=64		n=128	
	CPU	iter	CPU	iter	CPU	iter
1 processor	0.307	(16,16),6	2.627	(16,15),11	20.172	(15,14),20
2 processors	0.231	(16,16),6	1.934	(16,15),11	15.033	(15,14),20
3 processors	0.173	(16,16),6	1.487	(16,15),11	11.801	(15,14),20
4 processors	0.136	(16,16),6	1.211	(16,15),11	9.931	(15,14),20
8 processors	0.127	(16,16),6	1.063	(16,15),11	8.693	(15,14),20

6.4 Concluding Remarks

I studied the parallelization and vectorization potential of the Gauss-Seidel relaxation by partitioning the grid space with four colors. It was argued by Bandy [7] that eight

Table 6.5: Multiprocessors CPU costs and number of iterations for Problem 2 with different meshsizes and $Re = 100$.

	n=32		n=64		n=128	
	CPU	iter	CPU	iter	CPU	iter
1 processor	0.665	(30,37),8	7.004	(31,45),12	61.426	(45,47),16
2 processors	0.483	(30,37),8	5.003	(31,45),12	46.127	(45,47),16
3 processors	0.347	((30,37),8	3.843	(31,45),12	34.412	(45,47),16
4 processors	0.292	(30,37),8	3.062	(31,45),12	27.008	(45,47),16
8 processors	0.260	(30,37),8	2.632	(31,45),12	23.131	(45,47),16

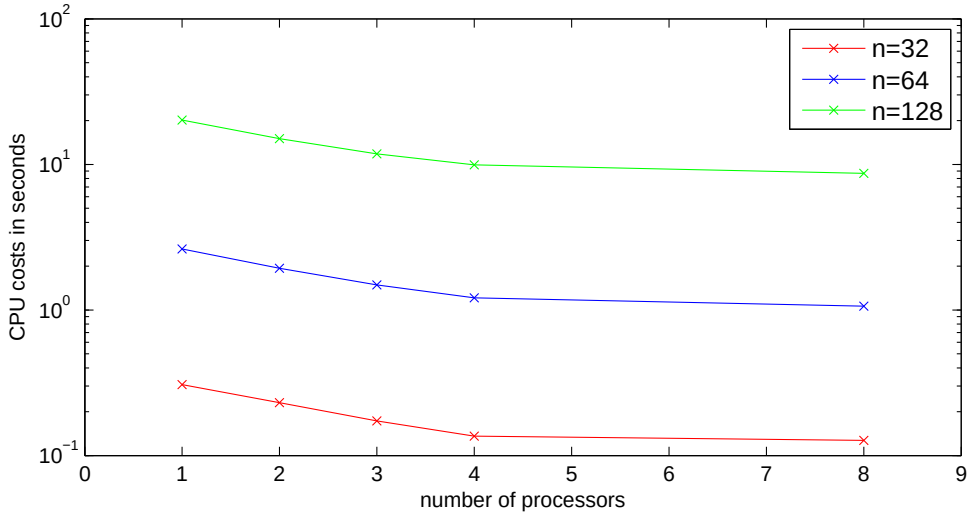


Figure 6.8: Comparison of CPU costs for different number of processors for Problem 2 with $Re = 10$.

colors are needed to decouple the grid points if all 27 points of the unit cubic grid are used in a 3D finite difference scheme. Our compact scheme is truly advantageous in the sense that I am able to reduce the number of stencil points by a quarter and reduce the number of necessary colors by a half.

Our test problems include both constant and variable coefficients with different Reynolds numbers. The numerical results show that the parallelized and the sequential implementation have the same convergence rate and the accuracy of the computed solutions.

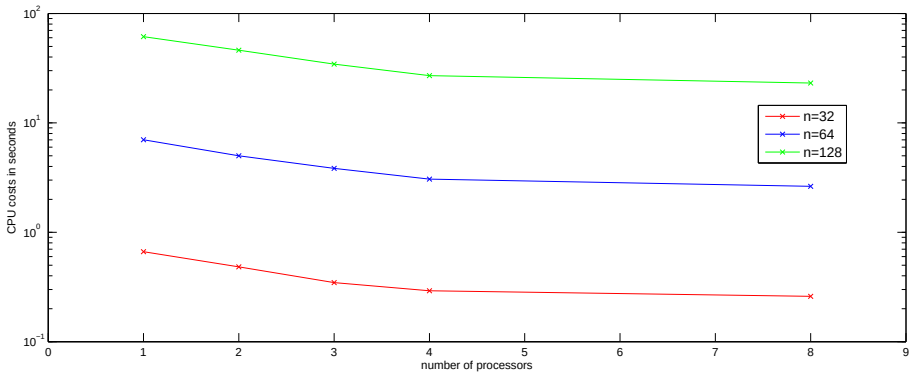


Figure 6.9: Comparison of CPU costs for different number of processors for Problem 2 with $Re = 100$.

Chapter 7

Conclusion and Future Work

This dissertation presents my research work on building efficient and scalable computational framework to solve partial differential equations. Our work involves high order discretization of partial differential equations, efficient solver for discretized linear systems, and mathematical verifications. In this chapter, I summarize my dissertation work and present my future research plan.

7.1 Research Accomplishments

Partial differential equations (PDEs) are widely used to model many computational science and engineering and industry (CSEI) problems. My dissertation concerns the development of high order compact finite difference discretization schemes and scalable linear system solvers for achieving fast and high resolution numerical solution of PDEs. It has been known that the two main tracks in developing high order compact difference schemes, the implicit schemes [32] and the explicit schemes [24], have their own advantages and disadvantages. The major advantage of implicit schemes is that they can be used to solve many types of PDEs without major modifications. However, the approximations of the first and second derivatives are unnecessary for some applications and it has been proved in [82] that some implicit schemes may produce nonphysical oscillations. In contrary, the explicit schemes approximate the PDEs directly and have additional advantage to suppress the nonphysical numerical

oscillations. However, this class of high order compact schemes become more complicated to develop in higher dimensions. As far as I know, there is no existing explicit compact scheme on a single scale grid that is higher than the fourth order accuracy. In response to this gap, I developed sixth order compact difference schemes for solving different PDEs using multiscale multigrid method, operator based interpolation, multiple coarse grid computation, and W-cycle multiscale multigrid method.

New Sixth Order Compact Schemes for 2D/3D PDEs

I developed a new sixth order compact finite difference scheme which is based on the nine-point fourth order computational stencil on two scale grids. By using the Richardson extrapolation technique on two grids with different scales, I computed the sixth order solution on the coarse grid. Then the sixth order coarse grid solution was interpolated to the corresponding fine grid points as the initial value to approximate the sixth order solution for other fine grid points. Based on this idea, different operator-based interpolation algorithms are presented in chapters 2, 3 and 4 to approximate the sixth order fine grid solution for different types of PDEs.

Anisotropic and Convection Dominated Problems

Multigrid method is among the fastest and most efficient iterative methods for solving many types of PDEs. However, standard multigrid method may fail to achieve optimal grid-independent convergence rate for convection diffusion equation with high Reynolds number and Poisson equation that has anisotropic discrete operators. In Chapter 2, I used alternating line relaxation scheme to treat the 2D anisotropic Poisson equations, which solved the problems with less iterations and better grid independent convergence rate compared with point relaxation scheme. For 2D and 3D convection dominated convection-diffusion equations, the high Reynolds numbers affect the solution accuracy and the convergence rate inversely. I conducted the

mathematical analysis to prove that the computed solutions from the fourth order discretization will be degraded to second order if the Reynolds number is bigger than a certain magnitude. To treat such problems I used residual scaling techniques together with alternating line relaxation and plane relaxation schemes to solve the 2D and 3D equations, respectively.

V-cycle Based Multiscale Multigrid Method

For solving the large sparse linear systems arising from discretization, I developed a multiscale multigrid method. The multiscale multigrid method is similar to the full multigrid method, but it does not start from the coarsest level. It is a V-cycle based multiscale multigrid method, which contains three independent V-cycle procedures on three grids with different scales. The major advantage of such multiscale multigrid method is that it has an optimal computational cost similar to that of a full multigrid method and can bring us the converged fourth order solutions on both the Ω_{2h} and Ω_h.

Multiple Coarse Grid Computation

In general, a fine grid in d-dimensional space can be easily coarsened into 2^d coarse grids which can be used in the parallel superconvergent multigrid method to speed up the convergence. But this idea has never been used to increase the order of solution accuracy for the fine grid solution. For both 1D and 2D problems, I derived the fourth order computational stencil for each coarse grid and solve these coarse grid problems independently. Therefore, these fourth order coarse grid solutions can cover every fine grid point, and the standard Richardson extrapolation can be used for every fine grid point to approximate the sixth order solution without operator based interpolation. This idea has a very good potential to be used in parallel machines.

W-cycle Based Multiscale Multigrid Method

Since the multilevel grids are already available for geometric multigrid method, the primary goal of the multiscale multigrid method is to use the existing multilevel grids to solve the problem with better solution accuracy than that computed from the standard geometric multigrid method, which only accelerates the convergence. In order to further using this advantage of the geometric multigrid method, I developed a W-cycle multiscale multigrid method that can approximate the sixth order solutions more accurate than the V-cycle based multiscale method and converge faster. Our improved W-cycle multiscale multigrid method modified part of the standard W-cycle multigrid method to solve both the fine grid and the coarse grid solutions. In addition, it integrated the Richardson extrapolation technique in each cycle which can increase the order of accuracy and speed up the convergence.

Parallel Multiscale Multigrid Method using Multicolor Relaxation

Our goal is to develop a new sixth order parallel solver that can be used to solve many types of PDEs. The first step is to use OpenMP to develop an parallel multiscale multigrid method in shared memory computers. In order to achieve good parallelism, I performed multicolor Gauss-Seidel relaxation in multigrid method. The numerical results showed that the parallel multicolor Gauss-Seidel method has the same convergence rate as the sequential algorithm. However, since the multicolor scheme is based on the point relaxation scheme and it is not efficient for solving convection-diffusion equation with very huge Reynolds numbers, one can provide a new parallel multiscale multigrid method by using line relaxation or plane relaxation.

7.2 Future Work

Some fundamental work has been done to build the numerical computational framework for solving large scale PDEs in many CSEI applications. However, my research

work is just the first step in developing a useful model that can be applied to real-world problems. In the near future, I will continue my research on problems like non-rectangular domains, solutions requiring mesh adaptation, discontinuous coefficients or forcing, time-dependent PDEs, etc. However, my first plan for future work is in the following.

More Efficient and Practical Computational Framework

It is worth pointing that our solution method can be applied to solve general 2D/3D convection-diffusion equations with Dirichlet boundary conditions, but there is still a lot of work that needs to be done to develop a useful method that can be applied to real-world problems. For example, if a problem involves a derivative boundary conditions such as Neumann boundary conditions, a one-sided finite difference approximation for the derivative can be used [46]. For PDEs with nonsmooth coefficients, the multigrid method converges slowly when the coefficients exhibit large jumps in discontinuity [2, 9, 16] or large oscillations [19, 60], and the benefits of employing high-order compact schemes and Richardson extrapolation are mostly or completely lost. Special techniques like semicoarsening, the algebraic multigrid method, and frequency decomposition [18, 28] can be used to solve these types of PDEs. For PDEs with irregular domains, finite element methods may be more suitable to handle these cases. Special finite difference discretization methods can also be used, like Sun's fourth order scheme on face-centered cubic grids [58]. I am also concerned with some problems in the field of computational fluid dynamics exhibiting local solution behaviors that require higher-level resolution in one area of the domain than in others. Higher-level resolution may be computed by using local mesh refinement techniques [61, 77].

Parallel Multiscale Multigrid using Multiple Coarse Grid Computation

For very large scale problems in realistic applications, the number of unknowns is at least in the order of millions. The load of computing is very heavy for some 2D applications with fine mesh, and even more burdensome for 3D applications. It may take a single workstation or PC hours, days, or weeks for a computation task. My first plan is to develop parallel numerical linear algebra library for high performance solution of large sparse linear systems arising from systems of PDEs frequently encountered in CSEI modeling and simulation applications. I will parallelize the code through message passing interface (MPI) libraries by the method of domain decomposition to cut the solution time. The parallel code may be run on super computers or Linux clusters.

High Performance Preconditioning

In a joint research work with Dr. Jeonghwa Lee and Jun Zhang, we have developed an efficient and scalable preconditioned iterative method for solving large dense linear systems [69, 70], where the coefficient matrix is a complex valued matrix arising from discretizing the integral equation of electromagnetic scattering. For some scattering structures, the coefficient matrix can be very ill conditioned and the standard block diagonal preconditioner based on LU factorization makes the Krylov iterative methods converge more slowly or even diverge. To handle this problem, I apply stabilized singular value decomposition (SSVD) to each diagonal block to improve the stability and efficiency of the block diagonal preconditioner. The SSVD preconditioner can solve some scattering problem that cannot be solved by the standard block diagonal preconditioner and provide us a good structure to parallelize the code. I plan to continue my research in designing robust preconditioners for both the sparse and dense linear systems. In particular, I will focus on employing the multilevel preconditioned Krylov subspace method to solve convection-diffusion equations. Furthermore, I would like to collaborate with experts in electromagnetics to analyze and study some

other types of scattering structures and build efficient preconditioners for these real cases.

Bibliography

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

[2] R. E. Alcouffe, A. Brandt, J. E. Dendy Jr., and J. W. Painter, The multigrid method for the diffusion equations with strongly discontinuous coefficients, *SIAM J. Sci. Statist. Comput.*, **2**(4), 430-454, 1981.

[3] U. Ananthakrishnaiah, R. Monahar, and J. W. Stephenson, Fourth-order finite difference three-dimensional general linear elliptic problems with variable coefficients, *Numer. Meth. Partial Diff. Eqns.*, **3**, 229-240, 1987.

[4] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation, Washington, 1984.

[5] F. F. Ashby and R. D. Falgout, A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations, *Nuclear Sci. Engrg.*, **124**, 145-159, 1996.

[6] O. Axelsson, *Iterative Solution Methods*, Cambridge Univ. Press, Cambridge, 1994.

[7] V. A. Bandy, *Black box multigrid for convection-diffusion equations on advanced computers*, Ph.D. Thesis, University of Colorado at Denver, 1996

[8] T. L. Beck, Multigrid high order mesh refinement techniques for composite grid electrostatics calculations, *J. Comput. Chem.*, **20**(16), 1731-1739, 1999.

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

[10] A. Brandt and I.Yavneh, Accelerated multigrid convergence and high-Reynolds recirculating flows, *SIAM J. Sci. Comput.*, **14**, 607-626, 1993.

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

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

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

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

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

[16] J. E. Dendy Jr., Black box multigrid for nonsymmetric problems, *Appl. Math. Comput.*, **13**, 261-283, 1983.

[17] B. Düring, M. Fournié, and A. Jüngel, High order compact finite difference schemes for nonlinear Black-Scholes equation, *Int. J. Theor. Appl. Finance*, **6**(7), 767-789, 2003.

[18] J. E. Dendy Jr., and C. C. Tazartes, Grandchild of the frequency decomposition multigrid method, *SIAM J. Sci. Comput.*, **16**(2), 307-319, 1994.

[19] B. Engquist and E. Luo, Convergence of a multigrid method for elliptic equations with highly oscillatory coefficients, *SIAM J. Numer. Anal.*, **34**, 2254-2273, 1997.

[20] M. Fournié, Numerical discretization of energy-transport model for semiconductors using high-order compact schemes, *Appl. Math. Lett.*, **15**(6), 721-726, 2002.

[21] 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.

[22] G. H. Golub and J. M. Ortega, *Scientific Computing: An Introduction with Parallel Computing*, Academic Press, San Diego, 1993.

[23] P. M. Gresho and R. L. Lee, Don't suppress the wiggles-they're telling you something!, *Computers & Fluids*, **9**, 223-253, 1981.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] W. Hackbusch and S. Sauter, The frequency decomposition multigrid method, Part I: application to anisotropic equations, *Numer. Math.*, **56**, 229-245, 1989.

[29] J. Hyman, Mesh refinement and local inversion of elliptic partial differential equations, *J. Comput. Phys.*, **23**, 124-134, 1977.

[30] S. Karaa and J. Zhang, On convergence and performance of iterative methods for solving variable coefficient convection-diffusion equation with a fourth-order compact difference scheme, *Comput. Math. Appli.*, **44**, 457-479, 2002.

[31] Y. Kwon, R. Manohar, and J. W. Stephenson, A single cell finite difference approximations for Poisson's equation in three variables, *Appl. Math. Notes*, **2**, 13-20, 1982.

[32] S. K. Lele, Compact finite difference schemes with spectral-like resolution, *J. Comput. Phys.*, **103**, 16-42,1992.

[33] 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.

[34] I. M. Llorente and F. Tirado, Relationships between efficiency and execution time of full multigrid methods on parallel computers, *IEEE Trans. on Parallel and Distributed Systems*, **8**, 562-573, 1997.

[35] I. M. Llorente and N. D. Melson, Behavior of plane relaxation methods as multigrid smoothers, *Electron. T. Numer. Ana.*, **10**, 92-114, 2000.

[36] I. M. Llorente, M. Prieto and B. Diskin, A parallel multigrid solver for 3D convection and convection-diffusion problems, *Parallel Comput.*, **27**(13), 1715-1741, 2001.

[37] R. J. MacKinnon and R. W. Johnson, Differential-equation-based representation of truncation errors for accurate numerical simulation, *Int. J. Numer. Methods Fluids*, **13**, 739-757, 1991.

[38] W. A. Mulder, A new multigrid approach to convection problems, *J. Comput. Phys.*, **83**(2), 303-323, 1989.

[39] W. A. Mulder, A high-resolution Euler solver based on multigrid, semi-coarsening, and defect correction, *J. Comput. Phys.*, **100**(1), 91-104, 1992.

[40] S. Nagarajan, S. K. Lele, and J. H. Ferziger, A robust high-order compact method for large eddy simulation, *J. Comput. Phys.*, **191**(2), 392-419, 2003.

[41] S. Oh, A. B. Milstein, C. A. Bouman, and K. J. Webb, A general framework for nonlinear multigrid inversion, *IEEE Trans. Image Process.*, **14**(1), 125-140, 2005.

[42] C. W. Oosterlee, A GMRES-based plane smoother in multigrid to solve 3-D anisotropic fluid flow probems, *J. Compt. Phys.*, **30**, 41-53, 1997.

[43] D. Peaceman and H. Rachford, The numerical solution of elliptic and parabolic differential equations, *Journal. of SIAM*, 3:28-41, 1955.

[44] C. Reisinger and G. Wittum, On multigrid for anisotropic equation and variational inequalities: pricing multi-dimensional European and American options, *Comput. Vis. Sci.*, **7**(3-4), 189-197, 2004.

[45] L. F. Richardson, The approximate arithmetical solution by finite differences of physical problems including differential equations, with an application to the stresses in a masonry dam, *Philosophical Transactions of the Royal Society of London*, **Series A** 210, 307-357, 1910.

[46] K. Rahul and S. N. Bhattacharyya, One-sided finite-difference approximations suitable for use with Richardson extrapolation, *J. Comput. Phys.*, **219**, 13-20, 2006.

[47] P. J. Roache, *Computational Fluid Dynamics*, Hermosa, Albuquerque, NM, 1976.

[48] Y. Saad and M. H. Schultz, GMRES: a generalized minimal residual method for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, **7**, 856-869, 1986.

[49] Y. Saad, ILUT: a dual threshold incomplete LU preconditioner, *Numer. Linear Algebra Appl.*, **1**, 387-402, 1994.

[50] Y. Saad, *Iterative Methods for Sparse Linear Systems, 2nd edition*, SIAM, Philadelphia, PA, 2003.

[51] K. Sakurai, T. Aoki, W. H. Lee, and K. Kato, Poisson equations solver with fourth-order accuracy by using interpolated differential operator scheme, *Comput. Math. Appli.*, **43**, 621-630, 2002.

[52] S. Schaffer, Higher order multigrid method, *Math. Comput.*, **43**(167), 89-115, 1984.

[53] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition, Parallel Multigrid Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.

[54] W. F. Spotz, *High-Order Compact Finite Difference Schemes for Computational Mechanics*, PhD thesis, University of Texas at Austin, Austin, TX, 1995.

[55] 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.

[56] W. F. Spotz and G. F. Carey, A high-order compact formulation for the 3D Poisson equation, *Numer. Method. Partial Diff. Eqns.*, **12**, 235-243, 1996.

[57] J. C. Strikerda, *Finite Difference Schemes and Partial Difference Equations*, Wadsworth Publishing Corporation, Belmont, CA 1989.

[58] H. Sun, N. Kang, J. Zhang, and E. S. Carlson, A fourth-order compact difference scheme on face centered cubic grids with multigrid method for solving 2D convection diffusion equation, *Math. Comput. Simulat.*, **63**(6), 651-661, 2003.

[59] 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.

[60] S. Ta'asan, *Multigrid methods for highly oscillatory problems*, Ph.D Thesis, Weizmann Institute of Science, Rehovat, Israel, 1984.

[61] R. Teigland and I. K. Eliassen, A multiblock/multilevel mesh refinement procedure for CFD computations, *Int. J. Number. Meth. Fluids*, **36**(5), 519-538, 2001.

[62] C. Thole and U. Trottenberg, Basic smoothing procedures for the multigrid treatment of elliptic 3d operators, *Appl. Math. Comput.*, **19**, 333-345, 1986.

[63] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

[64] H. A. van der Vorst, Bi-CGSTAB: a Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems, SIAM J. Sci. Statist. Comput., **13**(2), 631-644, 1992.

[65] P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, Chichester, England, 1992.

[66] Y. Wang and J. Zhang, Sixth order compact scheme combined with multigrid method and extrapolation technique for 2D Poisson equation, *J. Comput. Phys.*, **228**, 137-146, 2009.

[67] Y. Wang and J. Zhang, Integrated fast and high accuracy computation of convection diffusion equations using multiscale multigrid method, to appear in *Numerical Methods for Partial Differential Equations*.

[68] Y. Wang and J. Zhang, High accuracy and scalable multiscale multigrid computation for 3D convection diffusion equation, to appear in *Journal of Computational and Applied Mathematics*.

[69] Y. Wang, J. Lee, and J. Zhang, A short survey on preconditioning techniques for large scale dense complex linear systems in electromagnetics, *Int. J. Comput. Math.*, **84**(8), 1211-1223, 2007.

[70] Y. Wang, J. Lee, and J. Zhang, SVD stabilized block diagonal preconditioner for large scale dense complex linear systems in electromagnetics, *Proceedings of the Seventh International Conference on Computational and Mathematical Methods in Science and Engineering*, pp:407-417, Chicago, IL, 2007.

[71] J. Zhang, *Multigrid Acceleration Techniques and Applications to the Numerical Solution of Partial Differential Equations*, PhD thesis, The George Washington University, Washington, DC, 1997.

[72] J. Zhang, On convergence of iterative methods with a fourth-order compact scheme, *Appl. Math. Lett.*, **10**, 49-55, 1997.

[73] J. Zhang, Accelerated multigrid high accuracy solution of the convection-diffusion equation with high Reynolds number, *Numer. Meth. Partial Diff. Eqns.*, **13**, 77-92, 1997.

[74] J. Zhang, Residual scaling techniques in multigrid, I: equivalence proof, *Appl. Math. Comp.*, **86**, 283-303, 1997.

[75] J. Zhang, An explicit fourth-order compact finite difference scheme for three dimensional convection-diffusion equation, *Commun. Numer. Methods Engrg.*, **14**, 209-218, 1998.

[76] J. Zhang, Preconditioned iterative methods and finite difference schemes for convection diffusion, *Appl. Math. Comput.*, **109**(1), 11-30, 2000.

[77] 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.

[78] 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.

[79] J. Zhang, A note on accelerated high accuracy multigrid solution of convection-diffusion equation with high Reynolds number, *Numer. Meth. Partial Diff. Eqns.*, **16**, 1-10, 2000.

[80] 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.

[81] 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.

[82] 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.

[83] J. Zhu, *Solving Partial Differential Equations on Parallel Computers*, World Scientific, Mississippi State University, 1993.

Vita

Personal Data:

Name: Yin Wang

Date of Birth: 09/01/1982

Place of Birth: Qionglai, Sichuan, China

Educational Background:

- Masters of Science in Computer Science, University of Kentucky, 2008.
- Bachelor of Science in Computer Science, University of Science and Technology of China, China, 2004.

Professional Experience:

- Research Assistant, 08/2004 - 01/2006.
  Department of Computer Science, University of Kentucky, Lexington, KY, USA.
- Teaching Assistant, 01/2006 - 08/2009.
  Department of Computer Science, University of Kentucky, Lexington, KY, USA.
- Undergraduate Research Assistant, 07/2003 - 07/2004.
  National High Performance Computing Center, University of Science and Technology of China, Hefei, Anhui, China.

Awards:

- Presidential Graduate Fellowship, University of Kentucky, 2009-2010.
- Travel support from NSF funding to attend the Workshop on Mathematical Biology and Numerical Analysis, Athens, GA, August 2009.

- Student award for the IMACS World Congress, Computational and Applied Mathematics & Applications in Science and Engineering, Athens, GA, August 2009.

- Fellowship from Lawrence Berkeley National Laboratory to attend the Eighth DOE Advanced Computational Software (ACTS) Collection Workshops, Berkeley, CA, August 2007.

- Second prize for the poster section at the Seventh International Conference on Computational and Mathematical Methods in Science and Engineering, Chicago, IL, June 2007.

- Student travel support from Graduate School Fellowship, University of Kentucky, 2006-2009.

- Outstanding Student Fellowship, University of Science and Technology of China, China, 2001 - 2002.

Publications:

- Yin Wang and Jun Zhang, Integrated fast and high accuracy computation of convection diffusion equations using multiscale multigrid method, to appear in *Numerical Methods for Partial Differential Equations*.

- Yin Wang and Jun Zhang, Sixth order compact scheme combined with multigrid method and extrapolation technique for 2D Poisson equation, *Journal of Computational Physics*, **228**:137-146, 2009.

- Yin Wang, Jeonghwa Lee, and Jun Zhang, A short survey on preconditioning techniques for large scale dense complex linear systems in electromagnetics, *International Journal of Computer Mathematics*, **84**(8):1211-1223, 2007.

- Yin Wang and Jun Zhang, High accuracy and scalable multiscale multigrid computation for 3D convection diffusion equation, to appear in *Journal of Computational and Applied Mathematics*.

- Yin Wang, Jeonghwa Lee, and Jun Zhang, SVD stabilized block diagonal preconditioner for large scale dense complex linear systems in electromagnetics, in *Proceedings of the Seventh International Conference on Computational and Mathematical Methods in Science and Engineering*, pp:407-417, Chicago, IL, 2007.

- Yin Wang and Jun Zhang, Solving 3D convection diffusion equation using sixth order compact difference scheme and multiscale multigrid method, *The IMACS World Congress on Computational and Applied Mathematics & Application in Science and Engineering*, Athens, GA, 2009.

- Jeonghwa Lee, Yin Wang, Cai-Cheng Lu, and Jun Zhang, Two-step preconditioning techniques for electromagnetic wave scattering problems, *The IMACS World Congress on Computational and Applied Mathematics & Application in Science and Engineering*, Athens, GA, 2009.