Algebraic Multigrid Poisson Equation Solver

by

Xinchen Guo

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

May 2015

ABSTRACT

From 2D planar MOSFET to 3D FinFET, the geometry of semiconductor devices is getting more and more complex. Correspondingly, the number of mesh grid points increases largely to maintain the accuracy of carrier transport and heat transfer simulations. By substituting the conventional uniform mesh with non-uniform mesh, one can reduce the number of grid points. However, the problem of how to solve governing equations on non-uniform mesh is then imposed to the numerical solver. Moreover, if a device simulator is integrated into a multi-scale simulator, the problem size will be further increased. Consequently, there exist two challenges for the current numerical solver. One is to increase the functionality to accommodate non-uniform mesh. The other is to solve governing physical equations fast and accurately on a large number of mesh grid points.

This research first discusses a 2D planar MOSFET simulator and its numerical solver, pointing out its performance limit. By analyzing the algorithm complexity, Multigrid method is proposed to replace conventional Successive-Over-Relaxation method in a numerical solver. A variety of Multigrid methods (standard Multigrid, Algebraic Multigrid, Full Approximation Scheme, and Full Multigrid) are discussed and implemented. Their properties are examined through a set of numerical experiments. Finally, Algebraic Multigrid, Full Approximation Scheme and Full Multigrid are integrated into one advanced numerical solver based on the exact requirements of a semiconductor device simulator. A 2D MOSFET device is used to benchmark the performance, showing that the advanced Multigrid method has higher speed, accuracy and robustness.

Dedicated to my family for their unreserved support for the two years' study and research, and precious encouragement.

ACKNOWLEDGEMENTS

This work is motivated by a EEE 598 Advanced Device Modeling course project given by Dr. Dragica Vasileska. I thank her for guiding me into this field of study and providing abundant support in course study, research, and career development. Without her support, this research could not have been finished.

I am grateful to my committee members Dr. David K. Ferry and Dr. Stephen M. Goodnick for their advice on my thesis. Doing projects of Dr. Ferry's course brought me some new ideas and a new way of understanding numerical methods.

I would like to extend my appreciation to the School for Engineering of Matter, Transport, and Energy at Arizona State University for providing me the opportunity to purse my master's degree.

Finally, I would also like to thank all my colleagues who provided me precious help in these two years' research and study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

1.1 Technology Progress of FinFET Devices

Conventional scaling law worked for several decades before the gate length of MOSFET scaled down to tens of nanometers. Unlike the gate length, the thickness of gate oxide is hard to scale. For a 90nm gate length device, the gate oxide was scaled down to about 1.2nm, which caused exponential increase in oxide leakage current (Bohr, 2011). To suppress the short channel effect of conventional MOSFET, FinFET device is first introduced as a self-aligned double-gate MOSFET device (Hisamoto *et al.*, 2000). Because FinFET device is a quasi-planar device, it shares some of the layout and fabrication techniques of conventional MOSFET device (Huang *et al.*, 1999). Intel stopped the scaling down of conventional 2D MOSFET at 32nm and switched to 3D tri-gate device for 22nm gate length and beyond (Cartwright, 2011; Kuhn, 2012). 3D devices became the trend for future applications. Therefore, it is very important to have efficient 3D simulation tools to facilitate research and development.

1.2 Need for Multi-scale Simulation

Conventional circuit simulation software uses macroscopic continuous equations as its governing law. A particle-based device simulator only deals with single semiconductor device without considering its electrical and thermal properties under a certain circuit configuration. It typically uses semi-classical transportation theory as the governing law. Similarly, first principle simulation focuses on the quantum behavior of atoms and electrons in bulk materials ignoring various boundary conditions which exist in real applications. From electronic properties to electrical properties, both length and time span several orders. Thus, it is very important to properly couple each level of simulation and develop high performance simulation algorithms and computing hardware to put multi-scale simulation into real applications to discover macroscopic phenomenon on the basis of accurate nanoscale quantum theories.

1.3 Importance of a Fast Poisson's Equation Solver

A typical device simulator consists of two self-consistent kernels (Vasileska *et al.*, 2010). One is transport kernel, which varies largely for different device scale. The other is the electromagnetic kernel, which remains unchanged.

The breakdown of total runtime of a 2D MOSFET device simulator is shown in table 1.1. This program is written in FORTRAN. The test run is on ASU Advanced Computing Center computer node Nehalem. The Poisson equation solver code segment belongs to the electromagnetic kernel. The free-flight scattering code segment belongs to the Monte Carlo transport kernel. The rest of the segments are auxiliary code. In 2D case, current Poisson solver already takes about 45.3% of total run-time, while the transport kernel takes the second largest portion of 27.4%. Because the complexity of Successive-Over-Relaxation (SOR) method is higher than the complexity of any other segment, with increasing number of grid points, Poisson's equation solver will take longer time. When a simulator is expanded to 3D device, the Poisson's equation solver will take about 90% of total time. Therefore, it becomes very important to develop a very efficient Poisson's equation solver to enable 3D devices based multi-scale simulation.

There are two major steps to numerically solve a partial differential equation representing the Poisson equation. The first step is to discretize the continuous differential equation into a matrix system with either finite-element method or finite-difference

Table 1.1: The Breakdown of Total Runtime of a 2D MOSEFT Simulator.

Code Segment	**Complexity (N grid points)**	**Time Cost**
Initialization & Post-processor	5	6
Poisson Solver (SOR)	8	9
Free Flight Scattering	8	9
NEC Scheme	8	9
Output	8	9

method. The second step is to solve the matrix system. Direct methods include Gauss elimination, block method, fast Fourier transform, etc. Iterative methods include Jacobi, Gauss-Seidel, SOR, Conjugate Gradients, Multigrid, etc. These methods have different time complexity and requirement for computer memory space. Among them, Multigrid method has the lowest complexity and memory space requirements. Therefore, this research focuses on developing Multigrid solver to replace commonly used SOR and Conjugate Gradient solvers.

Chapter 2

NUMERICAL APPROACHES FOR THE SOLUTION OF THE POISSON EQUATION FOR SPATIALLY VARYING PERMITTIVITY AND ON NON-UNIFORM MESH

2.1 Poisson's Equation

In the electromagnetic kernel in a device simulator, Maxwell's equations are the governing laws (Vasileska *et al.*, 2010). Gauss's law is

$$\nabla \bullet \mathbf{D} = \rho. \tag{2.1}$$

Gauss's law for magnetism is

$$\nabla \bullet \mathbf{B} = 0. \tag{2.2}$$

Faraday's law of induction is

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}. \tag{2.3}$$

Ampere's law with Maxwell's extension is

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}. \tag{2.4}$$

where $\mathbf{D}$ is the displacement vector; ρ is the free charge density; $\mathbf{B}$ is the magnetic flux density; $\mathbf{E}$ is the electric field strength; $\mathbf{H}$ is the magnetizing field; and $\mathbf{J}$ is the current density.

They are a set of four partial differential equations disregarding their physical meanings. Maxwell's equations cannot be used without specifying the relationship between $\mathbf{D}$ and $\mathbf{E}$ and the relationship between $\mathbf{H}$ and $\mathbf{B}$. For homogeneous media, the relations are

$$\mathbf{D} = \varepsilon \mathbf{E}, \tag{2.5}$$

$$\mathbf{H} = \frac{\mathbf{B}}{\mu}, \tag{2.6}$$

where ε is the permittivity of the medium and μ is the permeability of the medium. Including both physical and mathematical degrees of freedom, $\mathbf{E}$ and $\mathbf{B}$ can be expressed as

$$\mathbf{E} = -\nabla\varphi - \frac{\partial \mathbf{A}}{\partial t}, \tag{2.7}$$

$$\mathbf{B} = \nabla \times \mathbf{A}. \tag{2.8}$$

where $\mathbf{A}$ is the vector potential and φ is the potential. The Coulomb Gauge imposes the constraint:

$$\nabla \bullet \mathbf{A} = 0. \tag{2.9}$$

With the Coulomb Gauge, Gauss's law can be transformed to the Poisson's equation:

$$\nabla(\varepsilon\nabla(\varphi)) = -\rho. \tag{2.10}$$

In oxide regions of the device, the concentration of electrons and holes are too small that the free charge density is treated as zero. In semiconductor regions, free charges density consists of four parts: hole concentration, electron concentration, ionized donor concentration, and ionized acceptor concentration. Thus the Poisson's equation becomes of the following form:

$$\nabla(\varepsilon\nabla(\varphi)) = \begin{cases} 0 & \text{In Oxide} \\ -q(p - n + N_D^+ - N_A^-) & \text{In Semiconductor,} \end{cases} \tag{2.11}$$

where q is the elementary charge, p is the hole concentration, n is the electron concentration, N_D^+ is the ionized donor concentration, and N_A^- is the ionized acceptor concentration. $N_D^+ - N_A^-$ equals to the net ionized impurity concentration (Falgout, 2006). p and n can be calculated from either Boltzmann or Fermi-Dirac statistics. This research focuses on non-degenerate semiconductors and equilibrium condition;

therefore Boltzmann statistics can be used to calculate electron and hole concentration (Aymerich-Humet *et al.*, 1981; Lundstrom and Schuelke, 1982). Then:

$$p = N_i \exp(-\frac{\varphi}{V_T}), \tag{2.12}$$

$$n = N_i \exp(\frac{\varphi}{V_T}), \tag{2.13}$$

where N_i is the intrinsic carrier density and V_T is the thermal voltage.

Finite-difference and finite-element are two commonly used methods to discretize the continuous Poisson's equation for numerical calculation. Finite-volume method is chosen for this research. To avoid second order finite difference, the integral form of Gauss's law is used instead of its partial differential form (Vasileska *et al.*, 2010):

$$\oint_A \varepsilon(\nabla\varphi)\mathrm{d}\mathbf{A} = -\rho \mathrm{d}x\mathrm{d}y\mathrm{d}z. \tag{2.14}$$

In 2D case, the integral is

$$\oint_C \varepsilon(x,y)(\frac{\partial\varphi}{\partial x}\hat{x} + \frac{\partial\varphi}{\partial y}\hat{y})\mathrm{d}\mathbf{L} = -\rho(x,y)\mathrm{d}x\mathrm{d}y. \tag{2.15}$$

The scheme of five-point stencil in 2D is shown in figure 2.1. Here x and y represent the mesh spacing rather than the coordinates. The line integral on the left hand side of the Poisson's equation can be calculated by adding the line integral on each edge of the dashed line rectangular box. Correspondingly, the Poisson's equation becomes

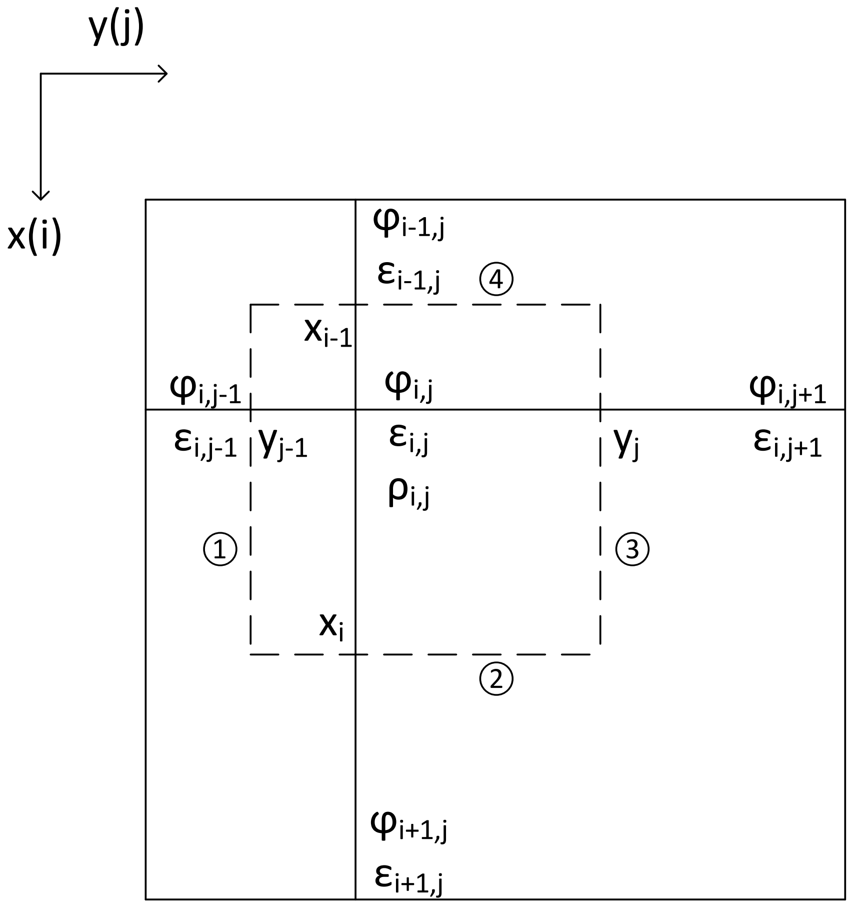


Figure 2.1: Scheme of 5-point Stencil in 2D.

$$\textcircled{1} + \textcircled{2} + \textcircled{3} + \textcircled{4} = \rho_{i,j}\mathrm{d}x\mathrm{d}y \tag{2.16}$$

$$\textcircled{1} = \frac{1}{2}(\varepsilon_{i,j-1} + \varepsilon_{i,j})\frac{\varphi_{i,j-1} - \varphi_{i,j}}{y_{j-1}}\mathrm{d}x$$

$$\textcircled{2} = \frac{1}{2}(\varepsilon_{i-1,j} + \varepsilon_{i,j})\frac{\varphi_{i-1,j} - \varphi_{i,j}}{x_{i-1}}\mathrm{d}y$$

$$\textcircled{3} = \frac{1}{2}(\varepsilon_{i,j+1} + \varepsilon_{i,j})\frac{\varphi_{i,j+1} - \varphi_{i,j}}{y_{j}}\mathrm{d}x$$

$$\textcircled{4} = \frac{1}{2}(\varepsilon_{i+1,j} + \varepsilon_{i,j})\frac{\varphi_{i+1,j} - \varphi_{i,j}}{x_{i}}\mathrm{d}y$$

$$\mathrm{d}x = \frac{x_i + x_{i-1}}{2}$$

$$\mathrm{d}y = \frac{y_j + y_{j-1}}{2}.$$

With proper substitution and simplification, the final expression of discretized Poisson's equation is

$$
\begin{aligned}
&\frac{(\varepsilon_{i+1,j}+\varepsilon_{i,j})}{x_i(x_i+x_{i-1})}\varphi_{i+1,j}+\frac{(\varepsilon_{i-1,j}+\varepsilon_{i,j})}{x_{i-1}(x_i+x_{i-1})}\varphi_{i-1,j}\\
&+\frac{(\varepsilon_{i,j+1}+\varepsilon_{i,j})}{y_j(y_j+y_{j-1})}\varphi_{i,j+1}+\frac{(\varepsilon_{i,j-1}+\varepsilon_{i,j})}{y_{j-1}(y_j+y_{j-1})}\varphi_{i,j-1}\\
&-[\frac{(\varepsilon_{i+1,j}+\varepsilon_{i,j})}{x_i(x_i+x_{i-1})}+\frac{(\varepsilon_{i-1,j}+\varepsilon_{i,j})}{x_{i-1}(x_i+x_{i-1})}\\
&+\frac{(\varepsilon_{i,j+1}+\varepsilon_{i,j})}{y_j(y_j+y_{j-1})}+\frac{(\varepsilon_{i,j-1}+\varepsilon_{i,j})}{y_{j-1}(y_j+y_{j-1})}]\varphi_{i,j}\\
&=-q(N_i\exp(-\frac{\varphi_{i,j}}{V_T})-N_i\exp(\frac{\varphi_{ij}}{V_T})+dop_{i,j}).
\end{aligned}
\tag{2.17}
$$

The final expression of discretized Poisson's equation is nonlinear because of the electron and hole concentration on the right hand side. Common iterative methods cannot work with nonlinear system. Therefore, proper linearization is necessary for Poisson's equation. Taylor series is employed to linearize the exponential function:

$$
e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots . \tag{2.18}
$$

Applying a small update δ to current potential, the new potential is

$$
\varphi^{new} = \varphi^{old} + \delta. \tag{2.19}
$$

So the electron and hole density becomes

$$
N_i\exp(-\frac{\varphi}{V_T}) \to N_i\exp(-\frac{\varphi^{old}+\delta}{V_T}) = N_i\exp(-\frac{\varphi_{old}}{V_T})\exp(-\frac{\delta}{V_T}), \tag{2.20}
$$

$$
N_i\exp(\frac{\varphi}{V_T}) \to N_i\exp(\frac{\varphi^{old}+\delta}{V_T}) = N_i\exp(\frac{\varphi_{old}}{V_T})\exp(\frac{\delta}{V_T}). \tag{2.21}
$$

Using the first two terms of Tayler series of exponential function, the δ terms become

$$
\exp(-\frac{\delta}{V_T}) \approx (1-\frac{\delta}{V_T}), \tag{2.22}
$$

$$
\exp(\frac{\delta}{V_T}) \approx (1+\frac{\delta}{V_T}). \tag{2.23}
$$

Substituting the linearized terms back to the electron and hole concentration, the new equations are

$$p = N_i \exp(-\frac{\varphi_{old}}{V_T})(1 - \frac{\varphi^{new} - \varphi^{old}}{V_T}), \tag{2.24}$$

$$n = N_i \exp(\frac{\varphi_{old}}{V_T})(1 + \frac{\varphi^{new} - \varphi^{old}}{V_T}). \tag{2.25}$$

Moving φ^{new} to the left hand side of Poisson's equation, the new Poisson's equation is

$$\begin{aligned}
&\frac{(\varepsilon_{i+1,j} + \varepsilon_{i,j})}{x_i(x_i + x_{i-1})}\varphi^{old}_{i+1,j} + \frac{(\varepsilon_{i-1,j} + \varepsilon_{i,j})}{x_{i-1}(x_i + x_{i-1})}\varphi^{old}_{i-1,j} \\
&+ \frac{(\varepsilon_{i,j+1} + \varepsilon_{i,j})}{y_j(y_j + y_{j-1})}\varphi^{old}_{i,j+1} + \frac{(\varepsilon_{i,j-1} + \varepsilon_{i,j})}{y_{j-1}(y_j + y_{j-1})}\varphi^{old}_{i,j-1} \\
&- [\frac{(\varepsilon_{i+1,j} + \varepsilon_{i,j})}{x_i(x_i + x_{i-1})} + \frac{(\varepsilon_{i-1,j} + \varepsilon_{i,j})}{x_{i-1}(x_i + x_{i-1})} \\
&+ \frac{(\varepsilon_{i,j+1} + \varepsilon_{i,j})}{y_j(y_j + y_{j-1})} + \frac{(\varepsilon_{i,j-1} + \varepsilon_{i,j})}{y_{j-1}(y_j + y_{j-1})}]\varphi^{new}_{i,j} \\
&- q[N_i \exp(-\frac{\varphi^{old}_{i,j}}{V_T}) + N_i \exp(\frac{\varphi^{old}_{i,j}}{V_T})]\frac{\varphi^{new}_{i,j}}{V_T} \\
&= -q[N_i \exp(-\frac{\varphi^{old}_{i,j}}{V_T})(1 + \frac{\varphi^{old}_{i,j}}{V_T}) \\
&- N_i \exp(\frac{\varphi^{old}_{i,j}}{V_T})(1 - \frac{\varphi^{old}_{i,j}}{V_T}) + dop_{i,j}].
\end{aligned} \tag{2.26}$$

Within one iteration, all the coefficients on the left hand side and the forcing term on the right hand side are known. Thus, the Poisson's equation is linear now for iterative solvers.

This linear system can either be solved as a matrix $Ax = b$ or multiple scalar equations. Matrix form is ideal to isolate the mathematical problem from changing geometries and physical problems. However, scalar equations are coupled to a certain geometry, which makes it more intuitive.

To further simplify the Poisson's equation and reduce computation load, some of the coefficients and variables can be normalized:

$$\frac{\varphi}{V_T} \rightarrow \varphi, \tag{2.27}$$

$$\varepsilon \rightarrow \varepsilon_0 \varepsilon_r, \tag{2.28}$$

$$x \rightarrow L_D x_r, \tag{2.29}$$

$$y \rightarrow L_D y_r, \tag{2.30}$$

$$dop \rightarrow \frac{dop}{N_i}, \tag{2.31}$$

where L_D is the intrinsic Debye length

$$L_D = \sqrt{\frac{\varepsilon V_T}{q N_i}}. \tag{2.32}$$

In a semiconductor device, contacts are treated as Dirichlet boundaries. Neumann boundary conditions are applied to the rest of device geometry. Boundary conditions can be implicitly buried in the coefficient matrix A. For scalar equations, treatment of boundary conditions is similar.

Neumann boundary condition is given by

$$\frac{\partial \varphi}{\partial \mathbf{n}} = f, \tag{2.33}$$

where $\mathbf{n}$ is the normal to the boundary and f is a scalar function. In semiconductor device, scalar function f is set to zero, creating so called ghost point to help solving matrix problems.

A 16 points mesh grid containing inner points, Dirichlet boundary, and Neumann boundary is shown in figure 2.2. Therefore, only 8 points (from φ_1 to φ_8), including inner points and Neumann boundary points, need to be calculated.

With typical 5-point 2D stencil, Poisson's equation at point 2 is

$$a_2\varphi_1 + b_2\varphi_6 + d_2\varphi_3 + e_2\varphi_b - c_2\varphi_2 = f_2. \tag{2.34}$$

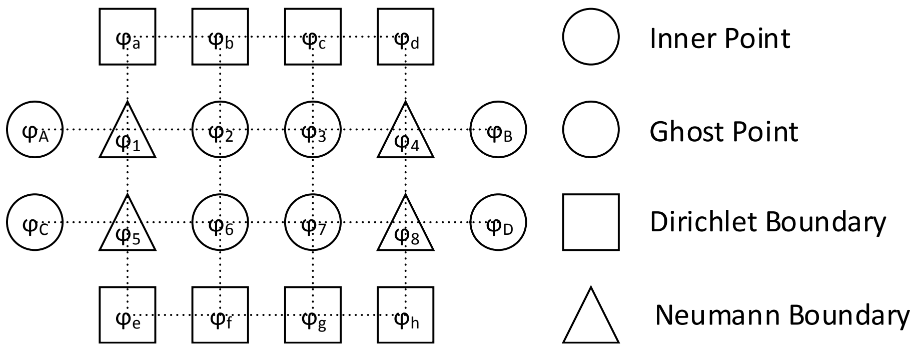


Figure 2.2: A 16 Points Mesh Grid to Discuss Boundary Conditions.

In this equation, e_2 and φ_b are known. Thus they can be moved to the right hand side of Poisson's equation, leaving the unknown variables on the left hand side:

$$a_2\varphi_1 + b_2\varphi_6 + d_2\varphi_3 - c_2\varphi_2 = f_2 - e_2\varphi_b. \tag{2.35}$$

Following this procedure, Dirichlet boundary points are eliminated from the coefficient matrix A and combined into forcing terms.

At point 1, Poisson's equation is:

$$a_1\varphi_a + b_1\varphi_5 + d_1\varphi_2 + e_1\varphi_A - c_1\varphi_1 = f_1. \tag{2.36}$$

A ghost point A is introduced to help solve Neumann boundary problem. Neumann boundary conditions in semiconductor devices give zero value for the first order derivative of the potential, which ensures that a_1 equals to d_1 and φ_A equals to φ_2. So Poisson's equation at point 1 can be simplified together with the treatment for Dirichlet boundary as following:

$$b_1\varphi_5 + 2d_1\varphi_2 - c_1\varphi_1 = f_1 - a_1\varphi_a. \tag{2.37}$$

Now only 8 unknown points are included in the Poisson's equation system. The matrix representation is

$$\begin{bmatrix} -c_1 & 2d_1 & & b_1 & & & & \\ a_2 & -c_2 & d_2 & & b_2 & & & \\ & a_3 & -c_3 & d_3 & & b_3 & & \\ & & 2a_4 & -c_4 & & & b_4 & \\ e_5 & & & -c_5 & 2d_5 & & & \\ & e_6 & & a_6 & -c_6 & d_6 & & \\ & & e_7 & & a_7 & -c_7 & d_7 & \\ & & & e_8 & & 2a_8 & -c_8 & \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \\ \varphi_5 \\ \varphi_6 \\ \varphi_7 \\ \varphi_8 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \end{bmatrix} - \begin{bmatrix} e_1\varphi_a \\ e_2\varphi_b \\ e_3\varphi_c \\ e_4\varphi_d \\ b_5\varphi_e \\ b_6\varphi_f \\ b_7\varphi_g \\ b_8\varphi_h \end{bmatrix}. \tag{2.38}$$

Source and drain contacts are Ohmic contact (charge neutrality):

$$\rho = (p - n + dop) = \left(\exp(-\frac{\varphi}{V_T}) - N_i \exp(\frac{\varphi}{V_T}) + dop\right) = 0. \tag{2.39}$$

With n-doping, potential is positive. So $N_i \exp(-\frac{\varphi}{V_T})$ is a small value and can be ignored. Then potential is

$$\varphi = V_T \ln(\frac{dop}{N_i}). \tag{2.40}$$

On the contrary, p-doping makes the potential negative. $N_i \exp(\frac{\varphi}{V_T})$ can be ignored. The potential is

$$\varphi = -V_T \ln(\frac{-dop}{N_i}). \tag{2.41}$$

Since this research only discusses equilibrium condition, no source or drain voltage is applied. Potential that results from the charge neutrality condition is used as Dirichlet boundary value at source and drain region.

Gate voltage can be applied in equilibrium condition. A gate voltage is chosen to neutralize the potential barrier resulting in zero potential for Dirichlet boundary condition in the gate region. This further simplifies physical model while leaving mathematical model unchanged. Similarly, substrate contact is simply set to zero.

For oxide region, the initial potential is set to zero. For semiconductor region, two initial guesses are used to test the robustness of the numerical solvers. One method is to set the potential to zero. The other method is to use charge neutrality to calculate the initial potential from doping profile.

2.2 Iterative Methods

With proper discretization, linearization, and normalization, Poisson's equation is transformed into a linear matrix equation $Ax = b$. Direct methods and iterative methods are two major category of matrix solver. Direct methods are based on Gauss elimination, which is intuitive but costs a large amount of time and computer memory. Therefore, iterative methods are typically used in computer solvers. Successive-Over-Relaxation method (SOR), which is based on Gauss-Seidel method, is commonly used in device simulators.

The coefficient matrix A can be decomposed into three parts:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}. \tag{2.42}$$

Strictly upper triangular matrix is U:

$$U = \begin{bmatrix} 0 & a_{1,2} & \cdots & a_{1,n} \\ 0 & 0 & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{2.43}$$

Strictly lower triangular matrix is L:

$$L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{2,1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & 0 \end{bmatrix}. \tag{2.44}$$

The diagonal matrix is D:

$$D = \begin{bmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n,n} \end{bmatrix}. \tag{2.45}$$

Then Gauss-Seidel iteration is

$$x^{p+1} = -\left(L + D\right)^{-1} U x^p + \left(L + D\right)^{-1} b. \tag{2.46}$$

However, this form is not actually implemented due to its complexity. A simplified version is commonly used (Vasileska *et al.*, 2010):

$$x^{p+1} = D^{-1}\left(b - Lx^{p+1} - Ux^p\right). \tag{2.47}$$

Introducing a weighting coefficient α, SOR method can be derived by combining old result x^p and new result x^{p+1} from Gauss-Seidel method linearly:

$$x_{new} = \alpha x^{p+1} + \left(1 - \alpha\right) x^p. \tag{2.48}$$

where $0 < \alpha < 2$. The value of α can be determined from mathematical equations and trial-and-error. Typically, α is set to 1.8.

Additionally, Jacobi method is also discussed in this thesis. The iteration equation is (Saad, 2003):

$$x^{p+1} = D^{-1}\left(b - \left(L + U\right) x^p\right). \tag{2.49}$$

Similarly, by introducing a weighting coefficient ω, weighed Jacobi method can be derived:

$$x^{p+1} = \omega D^{-1} \left(b - (L + U)\, x^{p}\right) + (1 - \omega)\, x^{p}. \tag{2.50}$$

2.3 Standard Multigrid Method

Iterative methods work on the default mesh grid generated at the beginning of the simulator. Contrary to this, the multigrid method works on not only the default mesh grid, but also on coarser mesh grids. To better illustrate the essence of Multigrid method, two-grid method, which works on fine grid and coarse grid, is discussed here. Multigrid method can be easily achieved by recursively calling the two-grid method. Figure 2.3 shows a fine grid v^h, which is generated at the beginning of a program to

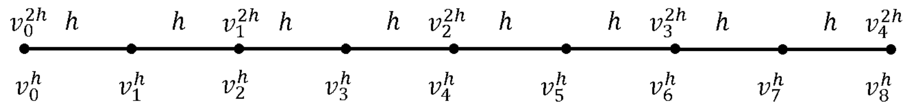


Figure 2.3: Fine and Coarse Grids for Two-grid Methods.

discretize the problem, and a coarse grid v^{2h}, which is generated for two-grid method. v_0^h, v_8^h, v_0^{2h}, and v_4^{2h} are Dirichlet boundaries. The linear matrix of this system can be expressed:

$$A^h u^h = f^h. \tag{2.51}$$

where u is the accurate solution and v means approximate solution.

The first step of two-grid method is, using an iterative method (usually weighted Jacobi method), to get an approximate solution v^h for the linear matrix system. This step is called relaxation process. Typically, the relaxation is repeated three times to get a good approximation. Then, the residual on fine mesh grid is calculated as follows:

$$r^h = f^h - A^h v^h. \tag{2.52}$$

In order to use coarse grid, a restriction step is necessary to transfer the residual from fine grid to coarse grid:

$$r^{2h} = Rr^h. \tag{2.53}$$

In the above expression R is the restriction operator.

Error e is given by the difference between accurate solution and approximate solution:

$$e = u - v. \tag{2.54}$$

If the solution is unique, residual r is zero if and only if error e is zero (Briggs *et al.*, 2000). Thus, it is easy to derive the equation for error and residual for such a linear system on coarse grid:

$$A^{2h}e^{2h} = r^{2h}. \tag{2.55}$$

If the coarse grid has very small number of points, direct methods can be used to calculate the exact solution of error e^{2h}. Otherwise, iterative methods are used to calculate an approximate solution for e^{2h}. By mapping the error on coarse grid back to fine grid, the correction to the approximate solution v^h can be calculated:

$$e^h = Ie^{2h}, \tag{2.56}$$

where I is the interpolation operator. Then,

$$v^h_{new} = v^h_{old} + e^h. \tag{2.57}$$

If the convergence criterion is not met, the two-grid solver iterates the above process until the solution is accurate enough.

For this two grids system, linear interpolation is used to map the error from coarse grid back to fine grid. Because the mesh spacing between each point is the same, each grid point has the same weight in the interpolation:

$$I = \frac{1}{2}\begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.58}$$

Restriction operator R is calculated once interpolation operator I is obtained. In 2D case R is calculated as follows:

$$R = \frac{1}{2}I^T, \tag{2.59}$$

The last unknown variable is the matrix operator A^{2h} on coarse grid. There are two approaches to calculate this matrix. One is to discretize the Poisson's equation on coarse mesh. The other is to use Galerkin principle (Strang, 2007):

$$A^{2h} = RA^hI, \tag{2.60}$$

The second method has no requirement for the shape of the coarse grid, making it suitable for both standard and Algebraic Multigrid which will be discussed later.

Figure 2.4 schematically shows the two-grid method. By adding coarser mesh grids and recursively calling two-grid method to replace the direct solver, Multigrid solver can be easily built. Figure 2.5 shows the schematics of a Multigrid solver consisting of five mesh layers.

One V-cycle is considered as a iteration in both two-grid and Multigrid methods, though iterative methods are performed multiple times within one V-cycle.

A 2D MESFET device is used to test the performance of SOR method and standard Multigrid method. Figure 2.6 is a simple 2D MESFET device structure used to

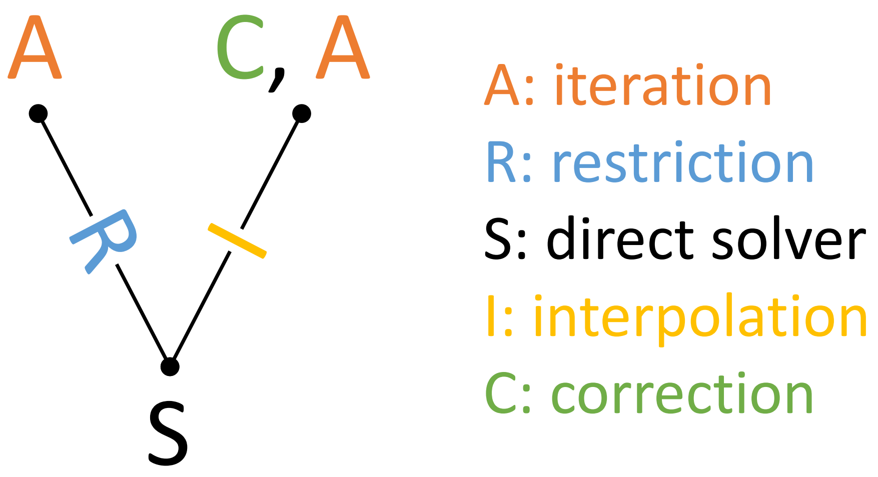


Figure 2.4: Schematic Description of Two-grid Method.

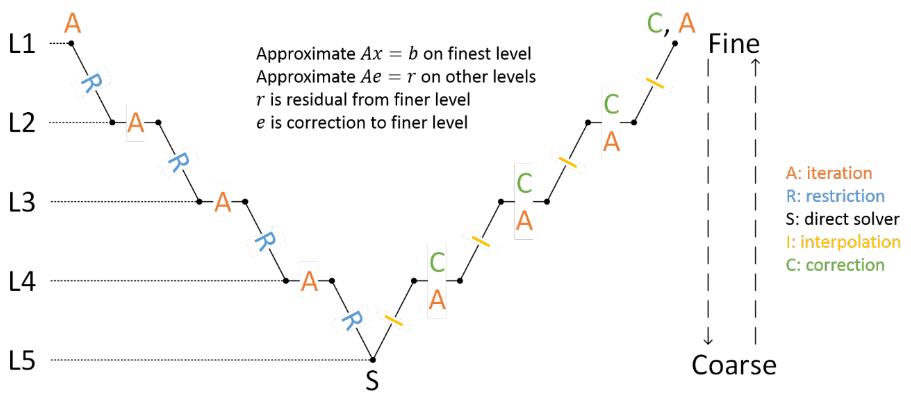

L1
L2
L3
L4
L5
A
R
A
R
A
R
A
R
S
I
C
A
I
C
A
I
C
A
I
C
A
I
C, A
Approximate $Ax = b$ on finest level
Approximate $Ae = r$ on other levels
r is residual from finer level
e is correction to finer level
Fine
Coarse
A: iteration
R: restriction
S: direct solver
I: interpolation
C: correction

Figure 2.5: Schematic Description of Multigrid Method.

calculate the Poisson's equation in equilibrium condition. Nehalem computer node in ASU Advanced Computing Center is used for this benchmark. Figure 2.7 shows the time cost of Multigrid method is linear to the number to total grid points, while the time cost of SOR increases fast. These results match well with the theoretical analysis that the complexity of SOR method is $O(n^{1.5})$, and the complexity of Multigrid method is $O(n)$, where n is the total number of grid points (Demmel, 1997).

A few more numerical experiments are conducted to explain why Multigrid, iterating on multiple mesh grids, has better performance than SOR method. The first

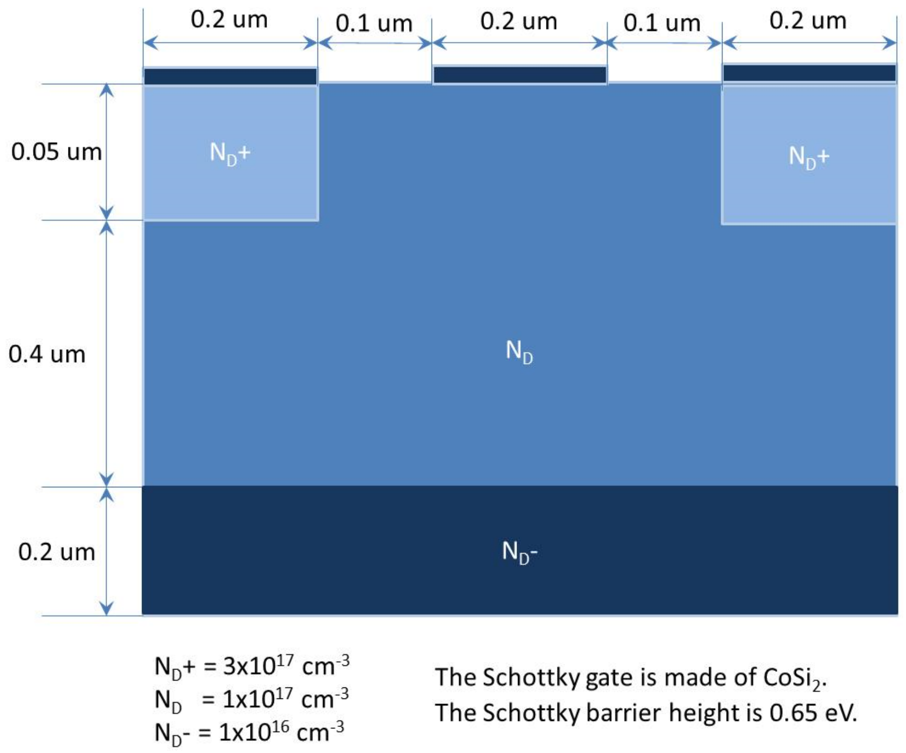

0.2 um
0.1 um
0.2 um
0.1 um
0.2 um
0.05 um
N_D+
N_D+
0.4 um
N_D
0.2 um
N_D-
$N_D+ = 3x10^{17}$ cm^{-3}
$N_D = 1x10^{17}$ cm^{-3}
$N_D- = 1x10^{16}$ cm^{-3}
The Schottky gate is made of CoSi$_2$.
The Schottky barrier height is 0.65 eV.

Figure 2.6: Structure of 2D MESFET Device from the Homework of Prof. Vasileska's EEE 598 Course.

numerical system is:

$$\begin{aligned} f''(x) &= -\pi^2 k^2 \sin(k\pi x), \\ f(0) = 0, f(1) &= 0, \\ f(x)|_{t=0} &= 0, \\ 0 < x &< 1, \\ k &= 1, 2, 4, 8, 16, 24, 64. \end{aligned} \tag{2.61}$$

The mesh grid has $2^8 + 1 = 257$ points, which are uniformly distributed. Norm of error is plotted against number of iterations for each case. Weighted Jacobi method is used in this experiment.

Error can be calculated by comparing the approximate solution and the analytical solution which is hard to obtain for real application. So, in numerical experiments,

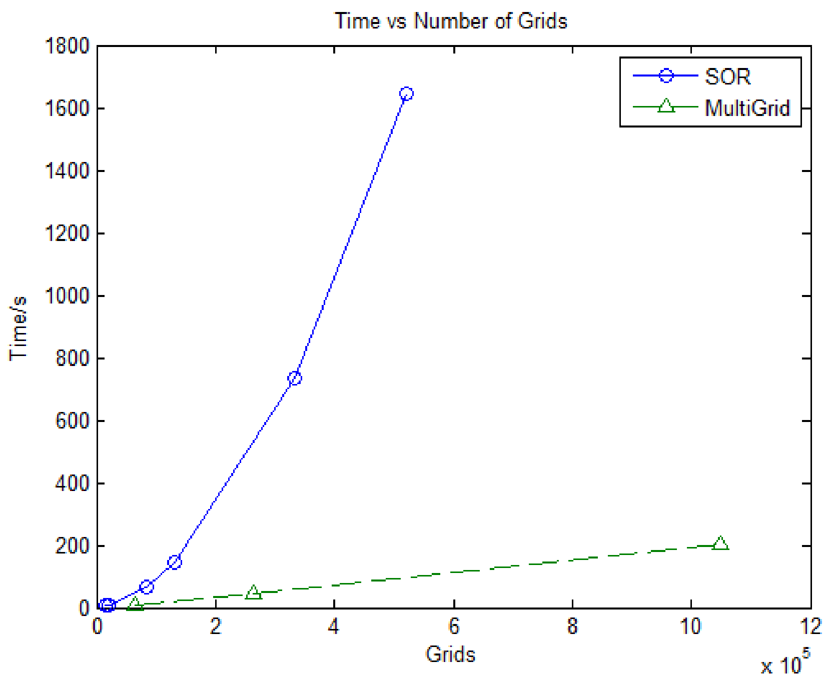


Figure 2.7: Performance of SOR and Multigrid Methods.

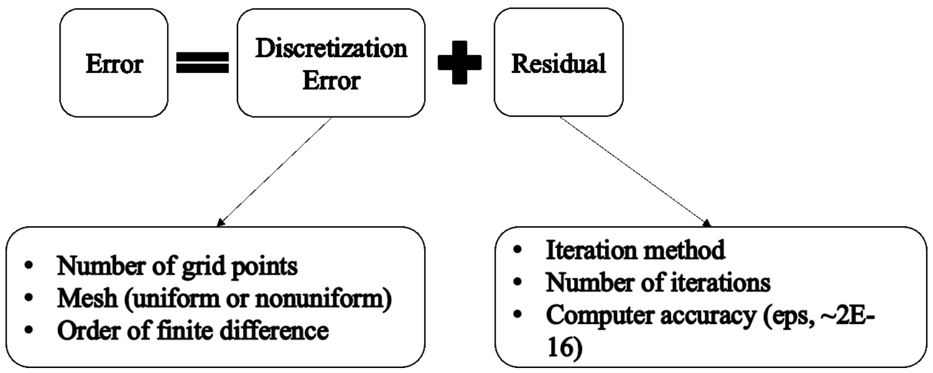


Figure 2.8: Affecting Factors for the Components of Error.

both norm of error and residual are recorded to analyze the performance of different numerical methods. In real applications, only residual is recorded to determine convergence and the performance of numerical solvers.

One affecting factor for residual term in figure 2.8 is computer accuracy, aka eps, which is the spacing between floating point numbers. This research is conducted with Matlab. The value of eps is 2.2204E-16. Therefore, it is reasonable to expect the minimum residual to be on the order of 1E-16. The minimum norm of residual can be larger taking the number of grid points into consideration.

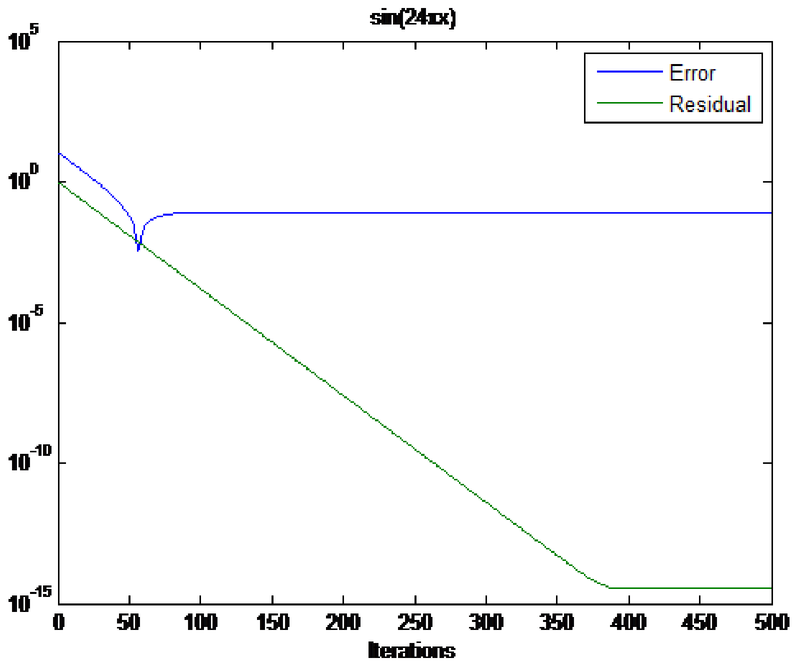


Figure 2.9: Relation between Error and Residual.

Residual in figure 2.9 behaves exactly as what is predicted above. Its norm decreases to about 1E-14, which is 2 orders higher than the value of eps because of these 257 mesh grid points. The constant value of error is much larger than the value of residual. The discretization error contributes the most to the overall error. Because uniform mesh is employed in this experiment, the mesh spacing is not small enough near peaks of this sine wave function, thus creating a large amount of discretization error. The necessity of non-uniform mesh to balance the accuracy and computation load is justified here. To properly calculate the restriction and interpolation operators, mesh spacing around each grid point should be taken into the calculation instead of a constant mesh spacing value. The spike in error curve may come from the coherence of Matlab's internal numerical solver, the discretization error, and the residual.

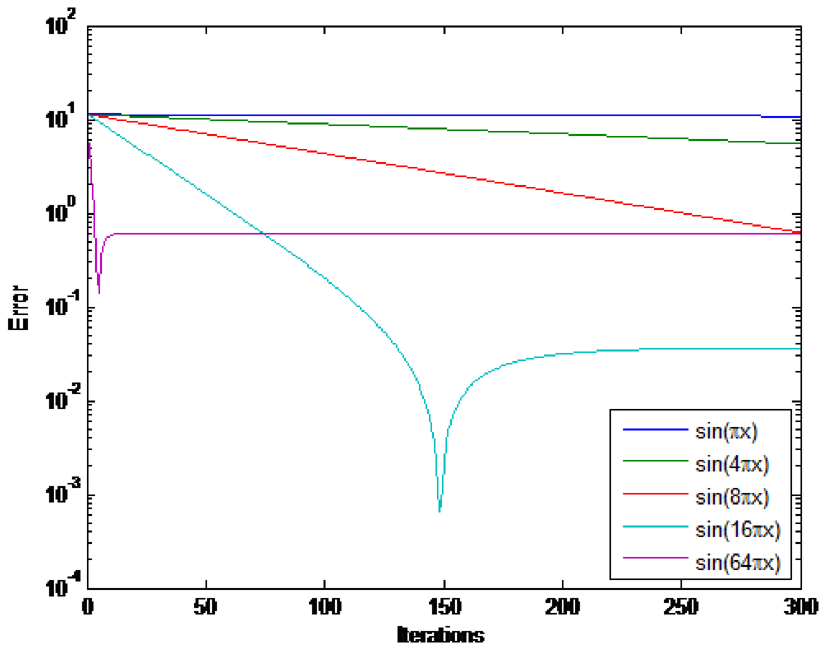


Figure 2.10: Norm of Error and Number of Iterations.

In figure 2.10, from $\sin(\pi x)$ to $\sin(16\pi x)$, the reduction of error gets faster when the wave number k gets larger. $\sin(16\pi x)$ case is the best. In $\sin(64\pi x)$ case, although error reduces even faster, the steady-state values increase as well because of the discretization error from insufficient number of mesh grid points comparing to its high frequency mode. This figure leads to the conclusion that an iterative method has the highest effectiveness to a certain error frequency, while the effectiveness drops fast with reducing error frequency. So the high performance of Multigrid method comes from the fact that error is iteratively relaxed on multiple meshes with different mesh spacing, which ensures that the error of various frequencies is properly reduced.

The two-grid method includes two mesh levels. The fine level has $2^8 + 1 = 257$ points, while the coarse level has $2^7 + 1 = 129$ points. In figure 2.11, because the coarse level suits well with $\sin(8\pi x)$ function, the residual of two-grid method reduces

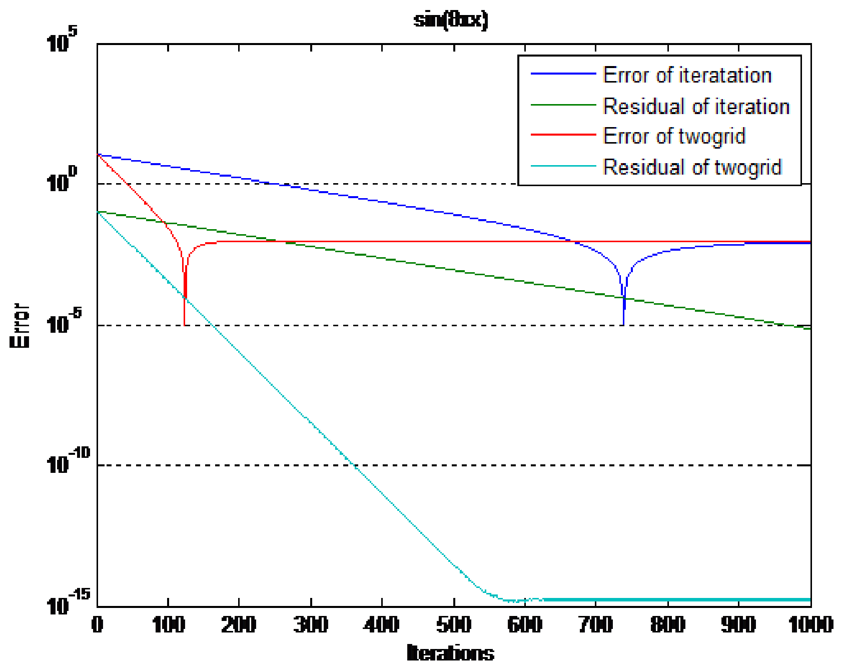


Figure 2.11: Performance of Weighted Jacobi and Two-grid Methods in $8\pi x$ Case.

fast to its minimal. Error in this figure also needs to be discussed. Although error of two-grid method reduces fast, both methods have the same constant value of error and the same spike because the residual component is properly reduced leaving the same discretization component.

Unlike the good performance in figure 2.11, the two-grid method in figure 2.12 does not display a substantial advantage though it is still better than weighted Jacobi method. This is caused by the fact that $\sin(2\pi x)$ is a small frequency for both 257 and 129 points mesh. To increase the performance, more mesh levels are necessary to handle such low frequency error.

For the purpose of adding error of all frequencies together to test overall perfor-

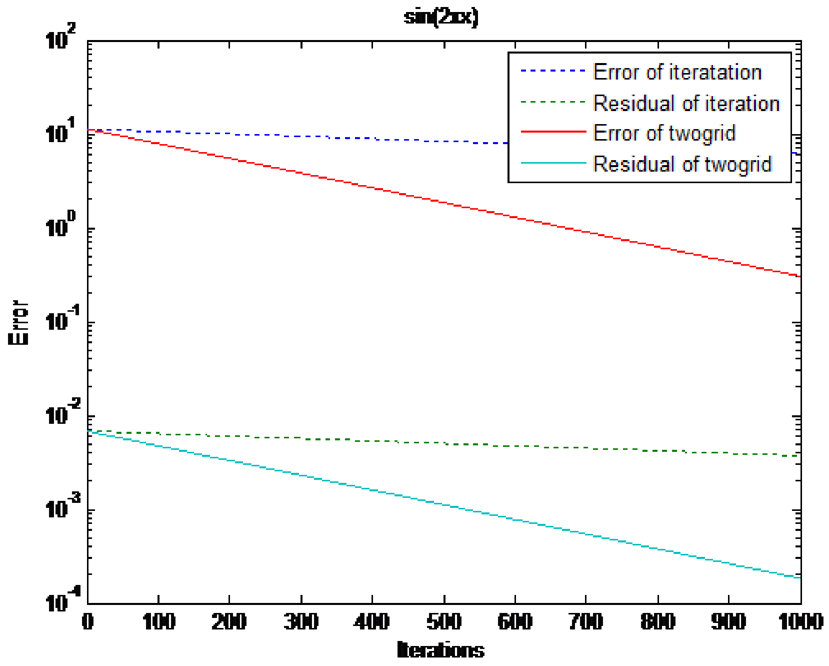


Figure 2.12: Performance of Weighted Jacobi and Two-grid Methods in $2\pi x$ Case.

mance of SOR and Multigrid methods, the second numerical experiment is created:

$$\begin{aligned} f''(x) &= -\pi^2 \sum k^2 \sin{(k\pi x)}, \\ f(0) = 0, f(1) &= 0, \\ f(x)|_{t=0} &= 0, \\ 0 < x &< 1, \\ k &= 1, 2, 4, 8. \end{aligned} \tag{2.62}$$

Similarly, the mesh has $2^8 + 1 = 257$ points, which are uniformly distributed.

Figure 2.13 and figure 2.14 schematically show the error components and total error in this experiment. Compared with the number of mesh points, the highest error frequency included in this experiment is small for iterative methods. Thus a very bad performance of weighted Jacobi method should be expected. The performance of weighted Jacobi method in figure 2.15 is exactly the same as what is predicted above. Neither the norm of error nor the norm of residual is substantially reduced.

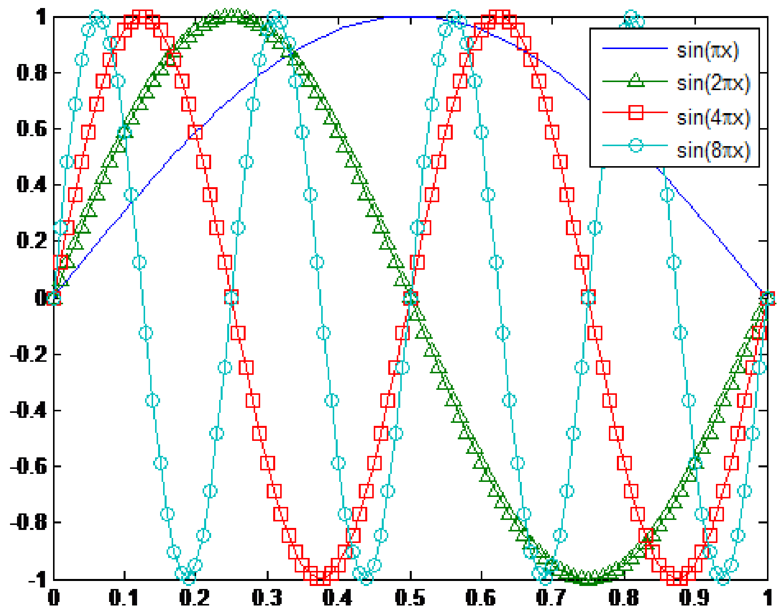


Figure 2.13: Plot of Error Components.

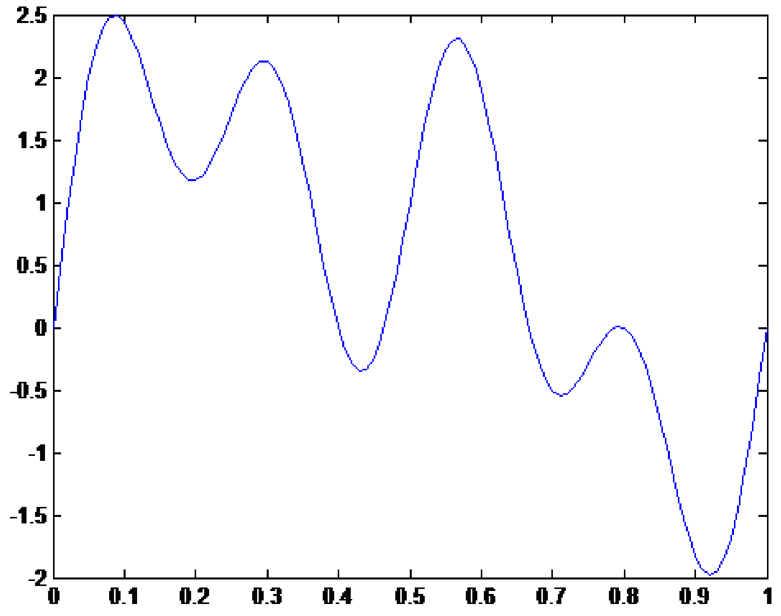


Figure 2.14: Sum of All Error Components.

However, Multigrid method shows very good performance. Error reduces to constant for about only three iterations, while residual reduces to a very small constant for

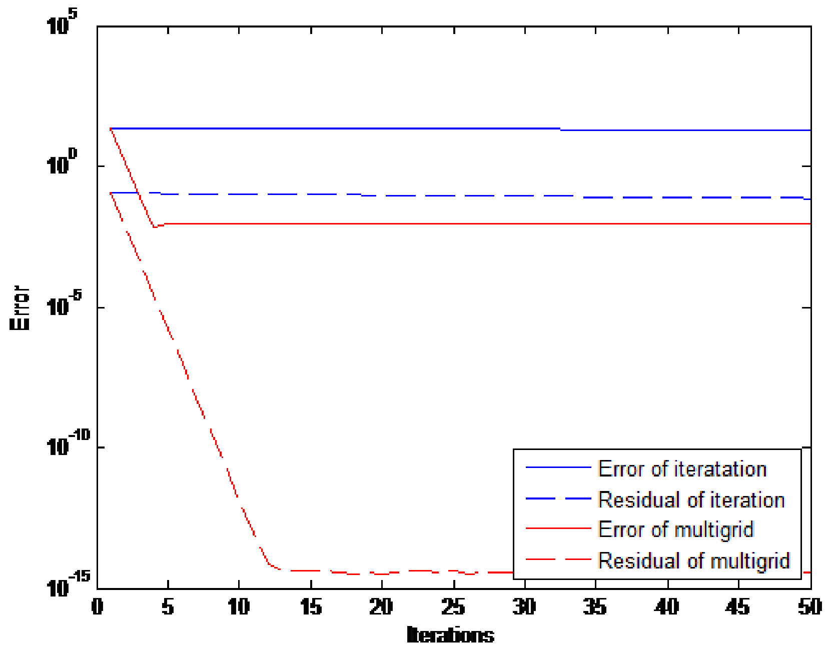
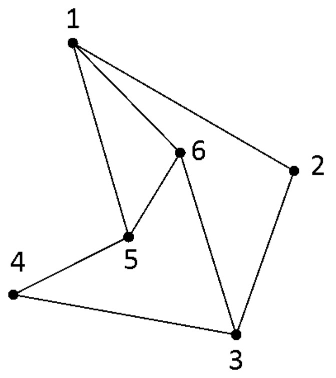


Figure 2.15: Performance of Weighted Jacobi and Multigrid Methods.

about thirteen iterations.

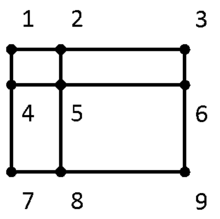
2.4 Algebraic Multigrid Method

Multigrid method shows large advantages over SOR method. Then what is the reason to develop Algebraic Multigrid? As mentioned in the previous section, Multigrid method is based on constructed meshes. Interpolation and restriction operators are based on rectangular mesh as well. This posts difficulties to the setup step of Multigrid method when problem geometry is complex. Moreover, standard Multigrid requires $2^N + 1$ grid points in each direction, which commonly results in either too large mesh spacing or too many mesh grid points. Besides, the interpolation operator is calculated by linear interpolation for 1D or bilinear interpolation for 2D, which means only mesh spacing is taken into consideration. Although non-uniform mesh spacing can be reflected in the interpolation operators, spatially varying dielectric constant is ignored. Therefore, developing Algebraic Multigrid method becomes important for real applications to overcome these problems.



$$A = \begin{bmatrix} a_{11} & a_{12} & & & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & & & \\ & a_{32} & a_{33} & a_{34} & & a_{36} \\ & & a_{43} & a_{44} & a_{45} & \\ a_{51} & & & a_{54} & a_{55} & a_{56} \\ a_{61} & & a_{63} & & a_{65} & a_{66} \end{bmatrix}$$

Figure 2.16: Coefficient Matrix of an Unconstructed Mesh Grid.



$$A = \begin{bmatrix} a_{11} & a_{12} & & a_{14} & & & & & \\ a_{21} & a_{22} & a_{23} & & a_{25} & & & & \\ & a_{32} & a_{33} & & & a_{36} & & & \\ a_{41} & & & a_{44} & a_{45} & & a_{47} & & \\ & a_{52} & & a_{54} & a_{55} & a_{56} & & a_{58} & \\ & & a_{63} & & a_{65} & a_{66} & & & a_{69} \\ & & & a_{74} & & & a_{77} & a_{78} & \\ & & & & a_{85} & & a_{87} & a_{88} & a_{89} \\ & & & & & a_{96} & & a_{98} & a_{99} \end{bmatrix}$$

Figure 2.17: Coefficient Matrix of a Constructed Mesh Grid.

Algebraic Multigrid has no requirement for the geometry. It is able to work with unconstructed mesh grid as shown in figure 2.16. A mesh grid is a bidirected graph, making the coefficient matrix A symmetric. Figure 2.17 shows a constructed rectangular mesh grid, which is typically used in semiconductor device simulators. As long as the constructed mesh grid is rectangular, 5-point stencil finite-volume method can still be applied to discretize the Poisson's equation. Despite the difference between the unconstructed mesh grid in figure 2.16 and the constructed grid in figure 2.17, two coefficient matrices show the same shape and symmetry. Thus, Algebraic Multigrid method works with these two kinds of mesh grids because only a discretized matrix system $Ax = b$ is required. Figure 2.18 shows the flowchart of the setup procedure

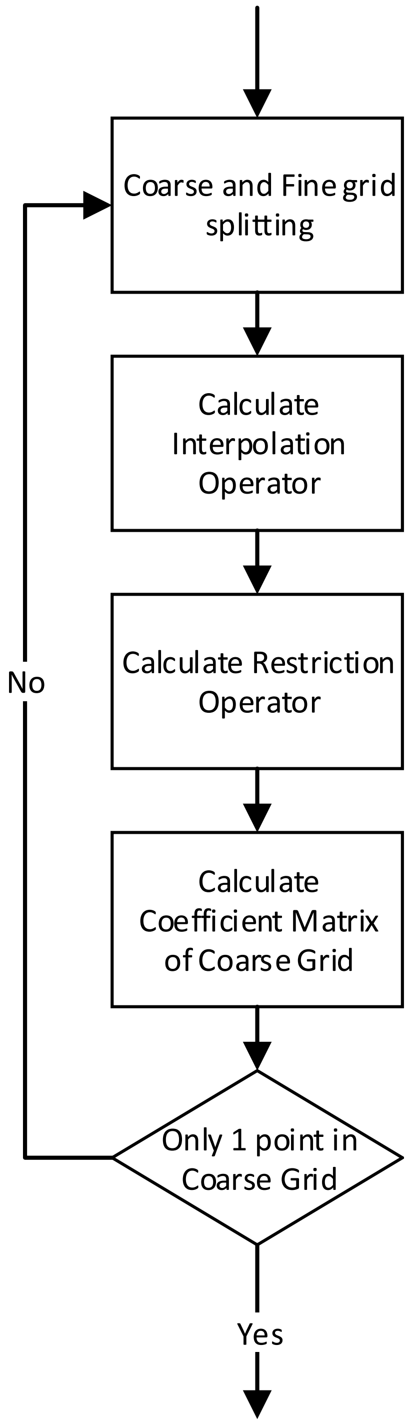

Coarse and Fine grid splitting
Calculate Interpolation Operator
Calculate Restriction Operator
Calculate Coefficient Matrix of Coarse Grid
Only 1 point in Coarse Grid
No
Yes

Figure 2.18: Setup Steps of Algebraic Multigrid Method.

in Algebraic Multigrid solver. Coarser grids are automatically selected based on the algebraic property of the coefficient matrix A. Then, based on coarse and fine grid split information, interpolation operator is calculated from the coefficient matrix A. Restriction operator is the transpose of interpolation operator. The coefficient matrix for coarser mesh grid can be calculated by using the Galerkin principle. Thus, one repeats these steps until the coarsest mesh grid has only one point.

Coarse and fine grid splitting is called coarsening as well. Unlike standard Multigrid method which requires manual selection, Algebraic Multigrid method uses the information stored in the coefficient matrix A to determine which points should be selected as coarse grid. The aggressive coarsening method is shown in figure 2.19.

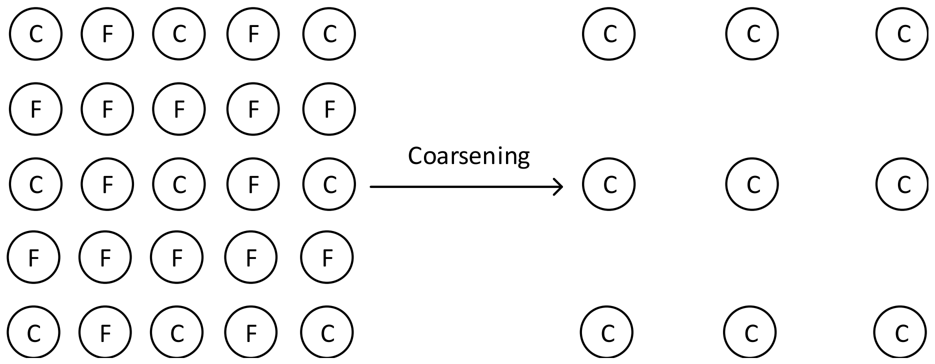


Figure 2.19: Aggressive Coarsening in 2D Case.

This method highly depends on device geometry and mesh grid and requires $2^N + 1$ points in each direction. Thus, it is not used in this Algebraic Multigrid because the aim is to provide accommodation for complex geometry and loose requirement for mesh grid. The other coarsening scheme is standard coarsening, which is used in this Algebraic Multigrid method. The coarsening process is slower than aggressive coarsening because less fine grid points are eliminated in each coarser level, resulting in more levels. Compared with the effort to generate more levels and consumption of more computer memory space, it is beneficial to use this standard coarsening in Algebraic Multigrid method because of no requirement for geometry, automated gen-

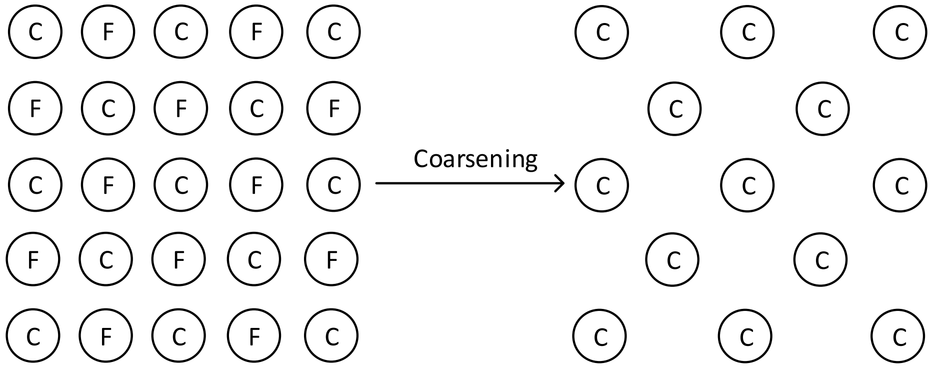


Figure 2.20: Standard Coarsening in 2D Case.

eration of coarse levels, and exact coupling with 5-point stencil. Each fine grid point has four neighboring coarse grid points (up, down, left, and right), which are used to interpolate the fine grid point when values are transferred from coarse grid back to fine grid.

The flowchart of standard coarsening method is shown in figure 2.21 (Trottenberg *et al.*, 2001). F represents the set of fine points. C represents the set of coarse grid points. U represents the set of undecided points. Ω is the set of all grid points on this level. These four sets have the following relation:

$$\Omega = C + F + U. \tag{2.63}$$

To better explain the meaning of S_i^T, a few more sets need to be discussed. N_i is a set of neighboring points of grid point i. It is determined by the following equation:

$$N_i = \{j \in \Omega : j \neq i, a_{ij} \neq 0\}. \tag{2.64}$$

The diagonal elements of coefficient matrix A are assumed to be positive, while other off-diagonal elements can have any sign. Therefore, it is important to define a relation called strongly negatively coupled:

$$-a_{ij} \geq \varepsilon_t hr \max_{a_{ik}<0} |a_{ik}|. \tag{2.65}$$

In Eq. (2.65) ε_{thr} is a constant between 0 and 1 used to determine the strength of coupling. Then a new set S_i can be created to represent neighboring points of i. i is strongly negatively coupled to:

$$S_i = \{j \in N_i : i \text{ is strongly negatively coupled to } j\}. \tag{2.66}$$

By transposing the set S_i, a new set S_i^T can be created:

$$S_i^T = \{j \in \Omega : i \in S_j\}. \tag{2.67}$$

It contains all the points that are strongly coupled to point i.

To properly determine the sequence of adding a point to either coarse set or fine set is based on the value of λ_i, the importance of a point (Stüben and Ruge, 1987). A point that has a large number of undecided neighboring and fine points is given high priority to be processed:

$$\lambda_i = |S_i^T \cap U| + 2|S_i^T \cap F|. \tag{2.68}$$

$|S|$ means the number of elements in a set S. The coarsening of a mesh grid consisting of 11 by 19 grid points is shown in Figure 22. Standard coarsening pattern can be observed at fine mesh levels, while a deviation from the standard pattern appears at coarse levels due to very few available grid points.

There are quite a few methods to calculate the interpolation operator (Stüben and Ruge, 1987). The restriction operator is typically calculated by transposing the interpolation operator. As for the coefficient operator for coarser level, both direct method and Galerkin method can be used. In this research, direct method, which uses 2D 5-point stencil, is used to calculate the coefficient operator for the finest level because a device simulator requires constructed rectangular mesh grids. However, the finer grid levels are not constructed nor rectangular. Therefore, Galerkin method is

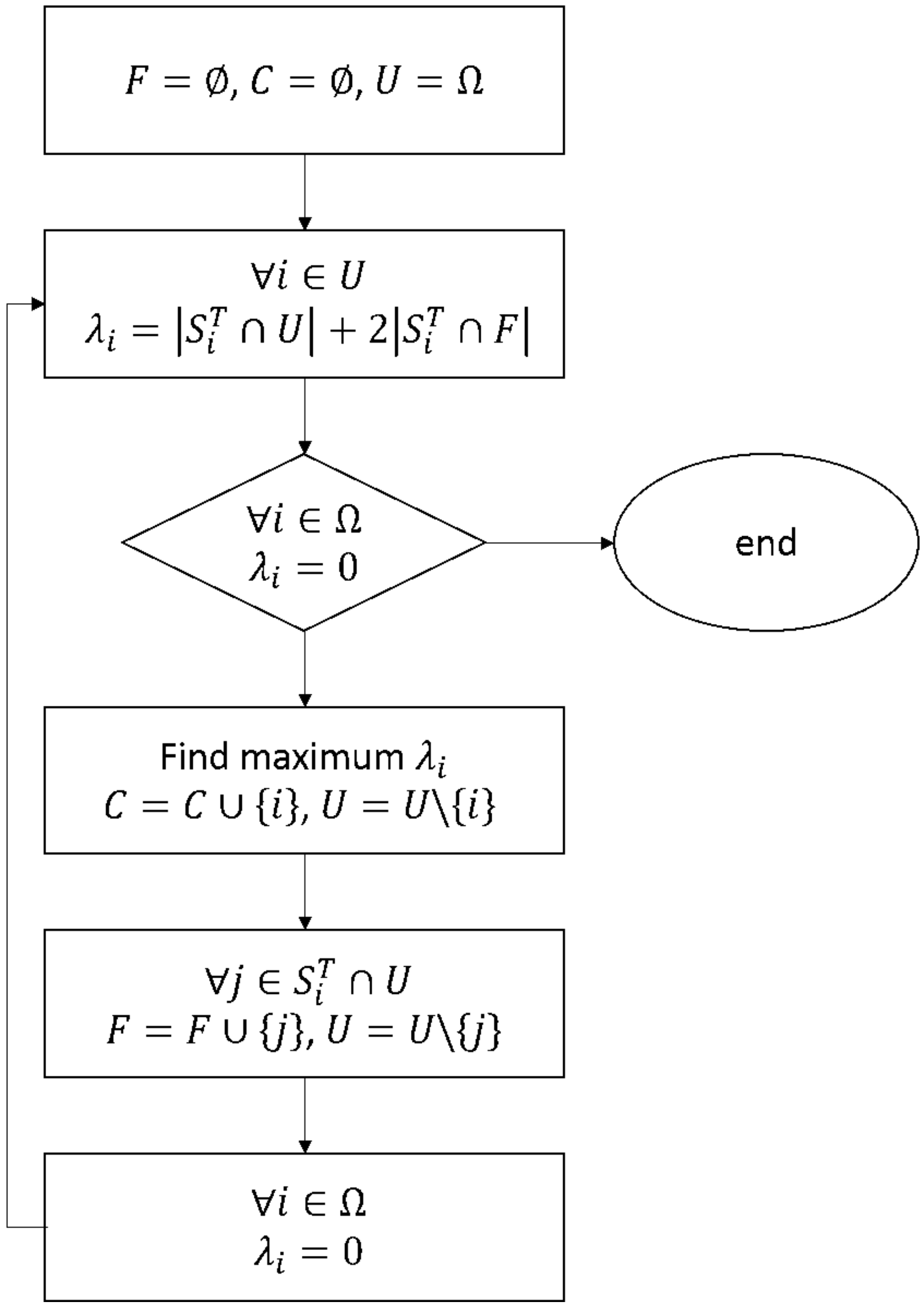

$F = \emptyset, C = \emptyset, U = \Omega$
$\forall i \in U$
$\lambda_i = |S_i^T \cap U| + 2|S_i^T \cap F|$
$\forall i \in \Omega$
$\lambda_i = 0$
end
Find maximum λ_i
$C = C \cup \{i\}, U = U \backslash \{i\}$
$\forall j \in S_i^T \cap U$
$F = F \cup \{j\}, U = U \backslash \{j\}$
$\forall i \in \Omega$
$\lambda_i = 0$

Figure 2.21: Flowchart of Standard Coarsening.

used because it calculates the coefficient matrix by multiplying interpolation operator, restriction operator, and fine grid coefficient matrix algebraically, which requires no information about mesh geometry. The remaining problem is how to accurately calculate the interpolation operator. Selecting standard coarsening ensures the core

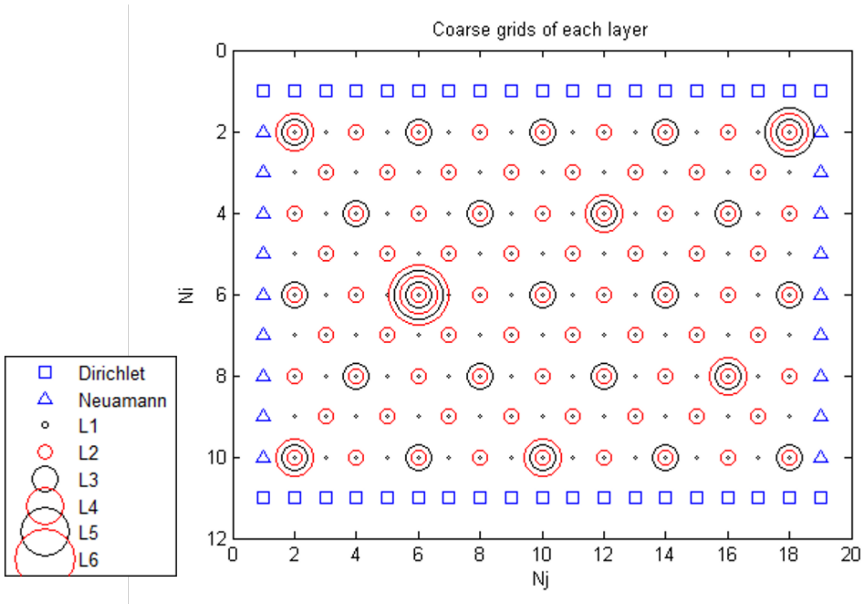


Figure 2.22: Coarsening of a 11 by 19 Mesh Grid.

requirement that each fine grid point must have at least one connection to neighboring coarse grid points. Then, there is no need to use complicated methods, like standard interpolation and multi-pass interpolation, because of the satisfaction of the core requirement.

A coarse grid in a fine level is a coarse grid in the coarse grid with the same coordinates. Thus, its value can be transferred from coarse grid to fine grid without interpolation. A find grid in a fine level does not have corresponding point in the coarse level. Its value can be interpolated from its coarse neighboring points with strong connections. The interpolation process can be expressed as following (Briggs *et al.*, 2000):

$$e_i = \begin{cases} e_i & i \in C \\ \sum_{j \in P_i} \omega_{ij} e_j & i \in F \end{cases}. \tag{2.69}$$

P_i is a set of points used for interpolation of point i. It is defined as C_i^s. C_i^s is the set of coarse points that i is strongly negatively coupled to:

$$C_i^s = C \cap S_i. \tag{2.70}$$

ω_{ij} is the interpolation weight that needs to be carefully treated. Before discussing the details of interpolation weight, a few new sets need to be introduced:

$$C_i = C \cap N_i, \tag{2.71}$$

$$F_i = F \cap N_i, \tag{2.72}$$

$$F_i^s = F \cap S_i. \tag{2.73}$$

The simplest method is ignoring the value of the element in the set of P_i. The interpolation weight ω_{ij} is set to the reciprocal of the number of elements:

$$\omega_{ij} \equiv \frac{1}{|P_i|}. \tag{2.74}$$

Figure 2.23 shows the constant interpolation weight schematically. Although this

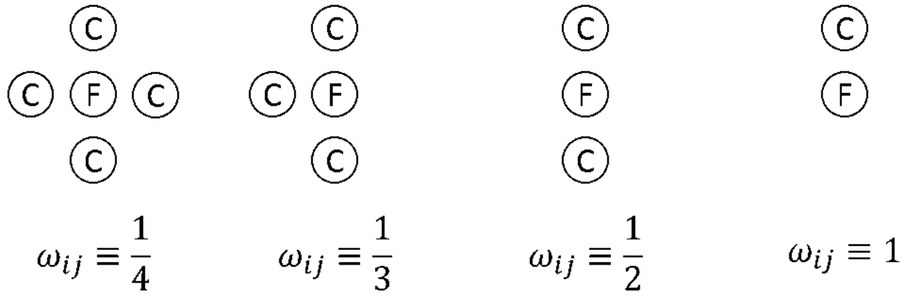

C
C F C
C
$\omega_{ij} \equiv \frac{1}{4}$
C
C F
C
$\omega_{ij} \equiv \frac{1}{3}$
C
F
C
$\omega_{ij} \equiv \frac{1}{2}$
C
F
$\omega_{ij} \equiv 1$

Figure 2.23: Scheme of Constant Interpolation Weight.

method is quite simple and easy to be implemented, the convergence of the method is not satisfactory due to the inaccurate calculation of interpolation weight.

To improve the accuracy of interpolation, the value of each element in P_i should be taken into consideration. A point with stronger connection is given larger weight by the following equation:

$$\omega_{ik} = -\frac{a_{ik}}{\sum_{j \notin P_i} a_{ij}}. \tag{2.75}$$

The summation of diagonal element and elements not used for interpolation acts as the denominator. A variable s_i needs to be introduced to discuss this interpolation method:

$$s_i = \sum_j a_{ij}. \tag{2.76}$$

In a symmetric matrix, s_i equals to 0. However, the method to implicitly deal with Dirichlet boundary conditions makes corresponding rows in coefficient matrix A strongly diagonally dominant by moving an off-diagonal element to the forcing term. Therefore, this method is not suitable for Dirichlet boundary conditions, though it can be used for Neumann boundary conditions and inner points.

A third method can deal with all kinds of points, which calculates positive and negative points, respectively. In this method, the interpolation set is $P_i = C_i^s$:

$$\omega_{ik} = \begin{cases} \frac{\alpha_i a_{ik}}{a_{ii}} & k \in P_i^- \\ \frac{\beta_i a_{ik}}{a_{ii}} & k \in P_i^+ \end{cases}, \tag{2.77}$$

$$\alpha_i = \frac{\sum_{j \in N_i} a_{ij}^-}{\sum_{k \in P_i} a_{ik}^-}, \tag{2.78}$$

$$\beta_i = \frac{\sum_{j \in N_i} a_{ij}^+}{\sum_{k \in P_i} a_{ik}^+}, \tag{2.79}$$

where

$$P_i^+ = \{j \in P_i : a_{ij} > 0\}, \tag{2.80}$$

$$P_i^- = \{j \in P_i : a_{ij} < 0\}, \tag{2.81}$$

$$a_{ij}^+ = \begin{cases} a_{ij} & a_{ij} > 0 \\ 0 & ka_{ij} \leq 0 \end{cases}, \tag{2.82}$$

$$a_{ij}^- = \begin{cases} 0 & a_{ij} > 0 \\ a_{ij} & ka_{ij} \leq 0 \end{cases}. \tag{2.83}$$

The coefficient matrix of the finest level contains non-positive off-diagonal elements and positive diagonal elements, which leads to $P_i^+ = \emptyset$. The denominator of β_i

becomes zero. In this case, β_i is set to zero and its numerator is added to the diagonal element a_{ii} to take the effect of positive elements not belonging to P_i into consideration.

Algebraic Multigrid method replaces the manual setup process in standard Multigrid method with automated coarsening and calculation. V cycles described in standard Multigrid method can be used to solve the problem.

A simple 2D numerical experiment is conducted to test the performance of Algebraic Multigrid method.

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -4\pi^2(\cos(2\pi x) + \sin(2\pi y)), \\ 0 < x < 1, \\ 0 < y < 2, \\ u(x,0) = \cos(2\pi x), \\ u(x,1) = \cos(2\pi x), \\ u(0,y) = 1 + \sin(2\pi y), \\ u(2,y) = 1 + \sin(2\pi y). \end{aligned} \tag{2.84}$$

The analytical solution is

$$u(x,y) = \cos(2\pi x) + \sin(2\pi y). \tag{2.85}$$

Three different mesh grids are tested. They are 11 by 21 mesh grid, 41 by 81 mesh grid, and 81 by 161 mesh grid. The numerical result of 81 by 161 mesh grid is shown in figure 2.24.

In figure 2.25, the convergence of the three different mesh grids are plotted against the number of iterations. These three cases use almost the same number of iterations to reach a constant residual value, which demonstrates the robustness of Multigrid

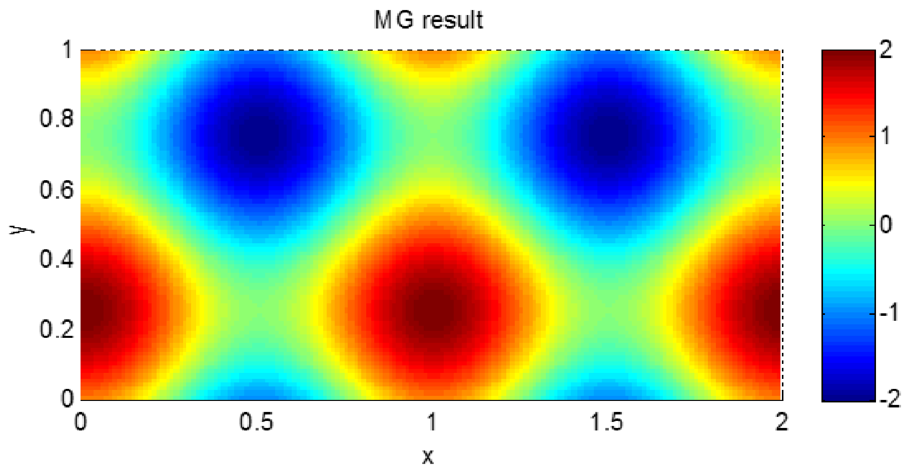


Figure 2.24: Result of AMG Numerical Experiment.

method because its performance is independent from the number of mesh grid points. As for the difference among these constant values, this is caused by the way used to calculate the 2-dimentional Euclidean norm for a vector $\mathbf{x} = (x_1, x_2, x_3, \cdots, x_n)$:

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{n} x_i^2}. \tag{2.86}$$

The norm is related to the number of elements in the vector. More grid points result in higher constant value.

Uniform mesh is used in each direction in the above tests. An additional test of 81 by 161 mesh grid is performed with random mesh spacing. Assume the uniform mesh spacing is h in one direction. The random non-uniform mesh spacing is

$$0.4h < h_{rand} < 1.6h. \tag{2.87}$$

Figure 2.26 shows the distribution of mesh spacing against its position in y direction. It can be observed that the mesh spacing is randomly selected and its distribution is random as well. In this fully random case, Algebraic Multigrid method reduces error perfectly well with convergence very close to the uniform case, which proves the effectiveness of Algebraic Multigrid method. In real applications, mesh spacing will

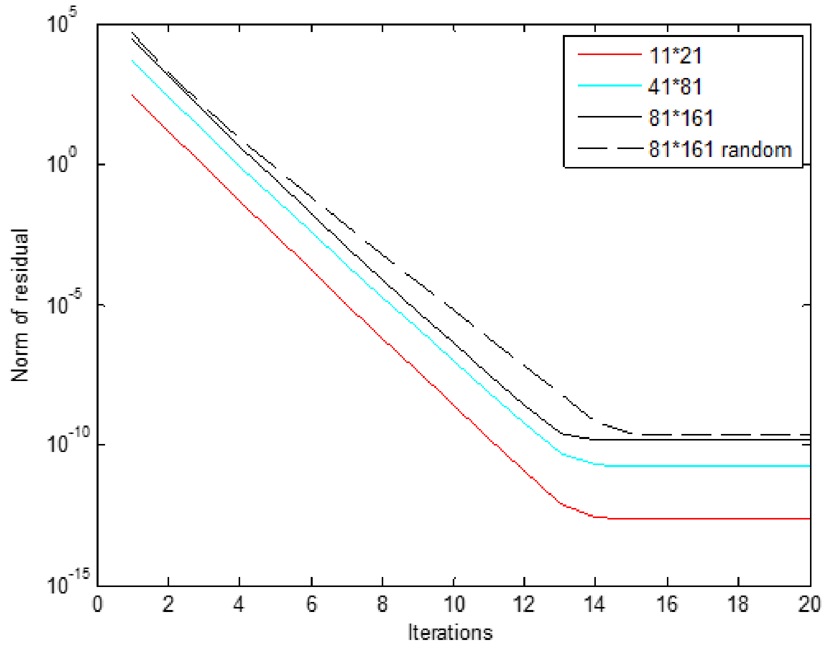


Figure 2.25: Convergence with Different Number Grid Points.

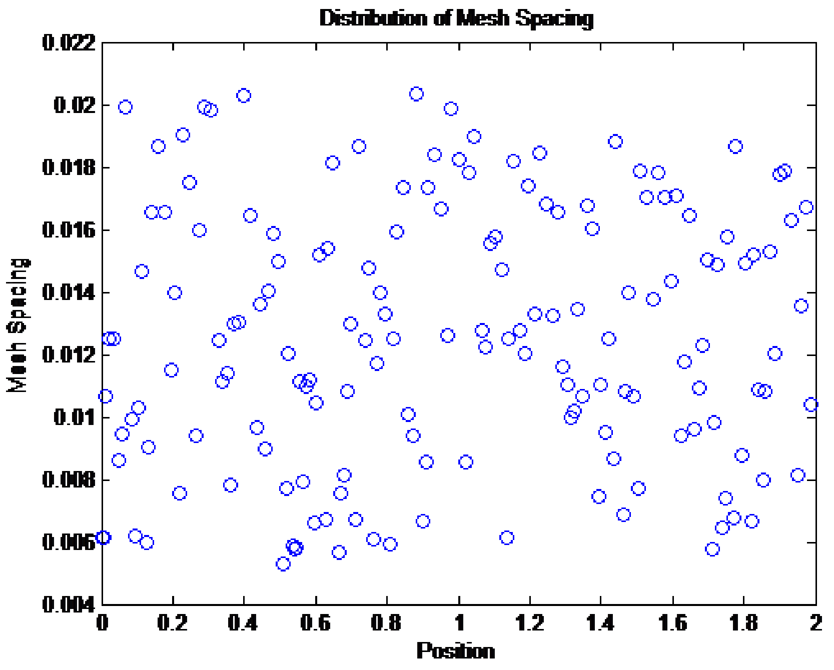


Figure 2.26: Distribution of Mesh Spacing.

not be randomly generated. Typically, it will be piece-wise constant or monotonically increasing or decreasing. Therefore, Algebraic Multigrid can be very effective.

The 81 by 161 mesh grid contains 7 mesh levels. A maximum of 2, 4, 6, and 7 levels are tested to observe the behavior of convergence. It is very clear that the convergence

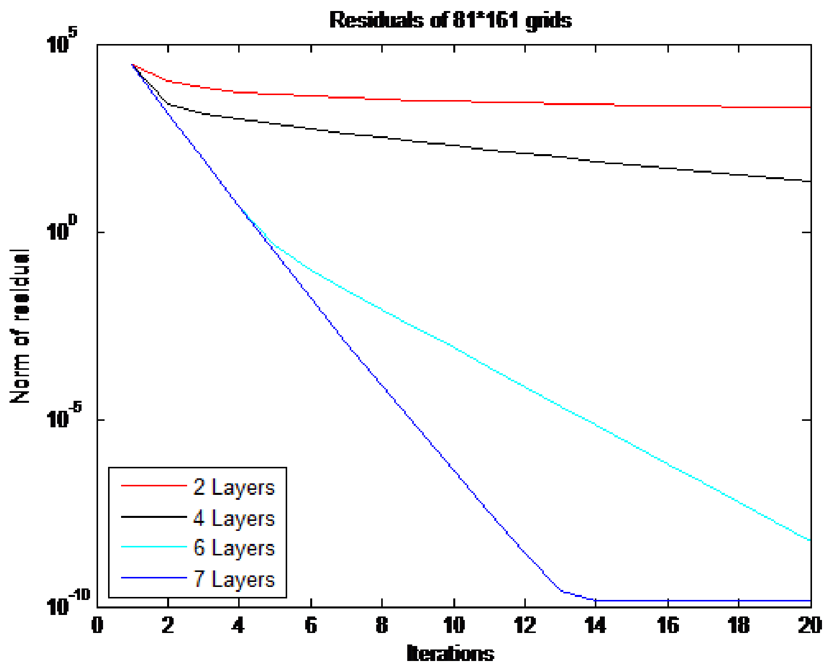


Figure 2.27: Convergence of Different Maximum Number of Levels.

is faster with more levels involved. This is another proof to the explanation why Multigrid method converges faster. Compared with the error frequency, this fine mesh grid represents a high frequency. A large number of maximum level means a coarse mesh level is used. With 7 levels taken into computation, all small error frequencies are included. Therefore, the blue line in figure 2.25 shows the best performance.

2.5 Full Approximation Scheme

The first section of this chapter discusses the linearization for the Poisson's equation. As iterative methods and standard Multigrid method can not solve nonlinear equations directly, the exponential term in Poisson's equation has to be expanded with Taylor series. This expansion leads to an update δ, which is the difference between an old and a new value of the potential. This process can be used to iteratively calculate the solution of Poisson's equation. Within one iteration, the system behaves like linear system of equation as all exponential terms use old potential value, which are treated as constants. This kind of linearization procedure works well with iterative method like SOR and standard Multigrid method. However, Algebraic Multigrid method cannot benefit from this procedure because once the new potential values become available, the property of such linear system, i.e. coefficient matrix A, changes. Correspondingly, the setup of Algebraic Multigrid method should be run again to generate coarser mesh levels, interpolation operators, restriction operators, and coefficient matrices.

To avoid this complication, Full Approximation Scheme (FAS) is utilized to solve nonlinear equation directly. The left hand side of Poisson's equation remains unchanged. 5-point stencil still applies to it. The right hand side of Poisson's equation will not be expanded using Taylor series. Then one moves the exponential terms to the left hand side:

$$\begin{aligned}
&\frac{(\varepsilon_{i+1,j}+\varepsilon_{i,j})}{x_i(x_i+x_{i-1})}\varphi_{i+1,j}+\frac{(\varepsilon_{i-1,j}+\varepsilon_{i,j})}{x_{i-1}(x_i+x_{i-1})}\varphi_{i-1,j}\\
&+\frac{(\varepsilon_{i,j+1}+\varepsilon_{i,j})}{y_j(y_j+y_{j-1})}\varphi_{i,j+1}+\frac{(\varepsilon_{i,j-1}+\varepsilon_{i,j})}{y_{j-1}(y_j+y_{j-1})}\varphi_{i,j-1}\\
&-[\frac{(\varepsilon_{i+1,j}+\varepsilon_{i,j})}{x_i(x_i+x_{i-1})}+\frac{(\varepsilon_{i-1,j}+\varepsilon_{i,j})}{x_{i-1}(x_i+x_{i-1})}\\
&+\frac{(\varepsilon_{i,j+1}+\varepsilon_{i,j})}{y_j(y_j+y_{j-1})}+\frac{(\varepsilon_{i,j-1}+\varepsilon_{i,j})}{y_{j-1}(y_j+y_{j-1})}]\varphi_{i,j}\\
&+q(N_i\exp(-\frac{\varphi_{i,j}}{V_T})-N_i\exp(\frac{\varphi_{ij}}{V_T}))\\
&=-q(dop_{i,j}). \qquad (2.88)
\end{aligned}$$

Similar normalization can be used to reduce the calculation load. Methods needed to process Dirichlet and Neumann boundary conditions remain unchanged. The generalization of this new nonlinear matrix system is as follows:

$$Ax + g(x) = b. \tag{2.89}$$

where $g(x)$ represents the nonlinear term. Now the system will not be changed while the potential values are updated each iteration. Algebraic Multigrid method is still useful to accommodate arbitrary number of grid points in each direction, generate coarser mesh levels automatically, and accurately calculate interpolation operators for non-uniform mesh. However, only linear part, i.e. matrix A, is used by Algebraic Multigrid.

Unlike the role of Algebraic Multigrid, Full Approximation Scheme replaces the relaxation V cycles in standard Multigrid, making possible that nonlinear systems are solved directly. In standard Multigrid method, the matrix system $Ax = b$ is relaxed only on the finest mesh grid level. Residuals and corrections are calculated on the rest of the levels. Full Approximation Scheme calculates the nonlinear system $Ax+g(x) = b$ on all levels together with corresponding residuals and corrections (Van Henson, 2003).

Relaxation methods like weighted Jacobi method, Gauss-Seidel method, SOR method, do not work with nonlinear system. Therefore, an iterative method that can work with nonlinear equations is necessary to relax residual in Full Approximation Scheme. The scalar format of Gauss-Seidel method is (Vasileska *et al.*, 2010)

$$x_i^{p+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{p+1} - \sum_{j=i+1}^{N} a_{ij} x_j^{p} \right). \tag{2.90}$$

Newton's method can be added to scalar format Gauss-Seidel method to deal with the nonlinear term. Define the nonlinear system as

$$F(x) = Ax + g(x) - b = 0. \tag{2.91}$$

Applying Newton's method, the new value of x is

$$x^{p+1} = x^p - \frac{F(x^p)}{F'(x^p)}. \tag{2.92}$$

Using Newton's method for local linearization, Gauss-Seidel-Newton method can be derived in scalar equation format (Trottenberg *et al.*, 2000):

$$x_i^{p+1} = x_i^p - \frac{1}{a_{ii} + g'(x_i)} \left(\sum_{j=1}^{i-1} a_{ij} x_j^{p+1} - \sum_{j=i}^{N} a_{ij} x_j^p + g(x) - b_i \right). \tag{2.93}$$

With the nonlinear iterative method for relaxation, the next step is to discuss how Full Approximation Scheme changes the v-cycle in two-grid method because nesting two-grid method results in Multigrid method. On the fine grid, the nonlinear system is

$$A^h(u^h) = A^h u^h + g(u^h) = f^h. \tag{2.94}$$

where u^h is the exact solution. After relaxing for a few times (typically three times), an approximation solution v^h can be achieved. The residual is

$$r^h = f^h - A^h(u^h). \tag{2.95}$$

Then, both the approximate solution v^h and the residual r^h are restricted to the coarse grid:

$$v^{2h} = R_{dir} v^h, \tag{2.96}$$

$$r^{2h} = R r^h, \tag{2.97}$$

where R is the restriction operator from Algebraic Multigrid. Whereas R_{dir} is injection type restriction, which means no averaging and only the value of coarse grid point itself is transferred to next level. The new forcing term becomes

$$f^{2h} = A^{2h}(v^{2h}) + r^{2h}. \tag{2.98}$$

Then, directly or iteratively solve the equation

$$A^{2h}(v^{2h}_{appr}) = f^{2h}. \tag{2.99}$$

The difference between the approximate solution and initial guess, which is restricted from fine grid, is the correction to fine grid:

$$e^{2h} = v^{2h}_{appr} - v^{2h}. \tag{2.100}$$

Just like what standard Multigrid does, the correction is then interpolated back to the fine grid and applied to the approximation value:

$$e^h = Ie^{2h}, \tag{2.101}$$

$$v^h = v^h + e^h. \tag{2.102}$$

A numerical experiment is taken to verify the effectiveness of Full Approximation Scheme to solve nonlinear system. Pure Gauss-Seidel-Newton method is used as the iterative method to be compared with Full Approximation Scheme:

$$\begin{aligned} & -\Delta u + \gamma u \exp(u) = f, \\ & 0 < x < 1, \\ & 0 < y < 1, \\ & \gamma = 1, \\ f = 2[(x - x^2) + (y - y^2)] + & \gamma(x - x^2)(y - y^2)\exp((x - x^2)(y - y^2)). \end{aligned} \tag{2.103}$$

Boundary conditions are

$$u(0, y) = u(1, y) = u(x, 0) = u(x, 1) = 0. \tag{2.104}$$

The analytical solution is

$$u(x, y) = (x - x^2)(y - y^2). \tag{2.105}$$

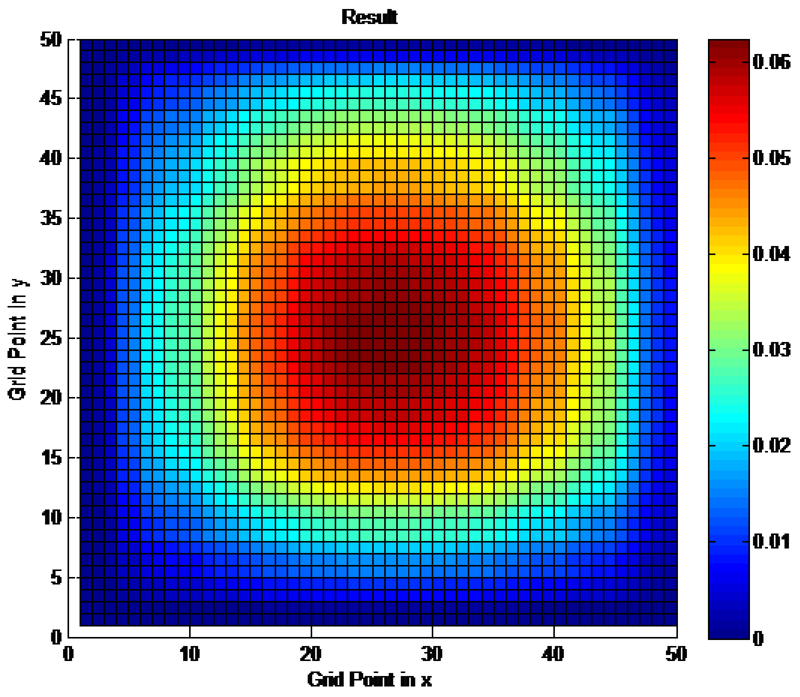


Figure 2.28: Result of Full Approximation Scheme Numerical Experiment.

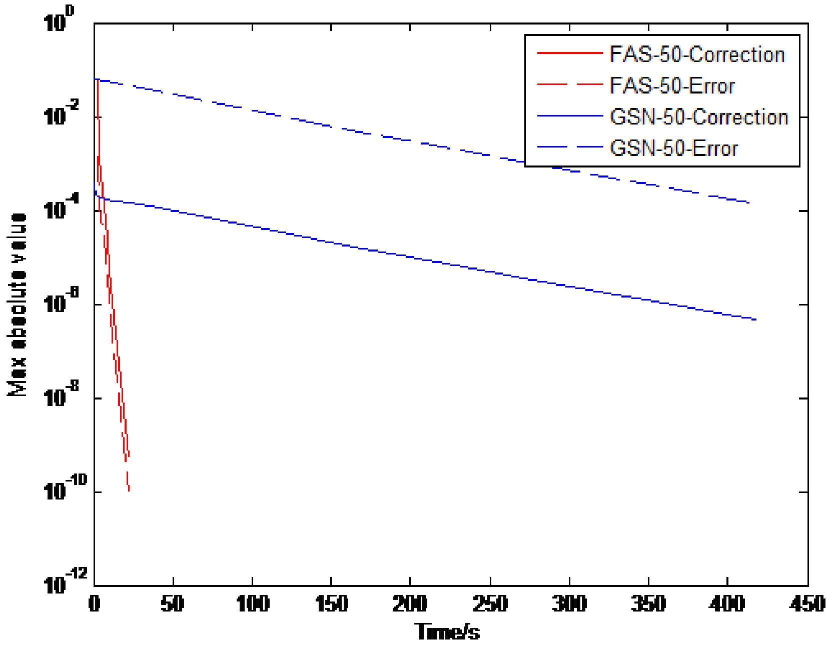


Figure 2.29: Convergence of Full Approximation Scheme Numerical Experiment. (50 by 50 Mesh Grid)

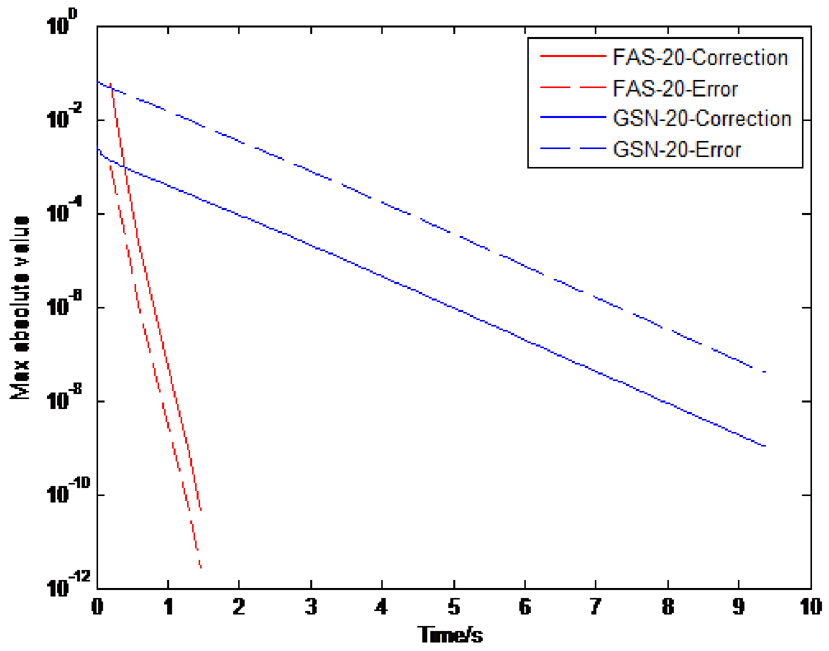


Figure 2.30: Convergence of Full Approximation Scheme Numercial Experiment. (20 by 20 Mesh Grid)

Two type of mesh grids are tested. They are 50 by 50 mesh grid and 20 by 20 mesh grid. The solution of 50 by 50 mesh grid is shown in figure 2.28. In figure 2.29 and figure 2.30, Full Approximation Scheme demonstrates very fast convergence speed. Especially in 50 by 50 mesh grid case, Full Approximation Scheme takes 16.42s to reach the maximum update of 5.3E-8, while Gauss-Seidel-Newton method takes 418.2s to reach the maximum update of 4.6E-7. That is Full Approximation Scheme is 24 times faster while achieving 10 times higher accuracy. In 20 by 20 mesh grid case, Full Approximation Scheme takes 1.27s to reach the maximum update of 1.07E-9. Gauss-Seidel-Newton method takes 9.348s to reach the maximum update of 1.08E-9. In this case, Full Approximation Scheme is 6.36 times faster with the same accuracy. In the comparison between 50 by 50 mesh grid results and 20 by 20 mesh grid results, it is explained why Multigrid method is faster when compared to pure iterative method.

2.6 Full Multigrid Method

Both standard Multigrid method and Full Approximation Scheme use V-cycle, which starts with the finest level. However, Full Multigrid method starts with the coarsest level and ramps up to the finest level with multiple V cycles like the Russian Dolls. V cycle in a 7 levels program is shown in 2.31, together with corresponding Full Multigrid cycle. Because Full Multigrid cycle starts with the coarsest level,

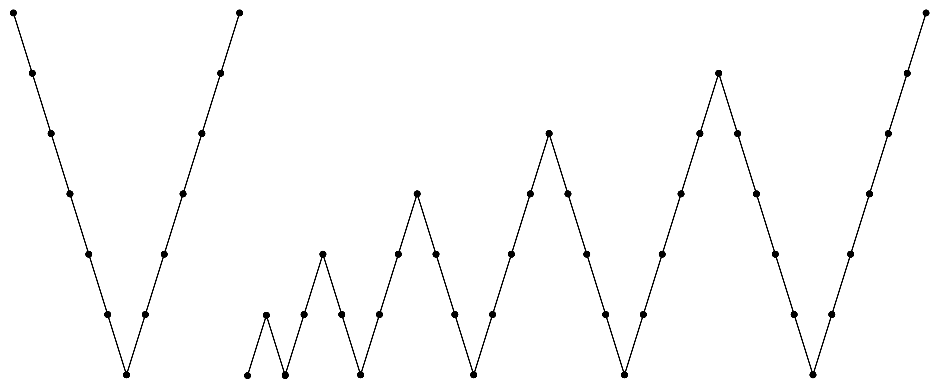
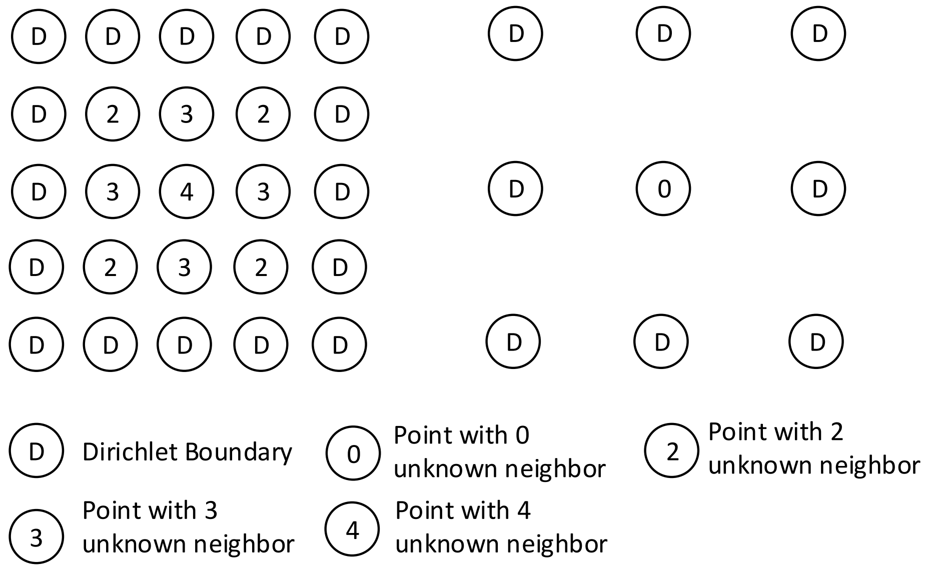


Figure 2.31: V Cycle and Full Multigrid Cycle in a 7 Levels Program.

which may contain only one unknown point, in this case, this unknown point can be rather accurately calculated from surrounding boundary conditions. Each time the mesh grid is interpolated to a new level, the coarse grid points can be assumed to be accurate while the inaccurate new fine grid points only take a small portion. This ensures relatively small accumulated error, which can be fatal for nonlinear system because exponential terms easily increase fast to NaN (not a number in computers) with accumulated error. The right panel of figure 2.32 is a coarse mesh. A type 0 point has four known boundary points around it. It contains no accumulative error and the only error is the discretization error. On the contrary, standard Multigrid method V cycle starts from the finest level where only very few portions of the total number of points are known boundary conditions, which makes most inner points to be calculated from their surrounding unknown points. The left panel of figure 2.32 is a fine mesh. A type 2 point has two unknown neighbor points. A type 4 point has four unknown neighbors, which has higher accumulative error than type 2 point.


D
2
3
4
0
Dirichlet Boundary
Point with 0 unknown neighbor
Point with 2 unknown neighbor
Point with 3 unknown neighbor
Point with 4 unknown neighbor
Figure 2.32: Scheme of a Fine Mesh and Coarse Mesh.

Because of its insensitivity to initial guess, Full Multigrid Method is typically used as a preconditioner to make the solver robust and independent from initial guess.

2.7 Summary of Various Multigrid Methods

In this chapter, various Multigrid methods are discussed. Some of them contain multiple modules which are actually interchangeable. Figure 2.33 illustrates their relationship. Three stages are defined. The setup stage is to generate coarser mesh levels and calculate restriction, interpolation and coefficient matrices. Depending on the properties and requirements of the problem, either standard Multigrid method or Algebraic Multigrid method can be selected. The initialization stage is to find a suitable initial guess as input for the real solver. Full Multigrid method is optional but strongly recommended for the robustness and accuracy of the whole solver. The third stage is to solve the matrix system. Depending on the nature of the system, standard Multigrid works for linear system while Full Approximation Scheme works

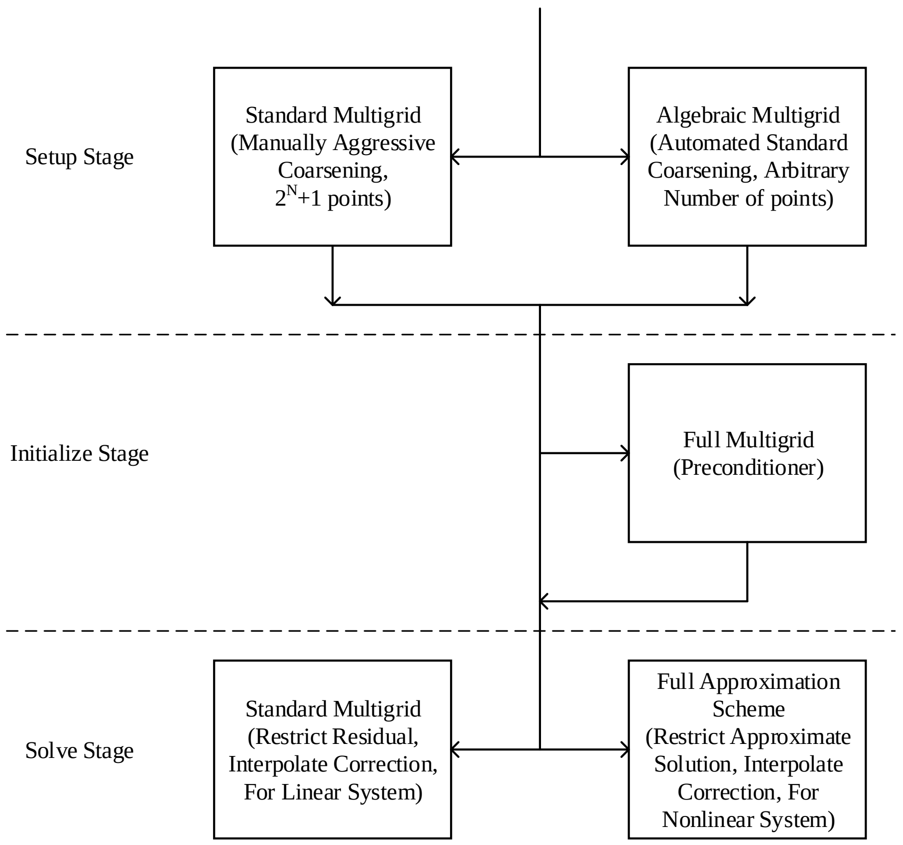

Setup Stage
Standard Multigrid (Manually Aggressive Coarsening, 2^N+1 points)
Algebraic Multigrid (Automated Standard Coarsening, Arbitrary Number of points)
Initialize Stage
Full Multigrid (Preconditioner)
Solve Stage
Standard Multigrid (Restrict Residual, Interpolate Correction, For Linear System)
Full Approximation Scheme (Restrict Approximate Solution, Interpolate Correction, For Nonlinear System)

Figure 2.33: Flowchart of Multigrid System.

for both linear and nonlinear systems.

For example, consider a case when strongly nonlinear partial differential equation needs to be solved on a square area without explicit requirement for mesh grid spacing. Then standard Multigrid method will be selected for setup stage. Because of the strong nonlinearity, Full Multigrid cycle is used to generate a good initial guess input for the solver. Finally, Full Approximation Scheme is selected to solve the nonlinear problem directly.

Chapter 3

PHYSICAL MODEL OF COMMON-SOURCE AND COMMON-DRAIN FINFET DEVICES CONFIGURATION

3.1 Geometry and Structures

A common-source FinFET device can be treated as two FinFET MOSFET devices sharing their source contact. Similarly, a common-drain FinFET device contains two FinFETs sharing their drain contact. Putting one device in normal on-state, the other device can be used to accurately measure the heat of its neighbor. To create an accurate relation between the heat and output current, both experimental research and theoretical simulation need to be performed to understand its physical behavior.

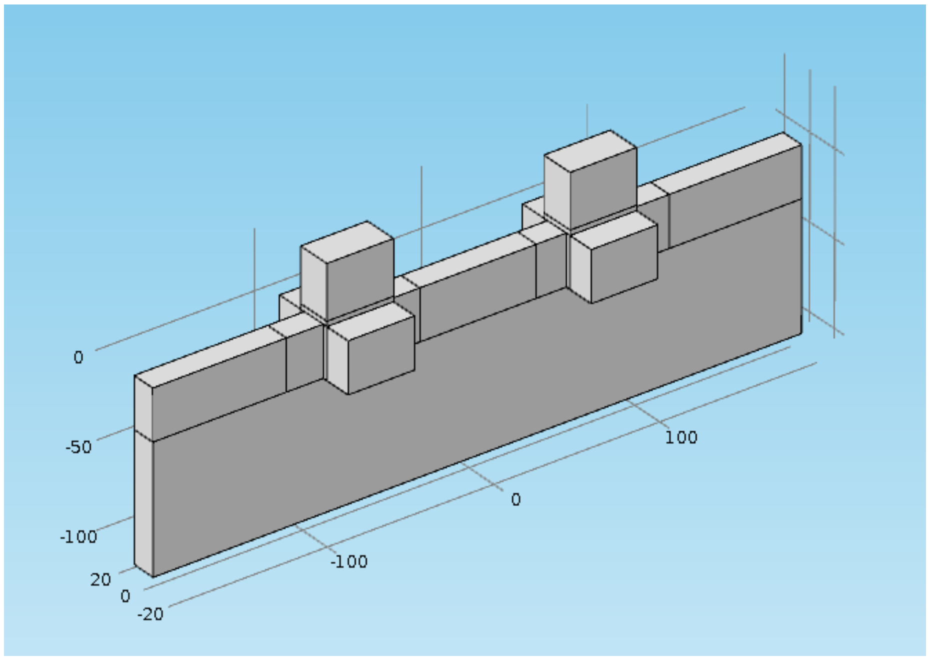


Figure 3.1: Scheme of 3D Common-source or Common-drain FinFET from IMEC.

Figure 3.1 shows the 3D geometry. Compared to conventional 2D planar devices, this geometry is much more complex, which leads to more mesh regions, more mesh grid points, different kinds of mesh spacing, and variation of materials properties. Algebraic Multigrid is an ideal choice because it can deal with complex mesh grids and material properties automatically. Moreover, from 2D to 3D, the number of mesh grid points increases tens, even hundreds, of times. Conventional Successive-Over-Relaxation and Conjugate Gradient methods take the largest part of overall time to calculate electrical field from charge density via Poisson's equation. They have the same complexity of $O(N^{1.5})$, which means that the number of grid points becomes 100 times of its original number. Time consumption becomes 1000 times of its original time cost. If multi-scale simulation is taken into consideration for circuit level simulation, the total time cost needs to be multiplied by the number of devices utilized. This problem largely restricts the scalability of multi-scale simulation. Fortunately, the complexity of Multigrid method is $O(N)$. It saves more time with increasing number of grid points.

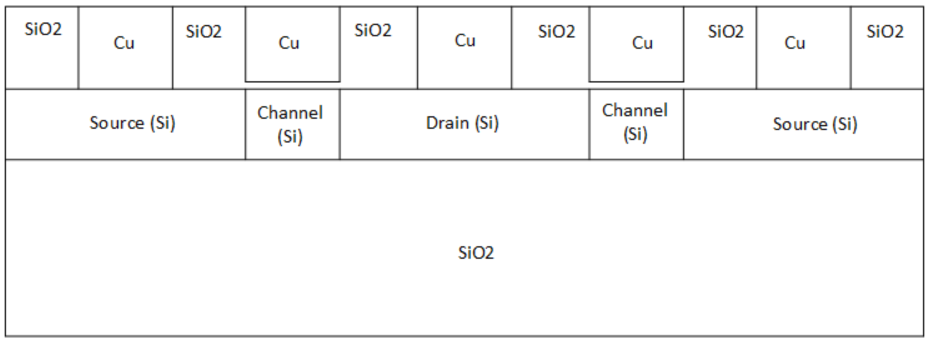


Figure 3.2: 2D Scheme of a Common-drain FinFET.

The 3D FinFET device is simplified to a 2D common-drain device for facilitate the application of Multigrid method. Figure 3.2 shows the 2D geometry. For the numerical solver, there is no difference between a typical MOSFET device and a common-drain device. Therefore, the 2D common-drain device is further simplified to a conventional 2D MOSFET device shown in figure 3.3.

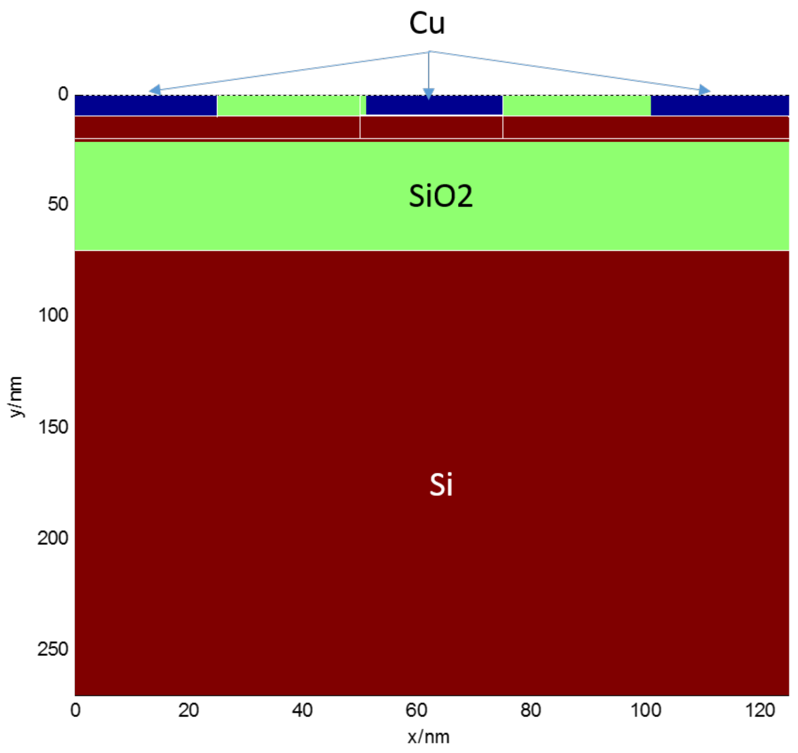


Figure 3.3: Geometry of Conventional MOSFET.

Source and drain are doped to $N_D^+ = 10^{20}cm^{-3}$. Chanel is doped to $N_A = 10^{18}cm^{-3}$. Substrate is doped to $N_A^- = 10^{16}cm^{-3}$.

Source and drain contacts are ohmic contacts, which implies charge neutrality. Numerically, they are Dirichlet boundary conditions. Gate contact is also Dirichlet boundary. All the rest are Neumann boundary conditions. No voltage is applied to source, drain, gate or substrate contact as equilibrium solution is being calculated.

3.2 Mesh Spacing

There are a few guidelines for determining mesh spacing. Source, drain, and channel regions require uniform square mesh (1nm by 1nm). The thickness of oxide under the gate is only 1.2nm. Thus a mesh spacing of 0.3nm is required to ensure sufficient grid points for such a thin layer (figure 3.5). For oxide box and semiconductor substrate, there is no specific requirements for spacing. Interfaces should have relatively small mesh spacing because the dielectric constant changes at the contact of two different materials. In this case, the largest mesh spacing is 19nm, which is larger than the smallest mesh spacing, 0.3nm, by a factor of 63. The complexity of mesh grid

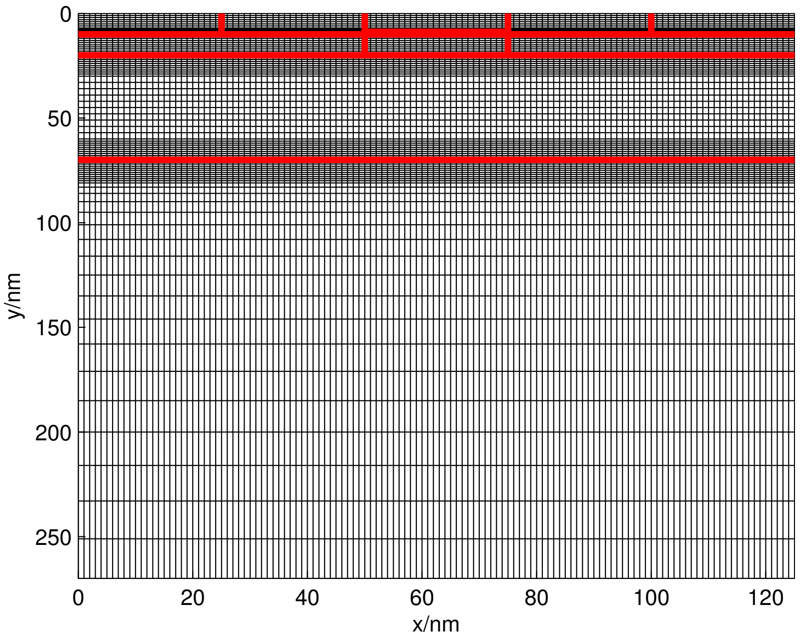


Figure 3.4: Fine Mesh of 2D MOSFET.

makes it hard to use a predefined number of $2^N + 1$ grid points in each direction. Therefore, Algebraic Multigrid is very suitable for this simulation.

3.3 Simulation Results

Initially, only Algebraic Multigrid method is developed to automatically generate coarser mesh grid levels and calculate interpolation, restriction, and coefficient matrices. It is used to replace the standard Multigrid method in setup stage, while leaving the solve stage unchanged. The convergence is shown in figure 3.6. Algebraic

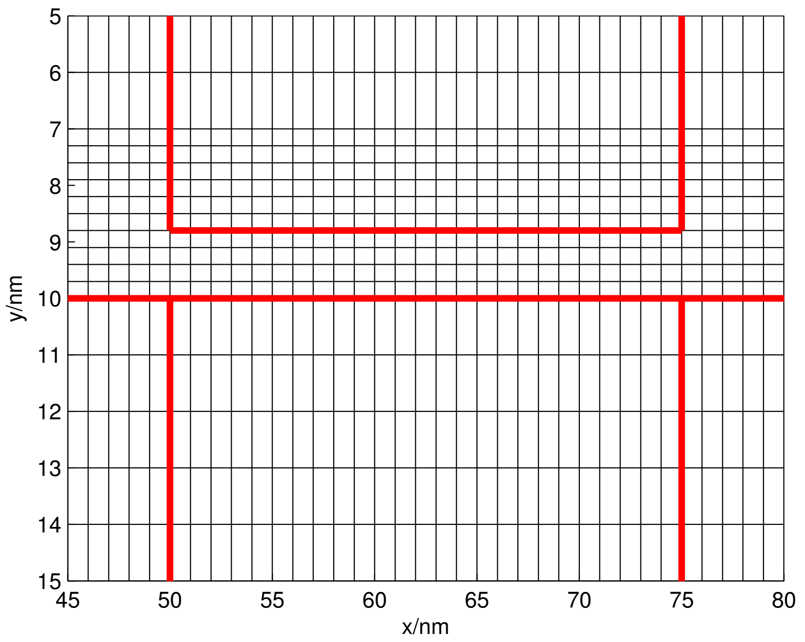


Figure 3.5: Mesh Grid under the Gate Contact.

Multigrid method shows no advantages, though it has steeper slope which means eventually it will become better. This is influenced by the way the Poisson's equation is linearized. The Taylor expansion method actually changes the linear system each time a value is updated. In this program, the Algebraic Multigrid setup is performed once every grid point is updated. This balances the frequency of setup and accuracy. However, the final result is not satisfactory, which leads to the development of Full Approximation Scheme and Full Multigrid method to solve nonlinear system directly. The flowchart of the final solver is shown in figure 3.7. With the final Poisson's equation solver, two initial guesses discussed in the second chapter are tested. The potential profile calculated by Multigrid method is shown in figure 3.8.

There is a dashed line at the left edge of the potential profile. A cutline of potential is plotted against the position in figure 3.9. The convergence criterion is 1E-5V in this simulation. However, SOR method is manually aborted because it could not reach

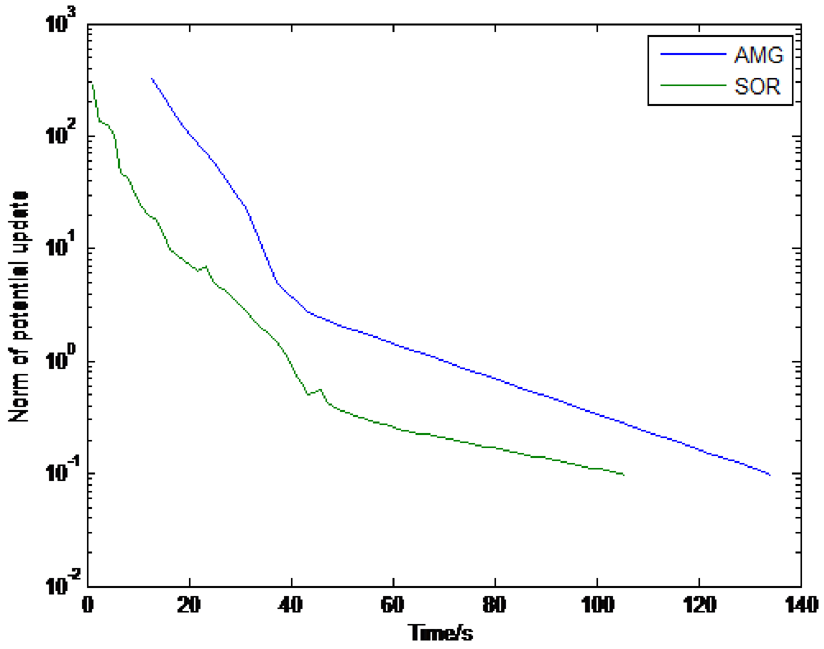


Figure 3.6: Convergence of Algebraic Multigrid Method on 2D MOSFET Device.

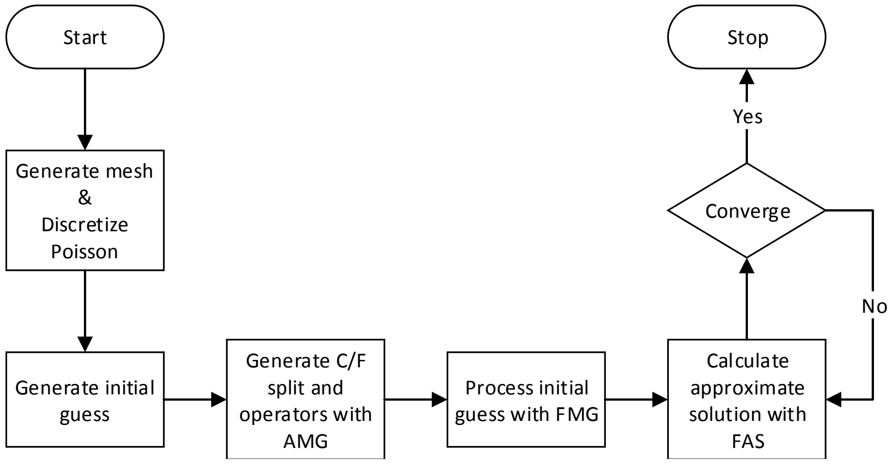

Start
Generate mesh & Discretize Poisson
Generate initial guess
Generate C/F split and operators with AMG
Process initial guess with FMG
Calculate approximate solution with FAS
Converge
No
Yes
Stop

Figure 3.7: Flowchart of Final Poisson's Equation Solver.

convergence for a very long time. Figure 3.10 shows the details about convergence.

The legend in figure 3.9 shows two lines for Multigrid solver with respect to two different initial guesses. However, only one red line can be found, while the two black lines have large difference between them. This demonstrates the accuracy and robustness of Multigrid method. Because Multigrid method is effective for all error

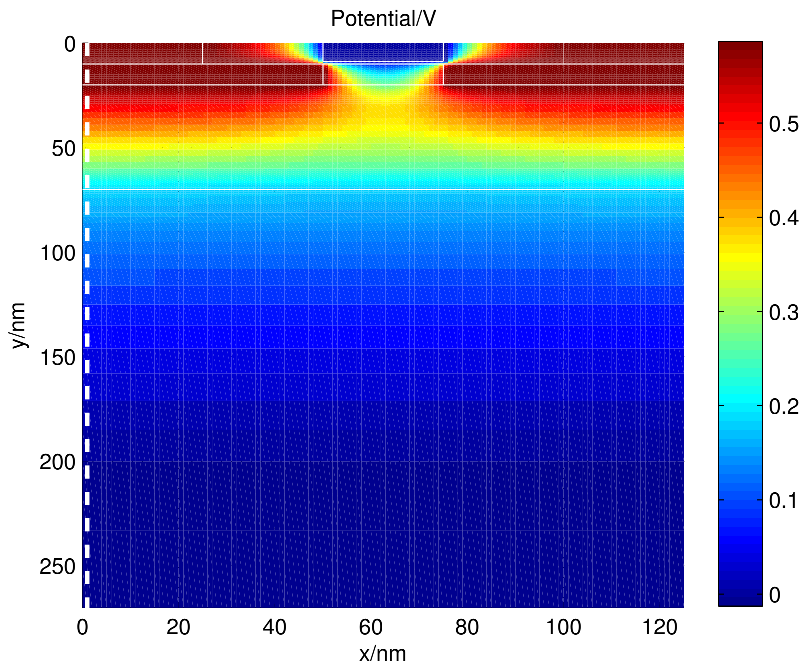


Figure 3.8: Potential Profile of Final Poisson's Equation Solver.

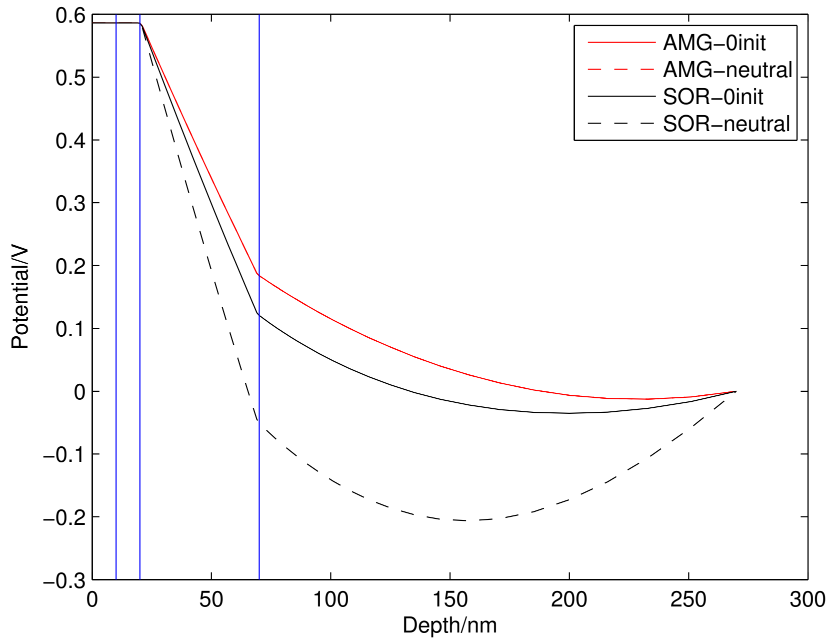


Figure 3.9: Cutline of Potential along Dashed Line.

frequency higher than the corresponding frequency of its finest mesh and Full Multi-gird cycle has almost no requirement for initial guess to generate high quality input for Multigrid V cycle, the final results are actually independent from these two initial guesses. On the contrary, SOR method is only effective to the error frequency that coheres with the mesh grid. An initial guess commonly introduces low frequency error because it is only a very rough estimation of the accurate solution. A large amount of low frequency error introduced by an initial guess remains even the maximum updates reduces to a small value. Therefore, it is accurate and reliable to use the maximum update as an indicator of convergence in Multigird method regardless of the quality of an initial guess.

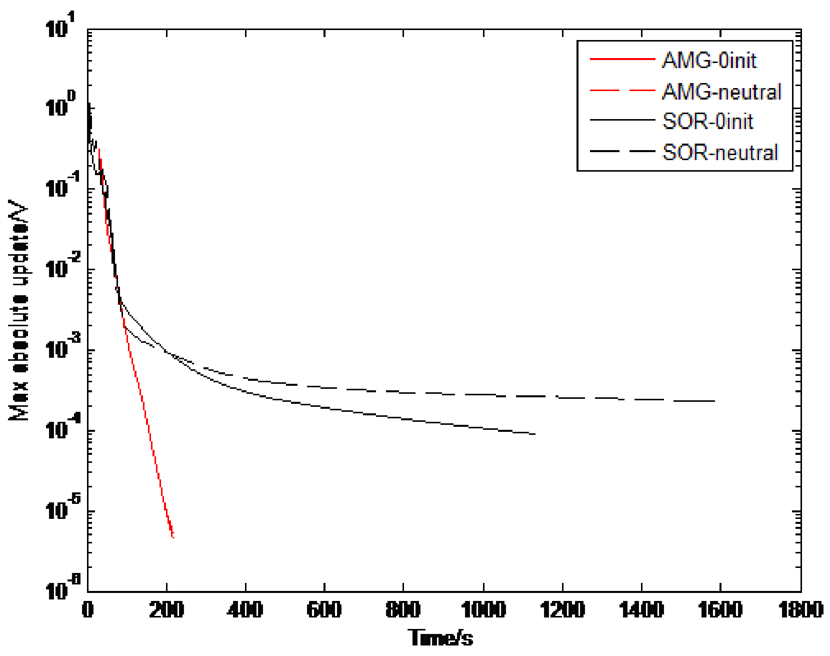


Figure 3.10: Convergence of Final Multigrid Solver and SOR Solver.

In figure 3.10, two lines of Multigrid solver overlap, which again demonstrates the robustness of Multigrid method. As for convergence speed, SOR method lines start to bend at about 2E-3V. Before this point, all four lines overlap showing that SOR method reduces high frequency error effectively. Afterward, red lines representing Multigrid method decline straightly while black lines representing SOR method tend to a constant value. In SOR case, the high frequency error has been reduced while the low frequency is being reduced slowly. In Multigrid case, error of all frequencies are effectively reduced in Multigrid solver, so the maximum update reduces linearly. However, SOR method has to inefficiently reduce low frequency error, which makes the maximum update remain a constant small value.

Chapter 4

CONCLUSIONS

Compared with conventional single level iterative methods such as Successive-Over-Relaxation method, Algebraic Multigrid method together with Full Approximation Scheme and Full Multigrid demonstrate very high efficiency in solving nonlinear matrix systems. Moreover, they are robust and independent of initial guess, which largely lowers the requirement for the quality of initial guess, thus reducing the cost to find a good guess.

Although the Algebraic Multigrid method requires larger setup time, it is only necessary for the first time a linear or nonlinear system is created. The high efficiency of the Multigrid method can easily justify the initial setup time cost. Besides, in most cases, mesh grids and the corresponding matrix system from the discretized physics equations do not change frequently. Thus, the Multigrid method is a good substitution to the single level iterative methods.

On coding level, this research uses MATLAB for fast and easy prototyping, which trades off the speed of the program. As the algorithm of Multigrid method is developed and tested in MATLAB, it would be a good idea to implement such code in a compiled language such as C or FORTRAN. Also, this solver can easily be extended to 3D case.

REFERENCES

Aymerich-Humet, X., F. Serra-Mestres and J. Millán, “An analytical approximation for the Fermi-Dirac integral”, Solid-State Electronics **24**, 10, 981–982, URL `http://www.sciencedirect.com/science/article/pii/0038110181901210` (1981).

Bohr, M., “The evolution of scaling from the homogeneous era to the heterogeneous era”, in “2011 International Electron Devices Meeting”, pp. 1.1.1–1.1.6 (IEEE, 2011), URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6131469`.

Briggs, W., V. Henson and S. McCormick, *A Multigrid Tutorial, Second Edition* (Society for Industrial and Applied Mathematics, 2000), URL `http://dx.doi.org/10.1137/1.9780898719505`.

Cartwright, J., “Intel enters the third dimension”, Nature URL `http://www.nature.com/news/2011/110506/full/news.2011.274.html` (2011).

Demmel, J., *Applied Numerical Linear Algebra* (Society for Industrial and Applied Mathematics, 1997), URL `http://epubs.siam.org/doi/abs/10.1137/1.9781611971446`.

Falgout, R., “An Introduction to Algebraic Multigrid Computing”, Computing in Science & Engineering **8**, 6, 24–33, URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1717312` (2006).

Hisamoto, D., W.-C. Lee, J. Kedzierski, H. Takeuchi, K. Asano, C. Kuo, E. Anderson, T.-J. King, J. Bokor and C. Hu, “FinFET-a self-aligned double-gate MOSFET scalable to 20 nm”, (2000).

Huang, X. H. X., W.-C. L. W.-C. Lee, C. K. C. Kuo, D. Hisamoto, L. C. L. Chang, J. Kedzierski, E. Anderson, H. Takeuchi, Y.-K. C. Y.-K. Choi, K. Asano, V. Subramanian, T.-J. K. T.-J. King, J. Bokor and C. H. C. Hu, “Sub 50-nm FinFET: PMOS”, in “International Electron Devices Meeting 1999. Technical Digest (Cat. No.99CH36318)”, pp. 67–70 (IEEE, 1999), URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=823848`.

Kuhn, K. J., “Considerations for ultimate CMOS scaling”, URL `http://apps.webofknowledge.com.ezproxy1.lib.asu.edu/full_record.do?product=WOS&search_mode=GeneralSearch&qid=3&SID=2F45Dg9TuwBOQ9P6D6H&page=1&doc=3` (2012).

Lundstrom, M. and R. Schuelke, “Modeling semiconductor heterojunctions in equilibrium”, Solid-State Electronics **25**, 8, 683–691, URL `http://www.sciencedirect.com/science/article/pii/0038110182901952` (1982).

Saad, Y., *Iterative Methods for Sparse Linear Systems* (Society for Industrial and Applied Mathematics, 2003), second edn., URL `http://epubs.siam.org/doi/abs/10.1137/1.9780898718003`.

Strang, G., *Computational Science and Engineering*, vol. 1 (Wellesley-Cambridge Press, 2007), URL `http://books.google.com/books?id=GQ9pQgAACAAJ&pgis=1`.

Stüben, K. and J. W. Ruge, *Algebraic Multigrid*, chap. 4, pp. 73–130 (Society for Industrial and Applied Mathematics, 1987), URL `http://epubs.siam.org/doi/abs/10.1137/1.9781611971057.ch4`.

Trottenberg, U., C. W. Oosterlee and A. Schuller, *Multigrid*, vol. 20 (Academic Press, 2000), URL `https://books.google.com/books?id=9ysyNPZoR24C&pgis=1`.

Trottenberg, U., C. W. Oosterlee and A. Schüller, *Multigrid* (Academic Press, 2001), URL `http://books.google.com/books?id=-og1wD-Nx_wC`.

Van Henson, E., "Multigrid methods for nonlinear problems: An overview", SPIE proceedings series pp. 36–48, URL `http://cat.inist.fr/?aModele=afficheN&amp;cpsidt=15283351` (2003).

Vasileska, D., S. M. Goodnick and G. Klimeck, *Computational Electronics: Semiclassical and Quantum Device Modeling and Simulation* (CRC Press, 2010), URL `https://books.google.com/books?id=QV_MBQAAQBAJ`.