

AN ABSTRACT OF THE THESIS OF

Can K. Sandalcı for the degree of Master of Science in Computer Engineering
presented on May 8, 1996.

Title:

Three Dimensional Monte Carlo Simulator with Parallel Multigrid Poisson Solver.

Redacted for Privacy

Abstract approved: _____

Çetin K. Koç

We present the results of embedding a multigrid solver for Poisson's equation into the parallel 3D Monte Carlo device simulator, PMC-3D. First we compare the sequential multigrid implementation to the sequential Successive Overrelaxation (SOR) Monte Carlo code used previously in PMC-3D. Depending on the convergence threshold, we obtain significant speedups ranging from 6 to 15. The parallel multigrid implementation is done by extending the partitioning algorithm and the interprocessor communication routines used in the SOR implementation to service multiple grids. The Monte Carlo code with the parallel multigrid Poisson solver is 4 to 9 times faster than the Monte Carlo code with the parallel SOR code, based on timing results on a 32-processor nCUBE multiprocessor.

©Copyright by Can K. Sandalcı

May 8, 1996

All rights reserved

Three Dimensional Monte Carlo Simulator with Parallel Multigrid Poisson Solver

by

Can K. Sandalcı

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed May 8, 1996
Commencement June 1996

Master of Science thesis of Can K. Sandalcı presented on May 8, 1996

APPROVED:

Redacted for Privacy

Major Professor, representing Computer Engineering

Redacted for Privacy

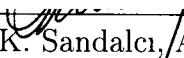
Chair of the Department of ~~Electrical and~~ Computer Engineering

Redacted for Privacy


Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy


Can K. Sandalcı, Author

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my major professor, Çetin K. Koç, whose constant support, encouragement and guidance made this research a very valuable academic experience. I also owe special thanks to Prof. Stephen M. Goodnick, for his guidance, valuable comments and support throughout the course of this research.

My special thanks are due to professors Ben Lee and Goran N. Jovanovic for serving on my defense committee.

I would also like to thank Marco Saraniti for helpful discussions and valuable comments in relation to this work. Additional thanks go to Shankar S. Pennathur for many helpful discussions about the PMC-3D simulation package.

Finally, with sincere appreciation, I would like to thank my parents, for their unending love, encouragement and support throughout my life. I am grateful to them, for their uncountable sacrifices to make me start the life at a better point than they once did.

Financial support for this research was provided by the National Science Foundation.

TABLE OF CONTENTS

	<u>Page</u>
1 INTRODUCTION	1
2 THE PMC-3D DEVICE SIMULATOR	4
2.1 The nCUBE Multicomputer System.....	5
2.2 Basic Monte Carlo Algorithm	6
2.3 Parallel 3-D Monte Carlo Device Simulator.....	8
2.4 The SOR Solver	9
2.5 Summary.....	15
3 THE MULTIGRID SOLVER.....	16
3.1 Principles of the Multigrid Method.....	17
3.2 Implementation Details	21
3.2.1 Discretization	21
3.2.2 Coarsening	22
3.2.3 Boundary Conditions	23
3.2.4 Relaxation Method	24
3.2.5 Restriction and Prolongation	25
3.2.6 Parallelization	27
3.3 Summary.....	29
4 RESULTS AND DISCUSSIONS.....	30
4.1 Timings for the Serial and Parallel PMC-3D.....	30

TABLE OF CONTENTS (Continued)

	<u>Page</u>
4.2 Discussion.....	32
4.3 Future Work	35
BIBLIOGRAPHY	36

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
2.1 Hypercube topology examples for dimensions 2, 3, and 4.	5
2.2 The flow diagram of the simple Monte Carlo algorithm.	7
2.3 The flow diagram of the parallel Monte Carlo device simulation. . . .	10
2.4 The geometrical partitioning of the semiconductor device domain onto a hypercube of 16 processors.	11
2.5 The finite differencing stencil for the Poisson's equation.	12
2.6 The 2-D representation of the communication pattern of the red-black ordered SOR solver.	13
2.7 Workload contributions of different parts of the PMC-3D code.	14
3.1 Error smoothing effect of the iterative schemes.	18
3.2 Two common multigrid cycles are (a) V Cycle ($\gamma = 1$) and (b) W Cycle ($\gamma = 2$).	20
3.3 Two dimensional representation of the multiprocessor coarsening scheme.	22
3.4 The intergrid transfer operators. a) Prolongation. b) Restriction. . .	26
3.5 Two dimensional representation of the parallel implementation of the restriction operation.	28
3.6 Two dimensional representation of the parallel implementation of the prolongation operation.	29
4.1 The MESFET structure used as a model problem for the simulations.	30
4.2 The communication overhead for the multigrid solver.	33
4.3 The computation time of the Poisson solver versus convergence threshold for the serial code running on a HP 712/80, for 100 PMC- 3D iterations.	34
4.4 The computation time of the Poisson solver versus convergence threshold for the serial code running on a 32 node nCUBE multi- computer, for 100 PMC-3D iterations.	34

LIST OF TABLES

<u>Table</u>		<u>Page</u>
4.1	The timings of the PMC-3D device simulator with SOR and MG solvers on a single HP 712/80 workstation.	31
4.2	The timings of the PMC-3D device simulator with SOR and MG solvers on a 32 node nCUBE multiprocessor.	31

THREE DIMENSIONAL MONTE CARLO SIMULATOR WITH PARALLEL MULTIGRID POISSON SOLVER

1. INTRODUCTION

The increasing computational power of today's supercomputers has made computer aided design (CAD) an important aspect of microelectronics. With constantly decreasing semiconductor device dimensions, there is a need for full, three-dimensional (3D) simulation tools. The feedback provided using these simulators will increase the efficiency and speed of the integrated circuit (IC) design process.

As the dimensions shrink and the internal electrical fields increase, the traditional semiconductor analysis tools which are based on low-order moments of the Boltzmann transport equation (e.g. the drift-diffusion model) fail to accurately represent the physical characteristics of the device. The Monte Carlo technique is a quite general and well established stochastic method, which has been applied to a variety of problems ranging from statistical physics to optimization problems. Solution of the Boltzmann transport equation using the Monte Carlo method is currently the most widespread technique used in semiconductor device simulation [1]. In the Monte Carlo method, the evolution of an ensemble of particles in energy or momentum space is simulated, where each simulated particle represents a collection of charge carriers. The motion of the charge carriers (electrons and holes) is assumed to be given by classical trajectories, interrupted by random, instantaneous scattering events. The motion and trajectories of the simulated particles are influenced by the electrical fields in the device, which are determined from the spatial charge distribution, by solving the Poisson's equation. The random scattering events are

generated stochastically using a random number generator and the quantum mechanical scattering probabilities for all possible mechanisms in the semiconductor. The solution of particle motion is synchronized with the solution of Poisson's equation, so that the time evolution of the fields in the device are accurately represented, which in turn are used in accelerating the particles over each time step.

The main complaint against the Monte Carlo modeling is the excessive computational time associated with it, particularly when combined with the solution of Poisson's equation in 3D. Consierable speedup compared to Monte Carlo technique has been reported in [3], where alternate particle methods using the lattice-gas cellular automaton are used for solving the Boltzmann equation. However, the principal bottleneck in the calculation is the solution of the Poisson's equation in 3D.

Parallel computing platforms provide some relief to the computational requirements of the Monte Carlo simulation. The parallel Monte Carlo Simulator, PMC-3D [4, 6] was developed at Oregon State University, which was implemented on a distributed memory nCUBE multiprocessor system. The development of parallel Monte Carlo simulation not only reduces the execution times associated with most typical simulation problems, but also enables the solution of problems larger in computational complexity, such as full three dimensional device simulations.

Significant speedup of the 2D Monte Carlo simulation has been reported by Saraniti et al [8] using a multigrid (MG) method for solving the Poisson's equation. In the Poisson solver module of the PMC-3D simulation package, a successive over relaxation (SOR) scheme was implemented. Subsequent studies have shown that the Poisson Solver uses up to 90% of the computation time in device simulation of real 3D structures [5]. Hence, a speedup obtained in the Poisson module imply a significant overall speedup in the PMC-3D code. The implementation of a MG

Poisson solver and the replacement of the former SOR solver in the PMC-3D code were considered in this work.

This thesis is organized as follows. In Chapter 2, a brief outline of the PMC-3D code is provided following the basic principles of the Monte Carlo technique as applied to semiconductor device simulations. The SOR Poisson solver is discussed and detailed internal timings of the PMC-3D package are provided to justify the need for a Poisson solver upgrade. In Chapter 3, the principles of the multigrid method are described and the implementation details of the three dimensional parallel multigrid Poisson solver are explained. Finally, in Chapter 4, the obtained results and accomplishments, as well as suggestions for future research are provided.

2. THE PMC-3D DEVICE SIMULATOR

With the increasing computing power of today's supercomputers, the integrated circuit (IC) development process is moving from an experimentally based approach to realistic computer simulations. For more than a decade, the most common tool for semiconductor device analysis has been the drift-diffusion model which represents the first two moments in the Boltzmann transfer equation. One of the main assumptions of this model is that a local relation exists between the electrical field and the carrier velocity of the form

$$v = \mu E,$$

where μ is the field dependent carrier mobility. With shrinking device sizes, this assumption is no longer valid and more realistic simulation models are needed. One of these approaches is the Monte Carlo method which by itself is quite general and has been applied to a variety of problems for decades.

Most of the current two-dimensional simulation packages are based on the assumption that the change in physical quantities over the third dimension (the width of the device) is negligible. This assumption is not always true, especially if high technology submicron devices are being simulated. The PMC-3D is a three dimensional device simulation package which is composed of a k-space Monte Carlo simulator and a Poisson solver as an extension to it. Since our main area of interest is improving the Poisson solver, only a brief outlook on the PMC-3D and the Monte Carlo method is provided in this chapter. The interested reader is referred to [6] and [7] for more detailed information.

This chapter is organized as follows. First some information about the nCUBE multicomputer system is provided on which the PMC-3D code is being

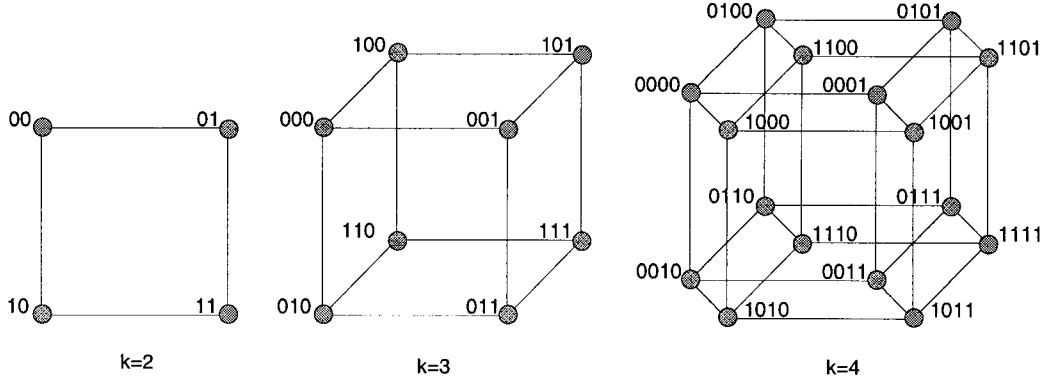


FIGURE 2.1. Hypercube topology examples for dimensions 2, 3, and 4.

developed. Then the PMC-3D algorithm is explained following some basic information about the Monte Carlo method. Finally the SOR Poisson solver is introduced, and detailed timings about the PMC-3D device simulator is provided to justify the need for a Poisson solver upgrade.

2.1. The nCUBE Multicomputer System

The nCUBE multicomputer system consists of a host processor and an array of processing elements interconnected in a hypercube topology. A hypercube of dimension k consists of 2^k processing elements which have direct physical connection to k other processors. The processing elements are numbered ranging from 0 to $2^k - 1$ and two processors are directly connected, if and only if their binary representations differ by only one bit. Figure 2.1 illustrates the hypercube topologies with dimensions 2, 3, and 4. The main advantage of this topology comes from the fact that, the message transfer time between any two processors is $\Omega(k)$.

The nCUBE2 multicomputer at Sandia National Laboratories consists of 1024 processors with 4 MB of local memory. The processors have 64 bit general purpose CPU's. The communication cost between two nodes can be characterized by a startup delay of 50–150 μsec . and a data transfer rate of about 2.2 MB/sec. The host programs execute on a Sun microcomputer under Unix operating system where the parallel code executes on nCUBE2 nodes under Vertex operating system.

2.2. Basic Monte Carlo Algorithm

The simulation of a particle representing a group of carriers in the device begins by generating its free-flight time according to the probability distribution determined by the scattering probabilities. The probability that a particle experiences a collision between the small time interval t and $t + dt$ is given by

$$P(t)dt = \Gamma e^{-\Gamma t} dt,$$

where Γ is the total scattering rate which is the sum of the rates corresponding to different scattering mechanisms, as

$$\Gamma = \sum_i \Gamma_i(v(t)) + \Gamma_s(v(t)).$$

The individual scattering rates are functions of the particle velocity (or equivalently energy or momentum). By adding a self-scattering term Γ_s , which does not change any of the particles parameters, the total scattering rate Γ remains constant which allows us to determine the random free-flight time as

$$t_r = \frac{1}{\Gamma} \ln r,$$

where r is a uniformly distributed random variable between 0 and 1. During the free flight, the electron wave vector \mathbf{k} changes continuously according to the relation

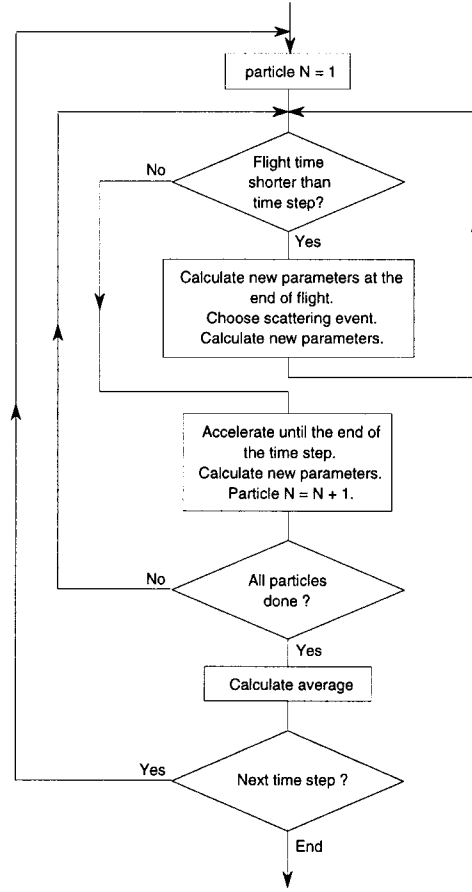


FIGURE 2.2. The flow diagram of the simple Monte Carlo algorithm [6].

$$\hbar \dot{\mathbf{k}} = eE$$

where \mathbf{k} , e and E represent the carriers wave vector, charge and energy respectively and \hbar is the Planck constant divided by 2π . Each particle is accelerated according to the above relationship during the free flight. Then the scattering mechanism responsible for terminating the free-flight is determined according to the relative probabilities of all possible mechanisms. Finally, the wave vector \mathbf{k} and the energy of the particle is updated according to the chosen scattering mechanism and the

particle is ready for the next free flight. The entire process is repeated for each particle until the end of the time step.

The ensemble of particles being simulated allows us to determine the instantaneous distribution function of the particles as well as the macroscopic average quantities such as drift velocity and average energy. The general structure of the algorithm is presented in Figure 2.1. The interested reader is referred to [1] and [2] for further details.

2.3. Parallel 3-D Monte Carlo Device Simulator

The PMC-3D algorithm is an extension of the standard k-space Monte Carlo method described in the previous section. Real space coordinates in three dimensions are added and the particle charges are assigned to the grid points. These charge values are used for solving the Poisson's equation on the entire grid. The Monte Carlo part is decoupled from the solution of the Poisson's equation over the interval of one time step. The particles are accelerated according to the forces derived from the solution of the Poisson's equation in the previous time step. Each simulated particle represents a superparticle with the effective charge selected so as to ensure the initial charge neutrality of the device.

The device grid is divided into three dimensional subgrids using the recursive bisectioning algorithm and assigned to processors using a gray code mapping. The recursive bisectioning works by splitting the device domain into two parts representing roughly equal amount of work and splitting the subgrids recursively until the desired number of subproblems are obtained. The mapping of a three dimensional grid on to a hypercube with 16 processing units is shown in Figure 2.3.

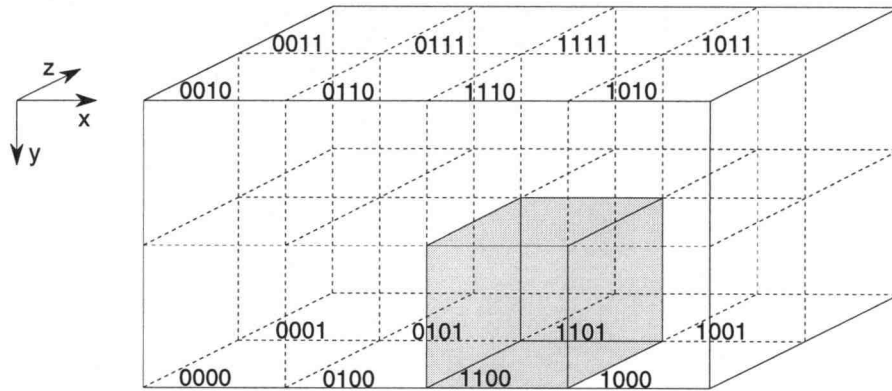


FIGURE 2.3. The geometrical partitioning of the semiconductor device domain onto a hypercube of 16 processors. The processors are labeled using binary numbers and the external interaction region between adjacent processors is shaded.

Each processor simulates the subensemble of particles using the Monte Carlo method. The particles that cross the subgrid boundary during the time-step are transferred among the neighboring processors. After the charge assignment is done, the Poisson's equation is solved. In the next time step the forces derived from the solution of the Poisson's equation are used for accelerating particles. The typical flow diagram of the Parallel Monte Carlo simulation is shown in Figure 2.4. Since our primary point of interest is solving the Poisson's equation, the interested reader is referred to [7] and [4, 6] for further details. The original PMC-3D code uses a red-black ordered SOR solver for solving the Poisson's equation. The implementation details of this solver will be discussed in the next section.

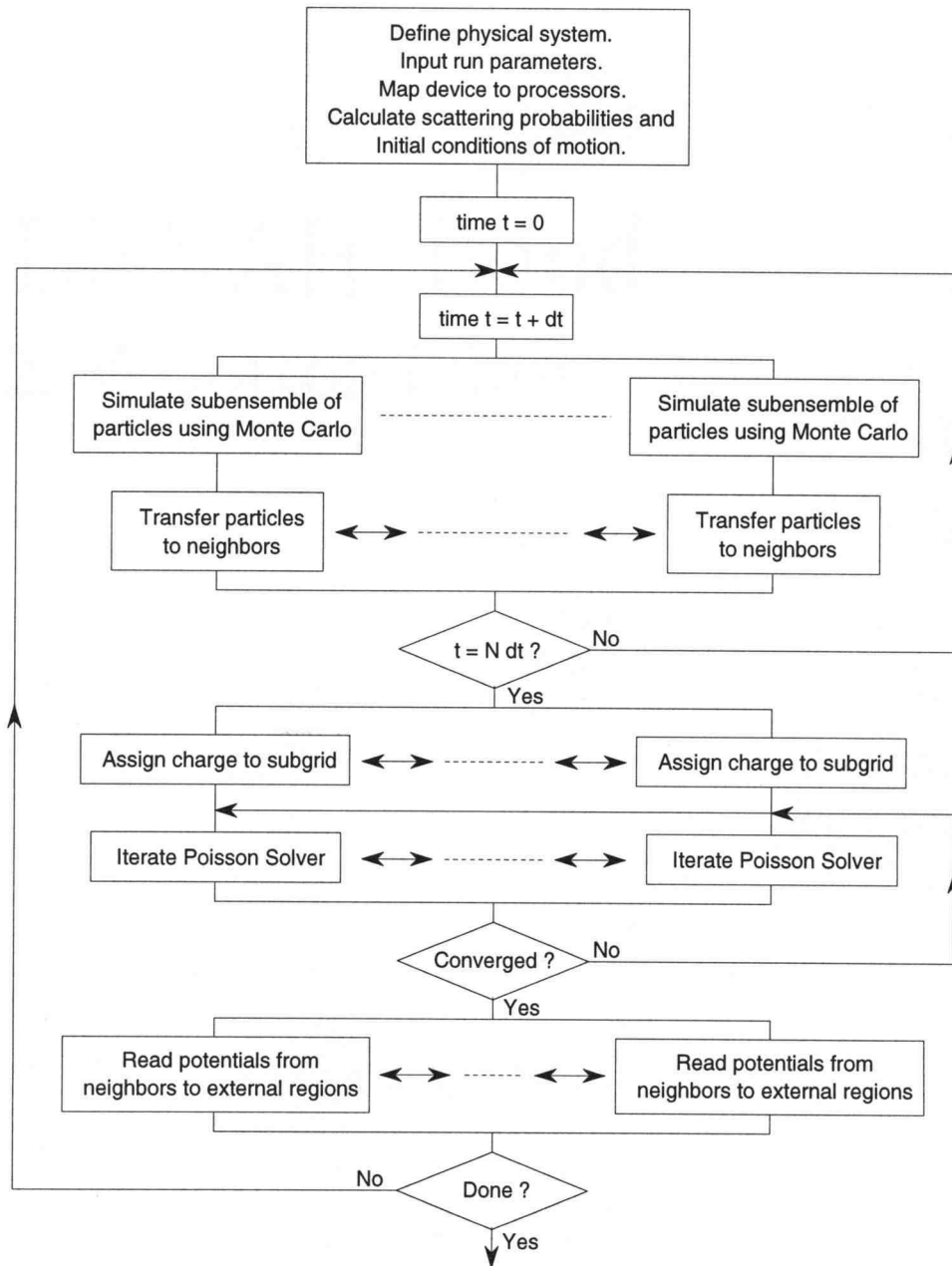


FIGURE 2.4. The flow diagram of the parallel Monte Carlo device simulation [4].

2.4. The SOR Solver

The solution of the Poisson equation specifies the position dependent potentials and thus, the electrical field across the device that is used to accelerate the particles appropriately. In the PMC-3D code, the Poisson's equation is solved using the Successive Overrelaxation method. In this section, a brief description of the method is given [4]. The Poisson's equation is given by

$$\nabla^2 \phi = -\frac{\rho}{\epsilon_s},$$

where ϕ , ρ , and ϵ_s refer to the spatially varying electrical potential, the charge density and the material dielectric permittivity respectively. The Poisson's equation can be expanded in three dimensions as

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} = -\frac{\rho}{\epsilon_s},$$

and can be discretized using finite differences on a general nonuniform grid as

$$\begin{aligned} & \frac{1}{h_{x-} + h_{x+}} \left(\frac{\phi_{x+1,y,z} - \phi_{x,y,z}}{h_{x+}} - \frac{\phi_{x,y,z} - \phi_{x-1,y,z}}{h_{x-}} \right) + \\ & \frac{1}{h_{y-} + h_{y+}} \left(\frac{\phi_{x,y+1,z} - \phi_{x,y,z}}{h_{y+}} - \frac{\phi_{x,y,z} - \phi_{x,y-1,z}}{h_{y-}} \right) + \\ & \frac{1}{h_{z-} + h_{z+}} \left(\frac{\phi_{x,y,z+1} - \phi_{x,y,z}}{h_{z+}} - \frac{\phi_{x,y,z} - \phi_{x,y,z-1}}{h_{z-}} \right) = -\frac{\rho_{x,y,z}}{2\epsilon_s}, \end{aligned}$$

where h_x , h_y , and h_z are the grid spacings in the x, y, and z directions respectively. The plus and minus signs in the subscript denote different directions as seen in Figure 2.5.

The value of the $\phi_{x,y,z}$ after the $(n+1)$ th iteration can be obtained by modifying the above equation as

$$\begin{aligned} \phi_{x,y,z}^* = & \frac{1}{\sum t_i} (t_1 \phi_{x+1,y,z} + t_2 \phi_{x-1,y,z} + t_3 \phi_{x,y+1,z} + \\ & + t_4 \phi_{x,y-1,z} + t_5 \phi_{x,y,z+1} + t_6 \phi_{x,y,z-1}) - \frac{\rho_{x,y,z}}{\epsilon_s}, \end{aligned}$$

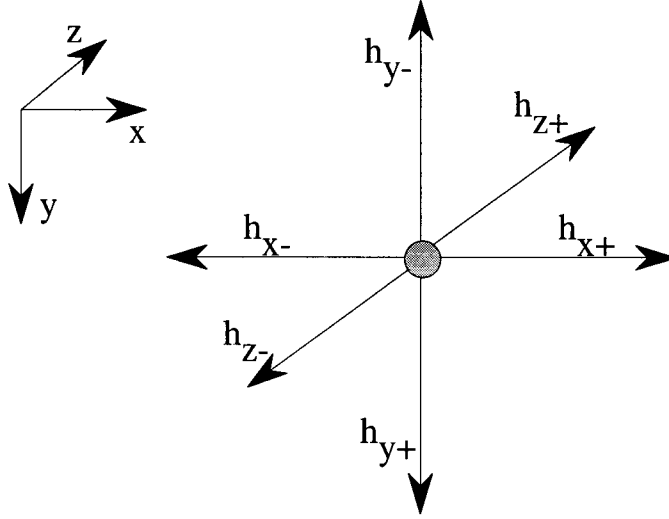


FIGURE 2.5. The finite differencing stencil for the Poisson's equation.

where t_i , $i = 1..6$ are constants calculated appropriately from the grid spacings. Using the above equation, the potential in the $(n + 1)$ th iteration is calculated as

$$\phi_{x,y,z}^{n+1} = \omega \phi_{x,y,z}^* + (1 - \omega) \phi_{x,y,z}^n,$$

where ω is the relaxation parameter in the range $1 \leq \omega \leq 2$.

The three dimensional grid is divided into two subgrids, corresponding to *red* and *black* orderings of the grid-points, like the black and white squares of a chess-board, as seen in Figure 2.6. Every relaxation sweep consists of two half sweeps, in which *red* and *black* points are updated alternatively. In the parallel implementation, before every half sweep, each processor communicates with its neighboring processors via message passing to obtain the updated potential values of the opposite colored grid points that are external to its subgrid. Convergence is reached

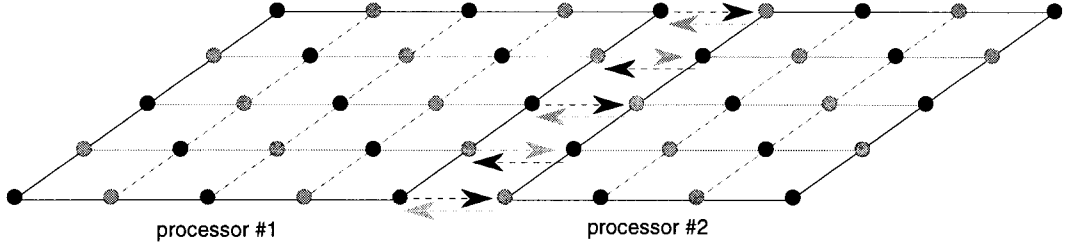


FIGURE 2.6. The 2-D representation of the communication pattern of the red-black ordered SOR solver. The light and dark arrows represent the communication attempts before relaxing the red and black ordered points respectively.

when the maximum norm of the residual in all subgrids is less than or equal to a fixed convergence threshold.

The relaxation parameter ω is determined dynamically using the Chebyshev acceleration method as,

$$\begin{aligned}\omega^{(0)} &= 1 \\ \omega^{(1/2)} &= 1/(1 - \rho_{Jacobi}^2) \\ \omega^{(n+1/2)} &= 1/(1 - \rho_{Jacobi}^2 \omega^{(n)}/4), \quad n = 1/2, 1, \dots, \infty\end{aligned}$$

where ρ_{Jacobi} is the spectral radius of the Jacobi iteration. The beauty of this approach comes from the fact that the norm of the error always decreases with each iteration while the asymptotic rate of convergence remains the same as the ordinary SOR.

The SOR method is easily parallelizable and fast compared to other methods, e.g, Jacobi and Gauss Seidel. However, the problems with the traditional iterative methods still prevail. The speed of convergence degrades with increasing iteration

numbers, while the number of iterations to reach a specific convergence threshold increase with increasing grid sizes.

The timing figures for the PMC-3D code, shown in Figure 2.7, clearly indicate that the SOR Poisson solver is the main bottleneck in the PMC-3D code. Hence, a significant speedup in this part, will greatly influence the performance of the PMC-3D algorithm.

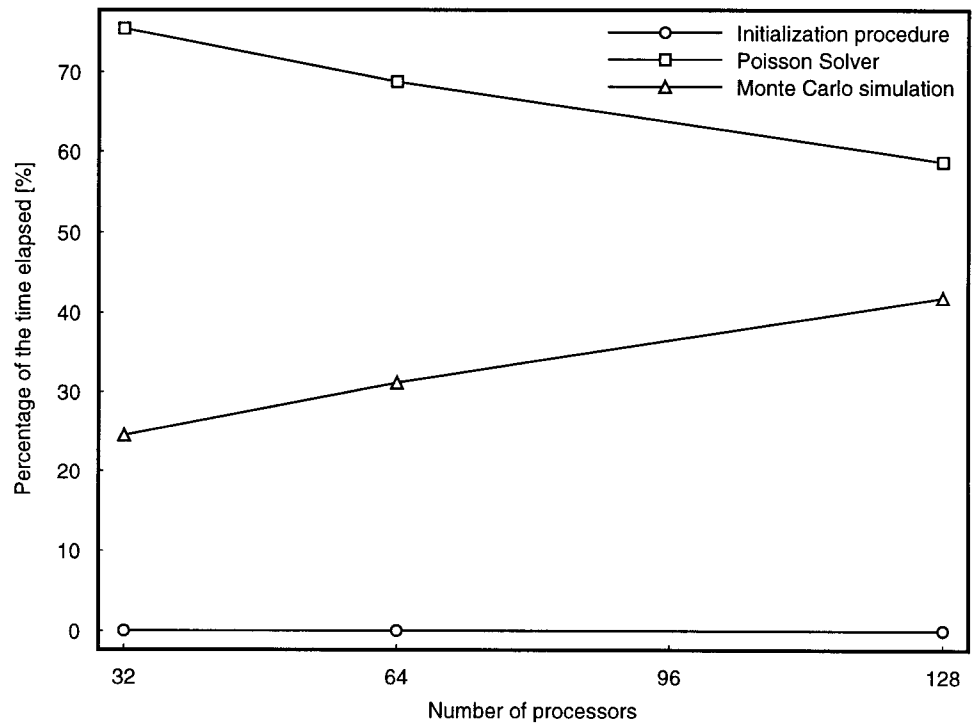


FIGURE 2.7. Workload contributions of different parts of the PMC-3D code. The device simulated is a typical MESFET device and 50000 particles are used in the simulation.

2.5. Summary

In this chapter we explained the basic principles of the Monte Carlo method and briefly discussed the parallel Monte Carlo device simulator, PMC-3D. Then we focused on the SOR Poisson solver, and provided detailed internal timings of the PMC-3D package. The Poisson solver takes around 60% to 90% of the simulation time depending on the number of carriers being simulated, hence the PMC-3D device simulator will highly benefit from a speedup achieved in this module.

3. THE MULTIGRID SOLVER

The multigrid technique is a well-established approach for solving ordinary and partial differential equations. Its main advantage over other iterative methods, e.g., the SOR, is that its convergence speed is immune to increasing grid point numbers or more accurate convergence thresholds [10, 11, 13, 16].

The well known iterative methods, Jacobi and Gauss-Seidel, were designed for solving small linear systems. A lot of variants have been proposed which are suitable for hand-held calculations. The successive overrelaxation method (SOR) achieved an improvement by a slight systematical modification, and the Chebyshev acceleration was yet another important improvement. Despite all improvements, the traditional iterative methods still exhibit decreasing convergence speeds.

The need for faster solvers was the main motive of the development of the multigrid method. Although it is referred to as “*the multigrid method*” in the literature, it has also become clear that multigrid is a family of methods, called “*multilevel techniques*” [9]. From device simulation point of view, its easily parallelizable and fast in nature, making the multigrid method one of the best choices for solving the Poisson’s equation, which takes around 60% to 90% of the PMC-3D [4] simulation time.

This chapter is organized as follows. In the first section, the basic principles of the multigrid approach are discussed, focusing on the aspects that are relevant for solving the Poisson’s equation. Section 2 concentrates on the details of our three-dimensional implementation including the parallelization and the limitations. Finally a brief summary of the discussed ideas is presented.

3.1. Principles of the Multigrid Method

In this section, we discuss the basic aspects of the multigrid method. The primary emphasis will be on the three dimensional Poisson's equation and its finite-difference discretization. For simplicity in parallel implementation, we have chosen to use homogenous, uniformly spaced grids to avoid line and/or plane relaxations. The details regarding the implementation can be found the next section.

We describe the main idea behind the multigrid approach, taking the three dimensional Poisson's equation as an example. The Poisson's equation can be expressed as follows

$$L\mathbf{u} = \mathbf{f},$$

where L represents the ∇^2 operator, \mathbf{u} is the potential distribution, and \mathbf{f} is the normalized charge distribution, $\rho(x, y, z)/\epsilon_s$. Let \mathbf{v} denote the approximation to \mathbf{u} , and \mathbf{e} denote the corresponding error, where $\mathbf{e} = \mathbf{u} - \mathbf{v}$. In this case, the residual \mathbf{r} is

$$\mathbf{r} = \mathbf{f} - L\mathbf{v},$$

where $L\mathbf{v}$ is the approximation to the forcing function \mathbf{f} . It is easy to show that the error \mathbf{e} obeys the so-called residual equation

$$L\mathbf{e} = \mathbf{r}.$$

Let $L_n\mathbf{u}_n = \mathbf{f}_n$ denote the finite difference discretization of the Poisson's equation on the grid, Ω_n and the next coarser grid be Ω_{n-1} . The simplest multigrid approach is the two level coarse grid correction. In this scheme, the residual \mathbf{r} is first transferred to the next coarser grid as

$$\mathbf{r}_{n-1} = \mathbf{I}_n^{n-1}\mathbf{r}_n,$$

where \mathbf{I}_n^{n-1} is the residual weighting or restriction which is a fine to coarse transfer operator. Then, the residual equation on the coarse level

$$L_{n-1}\mathbf{e}_{n-1} = \mathbf{I}_n^{n-1}\mathbf{r}_n$$

is solved exactly, either by means of an iterative method such as SOR or directly. L_{n-1} is some coarse grid approximation to the dense grid Laplacian L_n which corresponds to the same finite difference discretization of the problem on the coarser grid. After the residual equation is solved on the coarse level, the error is interpolated to the dense grid. This estimated error component is then added as a correction to \mathbf{v}_n as

$$\mathbf{v}'_n \leftarrow \mathbf{v}_n + \mathbf{I}_{n-1}^n L_{n-1}^{-1} \mathbf{I}_n^{n-1} \mathbf{r}_n.$$

The advantage of this scheme comes from the error smoothing effect of the relaxation operators [14, 15]. In the Fourier domain, the low frequency components of the error vector are slightly reduced while the high frequency components practically vanish in a few relaxation sweeps. This effect can be demonstrated by means of a simple one-dimensional example. The one dimensional Poisson's equation with the forcing function $\mathbf{f} = 0$ is,

$$\frac{\partial^2 \mathbf{u}}{\partial x^2} = 0, \quad u(0) = u(1) = 0$$

To demonstrate the error smoothing effect of the iterative schemes, a delta Dirac function is chosen as the initial \mathbf{v} value. After four simple Gauss Seidel iterations, Figure 3.1 clearly shows the rapid reduction of the high frequency components of the error. At this point, it is practical to define the boundary between high and low frequencies as $\pi/2$. If the error is transferred to a double spaced grid, due to *aliasing* [16], some of the low frequency components, overlap with the high frequencies as

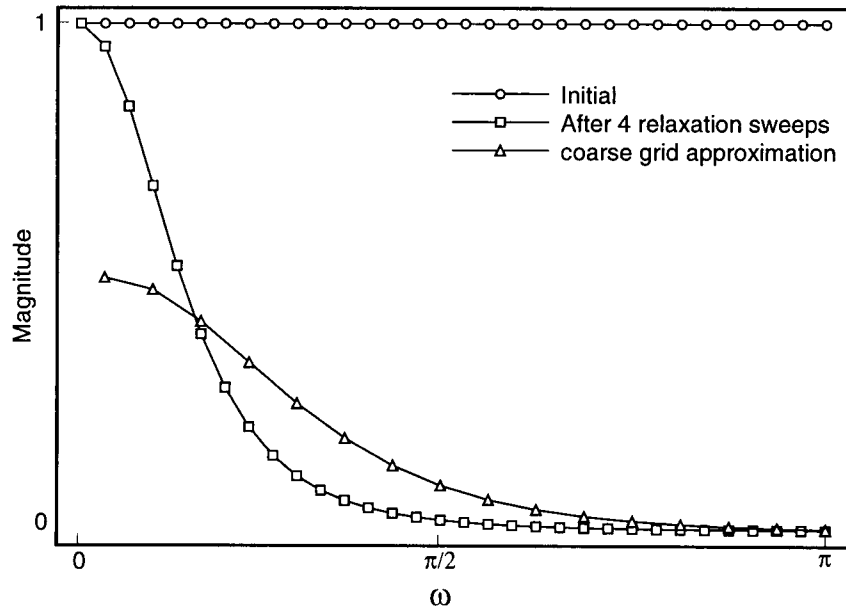


FIGURE 3.1. Error smoothing effect of the iterative schemes.

can be seen in Figure 3.1. Thus the same relaxation scheme can efficiently reduce these overlapped components on the coarse grid.

With these ideas in mind, a simple two-level coarse grid correction cycle can be described as follows:

1. Pre-smoothing: $\mathbf{v}_n \leftarrow S_n^{v_1} \mathbf{v}_n$.
2. Calculate the residual: $\mathbf{r}_n = \mathbf{f}_n - L\mathbf{v}_n$.
3. Restriction: $\mathbf{f}_{n-1} \leftarrow \mathbf{I}_n^{n-1} \mathbf{r}_n$.
4. Solve exactly on Ω_{n-1} : $\mathbf{u}_{n-1} = L_{n-1}^{-1} \mathbf{f}_{n-1}$.
5. Interpolation: $\mathbf{e}_n \leftarrow \mathbf{I}_{n-1}^n \mathbf{u}_{n-1}$.
6. Correction: $\mathbf{v}'_n \leftarrow \mathbf{v}_n + \mathbf{e}_n$.

7. Post-smoothing: $\mathbf{v}'_n \leftarrow S_n^{v_2} \mathbf{v}'_n$.

Here S_n^k denotes k relaxation sweeps of an appropriate relaxation scheme. The details about the interpolation, restriction and smoothing operators will be discussed in the next section. The notation change in steps 3, 4 and 5 is for the purpose of uniform expressions at all levels. Note that with this notation change, the equation in step 4 has the same form as the original equation, $L\mathbf{u} = \mathbf{f}$. Applying the entire procedure recursively γ times in step 4, one can produce different multigrid cycles, e.g., the V-Cycle for $\gamma = 1$ or the W cycle for $\gamma = 2$, as seen in Figure 3.2. Using W cycles with a pointwise red-black ordered Gauss-Seidel relaxation scheme and a homogenous grid with uniform grid spacings gives the best performance upgrade in a reasonable development time.

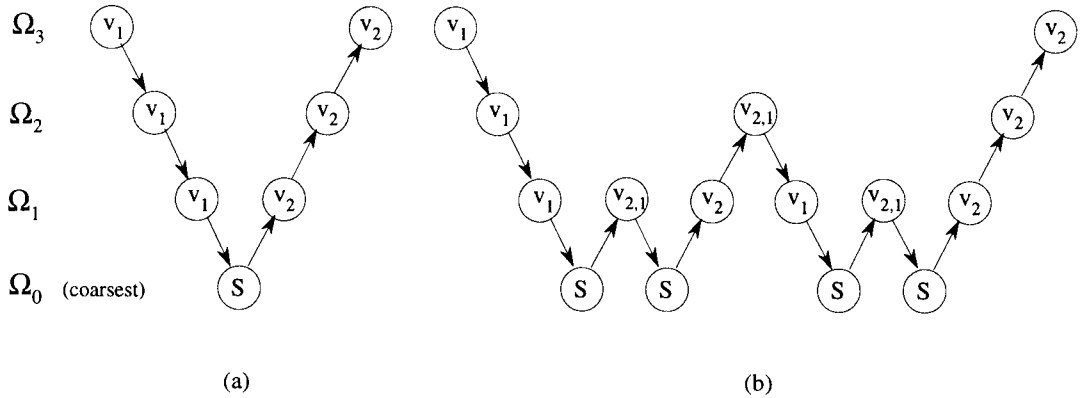


FIGURE 3.2. Two common multigrid cycles are (a) V Cycle ($\gamma = 1$) and (b) W Cycle ($\gamma = 2$). Here v_1 denotes pre-smoothing and v_2 denotes post-smoothing. Also S is the exact solution operator, \searrow is the fine to coarse grid restriction operator, and \nearrow is the coarse to fine grid prolongation operator.

3.2. Implementation Details

In this section, we discuss the implementation details of the multigrid Poisson solver. The coarsening scheme, intergrid transfer operators, relaxation scheme, discretization and the parallelization of the method are explained in the following parts.

3.2.1. Discretization

In any numerical solution of continuous equations, the formulation of a good discretization scheme is the first step. For a multigrid solver, the discrete equations need to be written for the entire set of grids with different grid spacings. The three dimensional Poisson's equation

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} = -\frac{\rho}{\epsilon_s}$$

can be discretized using finite differences and a homogenous grid with uniform spacing as

$$-\frac{1}{h_n^2} [\phi_{x-1,y,z} + \phi_{x+1,y,z} + \phi_{x,y-1,z} + \phi_{x,y+1,z} + \phi_{x,y,z-1} + \phi_{x,y,z+1} - 6\phi_{x,y,z}] = \frac{\rho_{x,y,z}}{\epsilon_s},$$

where h_n is the uniform grid spacing of the grid Ω_n . One of the problems we are facing here is the approximation of the L operator on the coarser grids [13]. The first approach is defining the equations $L_l \mathbf{u}_l = \mathbf{f}_l$ for all levels $l \in \{0, 1, \dots\}$ by the same discretization. This method is suitable for the Poisson problem. There is another method called, the Galerkin approximation, which in the finite difference case needs additional computation for defining L_{l-1} . As we are interested in solving the Poisson's equation, discretizing the problem on the entire set of grids is the suitable choice.

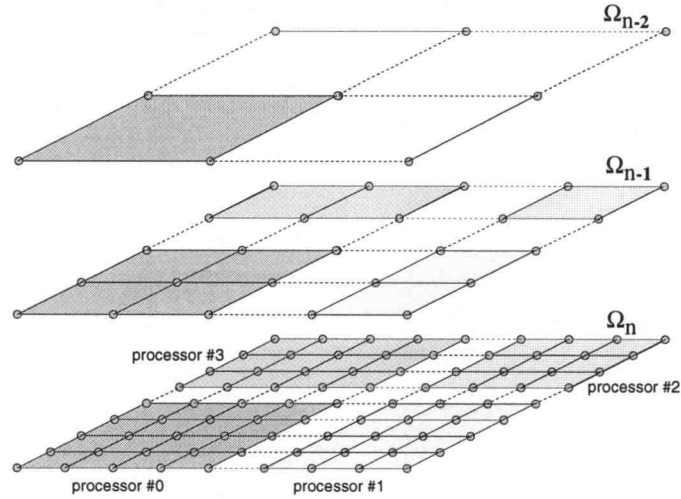


FIGURE 3.3. Two dimensional representation of the multiprocessor coarsening scheme. Here Ω_n is the densest, Ω_{n-1} is the next coarser, and Ω_{n-2} is the coarsest grid.

3.2.2. Coarsening

For the multigrid approach, the choice of the grid set is crucial. The first task is to create a hierarchical set of grids ranging from the finest grid Ω_n to the coarsest possible one, Ω_k . Here, determining the coarsest possible level is the key aspect. As long as the boundary conditions of the original grid can be represented on a coarser grid, coarsening is allowed. The representation of the boundary conditions on the coarser levels is discussed in the next part.

In our implementation, the coarsening factor we used is $1/2$, which implies that the grid spacing of the next coarser level is twice as big as the grid spacing of the relatively finer level. In Figure 3.3, a two dimensional example of the coarsening scheme we used is represented, considering the multi-processor case.

The multigrid method does not have any restrictions concerning the total number of grid-points. However, choosing the number of points of the form $2^k + 1$

for all three directions (but not necessarily with equal k values) would simplify the restriction and the prolongation operators and improve the convergence ratio of the Poisson Solver.

3.2.3. Boundary Conditions

The treatment of the boundary conditions is one of the most crucial parts of the multigrid method. For a semiconductor device simulation, there are two main types of boundary conditions. The first are Dirichlet boundary conditions, which arise around the contacts and have a fixed potential value. The second are Neumann boundaries, in which the potential value is unavailable but the value of the electrical field, hence the first derivative of the potential is fixed.

Dirichlet boundary conditions need to be mapped to all grids with at least one boundary point per contact. Let the grid point at (x_1, y_1, z_1) belong to an electrical contact with the potential value ϕ_a . Then the boundary value on the finest grid is the contact potential ϕ_a . On the coarser levels, we are trying to approximate the error on this potential value. The potential at the contact is fixed and known exactly, and thus, the corresponding error on the coarser grids must be zero, i.e.,

$$\phi_{x_1, y_1, z_1}^n = \begin{cases} \phi_a & n = 0 \text{ (finest level)} \\ 0 & n \neq 0 \text{ (on all other levels)} \end{cases}$$

Neumann boundaries are treated the same way over the entire grid set although their mapping is not as crucial as the Dirichlet boundaries [8]. For Neumann boundaries, a second order approximation is made where the first derivative of the potential is approximated as

$$\frac{1}{2} (v_{x,y,z_{\Gamma-1}} - v_{x,y,z_{\Gamma+1}}) = \varepsilon = \text{constant}$$

where z_Γ represents the point next to the Neumann boundary and ε is the electrical field at the boundary point. Hence a zero electrical field at the boundary implies $v_{x,y,z_\Gamma+1} = v_{x,y,z_\Gamma-1}$ as the boundary potential.

3.2.4. Relaxation Method

The main goal of the relaxation scheme is to reduce the high frequency components of the error on any given grid. There can be several suitable relaxation schemes for a specific problem depending on the boundary conditions and the coarsening method.

The efficiency of a relaxation scheme can be measured by the smoothing factor [14, 15]. For a cubic grid with $N \times N \times N$ grid points with periodic boundary conditions, the Fourier transform of the error \mathbf{e} is given by

$$\mathbf{e}_{x,y,z} = \sum_{r,s,t=-N/2+1}^{N/2} c(\theta_r, \theta_s, \theta_t) \exp [i(\theta_r x + \theta_s y + \theta_t z)],$$

where $\theta_r = r\pi/N$, $\theta_s = s\pi/N$, $\theta_t = t\pi/N$, and c is the magnitude of the frequency component of the error for a given frequency. The amplification factor of the $\theta_{\{r,s,t\}}$ component due to one relaxation is,

$$\mu(\theta) = \left| \frac{\bar{c}(\theta)}{c(\theta)} \right|,$$

where $\theta = (\theta_r, \theta_s, \theta_t)$, and \bar{c} represent the frequency components of the error after the relaxation sweep. Finally the smoothing factor is defined by

$$\mu = \max_{\rho\pi \leq |\theta| \leq \pi} \mu(\theta),$$

where ρ is the grid coarsening factor. Here a double coarsening scheme implies $\rho = 1/2$.

In our implementation we chose to use a red-black ordered pointwise Gauss-Seidel relaxation scheme, which has a typical smoothing factor $\mu \simeq 1/2$, [16] over the cubic grid discussed above. This smoothing factor implies that the high frequency components of the error are reduced by almost an order of magnitude in three relaxation sweeps. This smoothing rate is achieved only for the non-degenerate case where the grid spacings in all three dimensions are the same. The smoothing properties of a pointwise relaxation scheme are very poor if a standard coarsening on a nonuniform grid is used [9, 11].

The reason for the poor smoothing effect comes from the fact that a pointwise relaxation scheme has a smoothing effect only with respect to the direction that has the smallest grid spacing. Thus, for a decent smoothing effect, according to the various configurations of the grid spacings, line and/or plane relaxations are required, which are difficult to implement in parallel. As the multigrid solver is designed to be a replacement for the former SOR solver, we chose to use a pointwise red-black ordered Gauss-Seidel relaxation scheme and restricted the grids to be homogenous and uniformly spaced along all three dimensions.

3.2.5. Restriction and Prolongation

Another important component of the multigrid method is the restriction and prolongation operators. After generating the hierarchical grid set, the next step is designing the tools for residual transfers from coarse to fine grid and the opposite way for the error.

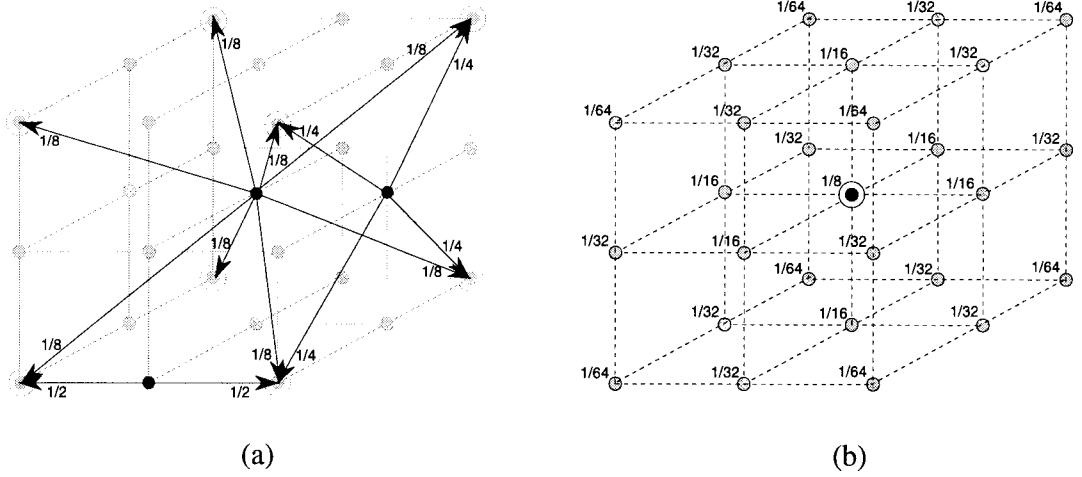


FIGURE 3.4. The intergrid transfer operators. a) Prolongation: The arrows denote the coarse grid points to be used for interpolating the dense grid point. The numbers attached to the arrows denote the contribution of the specific coarse grid point. b) Restriction: A 27-point full weighting scheme is used. The number in front of each grid point denotes its weight in this operation.

The prolongation operator we used is a modified version of the two dimensional *nine point prolongation* [13], symbolized by the stencil

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}.$$

The three cases for the three dimensional prolongation operation are shown in Figure 3.4a. The arrows denote the contributing coarse grid points, where the attached numbers are the corresponding weighting factors.

The restriction operator is a little more difficult to implement. There are two different useful approaches, namely the full weighting and the half weighting restriction [13, 16]. In our experience, a full weighting residual transfer operator is

necessary for a stable solution. According to [13], the two-dimensional *full weighting restriction*, also called *nine point restriction* can be symbolized by the stencil

$$\frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix},$$

and is the adjoint of the *nine point prolongation*. For our three dimensional problem, the dense grid points that take part in the regular full weighting scheme are listed with the corresponding weighting factors in Figure 3.4b. Although there are 27 points to be considered, the nature of the red/black ordered Gauss-Seidel relaxation scheme allows us to concentrate on 13 of those points as the residual values corresponding to the last updated color are always zero.

3.2.6. Parallelization

Several parallel implementations of the multigrid method has been reported in the literature [16–19]. Our parallelization of the multigrid code is essentially the same as the former SOR implementation described in Chapter 2. The partitioning and the communication routines are extended to service the hierarchical grids, hence the communication pattern and the partitioning logic is preserved. In Figure 3.3, the partitioning and the coarsening of the grid is represented, using a two dimensional example. Since the red-black ordered Gauss-Seidel relaxation operator is simply the SOR with $\omega = 1$, the communication pattern of the smoothing operator also remains unchanged [4, 5].

As in the SOR solver, each relaxation sweep consists of two half sweeps corresponding to the two orderings of the grid points. Each processor updates the potential values of the grid points belonging to the subgrid mapped to its memory.

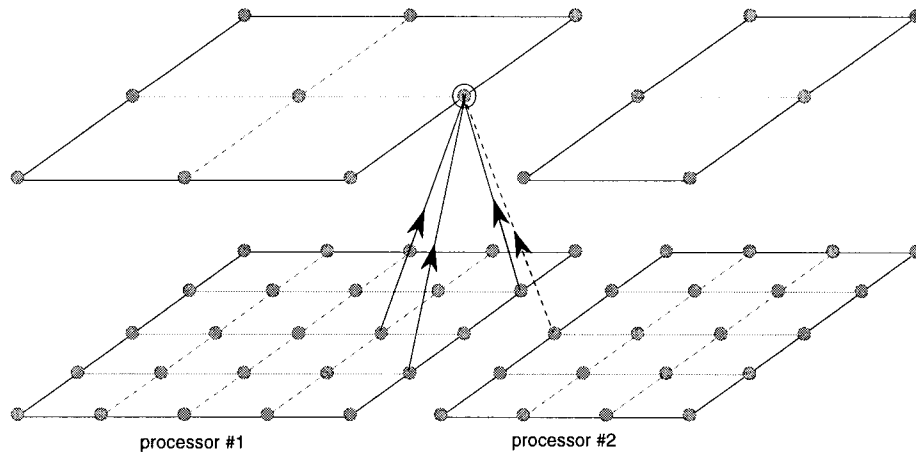


FIGURE 3.5. Two dimensional representation of the parallel implementation of the restriction operation. The dashed arrow denotes the communication operation that needs to be done for that specific case, before the residual restriction is performed.

Before each half sweep, the processors need to communicate via message passing with its neighboring processors. This way the potentials of the oppositely colored grid points external to the processor's subgrid are obtained. After the smoothing operation is performed, the residual values are calculated. The residual values of the last updated grid set is zero. Before the residual restriction is performed, each processor again communicates with its neighboring processors to obtain the non-zero residual values of the grid points external to its subgrid. This way a correct restriction to the coarser levels is achieved. A two dimensional representation of the parallel implementation of the restriction operator is shown in Figure 3.5.

The same situation is valid for the prolongation operator as well. The prolongation operation is performed after either a post-smoothing or an exact solution operation. In our implementation, these two operations, although different in functionality, are very similar. The exact solution operation is nothing but the former

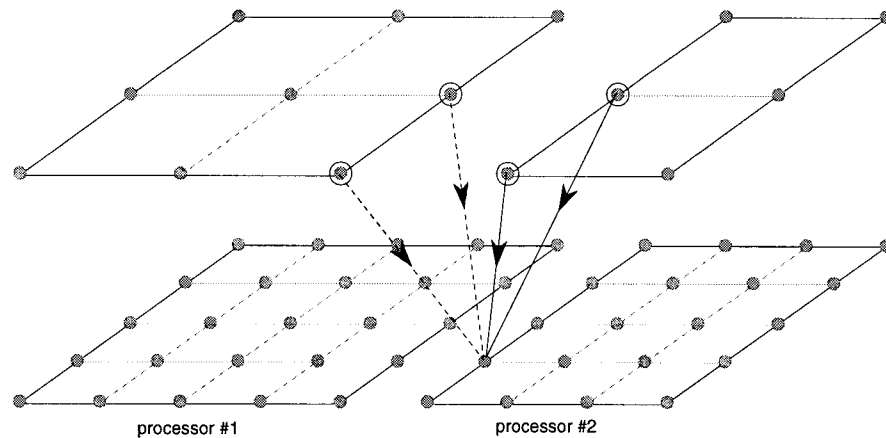


FIGURE 3.6. Two dimensional representation of the parallel implementation of the prolongation operation. The dashed arrows denote the communication operation that needs to be done for that specific case, before the error prolongation is performed.

SOR solver applied to the coarsest level. Before the prolongation is performed, each processor communicates with its neighboring processors to obtain the updated potentials of the grid points external to its subgrid. Then the prolongation operation is performed and the error is interpolated to the finer levels. A two dimensional representation of the parallel implementation of the prolongation operator can be seen in Figure 3.6.

3.3. Summary

In this chapter, we described the principles of the multigrid technique, emphasizing the solution of the Poisson's equation. Then we explained the implementation details, discussing our implementation choices and the reasons behind them. The simulation results and the discussions are presented in the next chapter.

4. RESULTS AND DISCUSSIONS

In this chapter we present the results of our experiments in simulating a MESFET device structure illustrated in Figure 4.1. We have executed the PMC-3D code with both the SOR and the multigrid solver for 100 iterations to compare their timings. The grid we used is a $129 \times 65 \times 33$, homogenous grid with uniform spacings in all three dimensions.

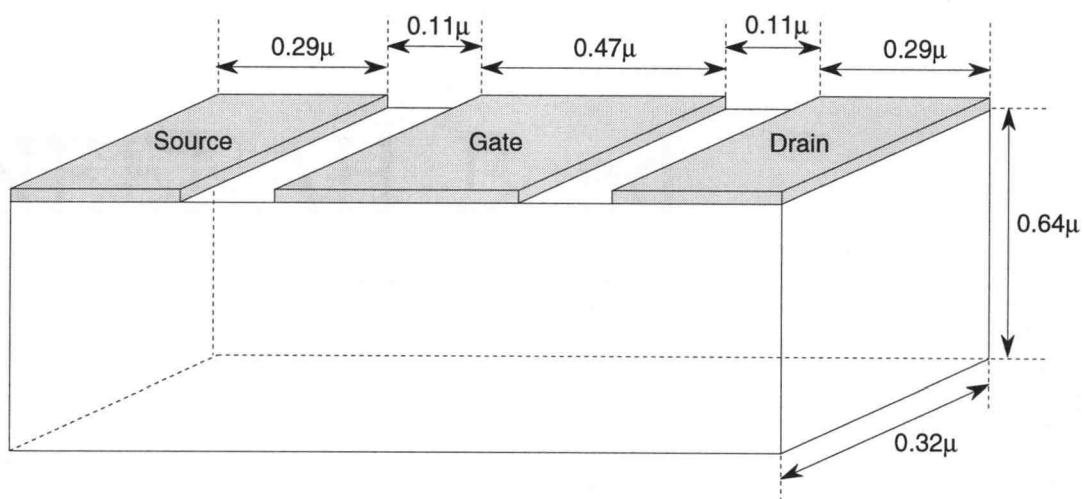


FIGURE 4.1. The MESFET structure used as a model problem for the simulations.

4.1. Timings for the Serial and Parallel PMC-3D

The timings and speedups of the serial PMC-3D code with the SOR solver and with the multigrid solver are presented in Table 4.1. The Monte Carlo simulation is performed on 32000 particles and the timings are in seconds. As can be seen from the table, the serial PMC-3D with the MG solver is 5 to 15 times faster than the PMC-3D with the SOR solver, while the Poisson solver alone has speedup values

TABLE 4.1. The timings of the PMC-3D device simulator with SOR and MG solvers. The simulation is run for 100 time steps with different convergence thresholds on a $129 \times 65 \times 33$ homogenous grid with uniform grid spacings on a single HP 712/80 workstation. 32000 particles are simulated. The timings are in seconds.

Threshold	PMC-3D with SOR		PMC-3D with MG		Speedup	
	Poisson	Total	Poisson	Total	Poisson	Total
10^{-3}	14,879.82	15,595.59	2,145.84	3,117.65	6.93	5.00
10^{-6}	76,208.28	77,029.05	5,664.77	6,576.83	13.45	11.71
10^{-9}	153,118.90	153,952.04	9,779.32	10,728.29	15.65	14.35
10^{-12}	225,867.00	226,735.04	14,160.49	15,124.02	15.95	14.99

TABLE 4.2. The timings of the PMC-3D device simulator with SOR and MG solvers. The simulation is run for 100 time steps with different convergence thresholds on a $129 \times 65 \times 33$ homogenous grid with uniform grid spacings on a 32 node nCUBE multiprocessor. 20000 particles are simulated. The timings are in seconds.

Threshold	PMC-3D with SOR		PMC-3D with MG		Speedup	
	Poisson	Total	Poisson	Total	Poisson	Total
10^{-3}	2917.611	3340.020	596.367	1121.035	4.89	3.05
10^{-6}	15093.156	15515.990	2064.199	2589.482	7.31	5.99
10^{-9}	31167.143	31653.002	3486.319	4011.031	8.94	7.89
10^{-12}			4927.825	5453.801		

between 7 to 16 depending on the convergence threshold. As stated in Chapter 2, the Poisson solver takes most of the simulation time. Thus, the actual speedup observed in the PMC-3D device simulator is close to that of the Poisson module. The same set of timings are taken on a 32 node nCUBE multiprocessors with 20000 particles and are presented in Table 4.2. Overall speedup values ranging from 3 to 8 are obtained corresponding to different convergence thresholds, while the Poisson solver alone has speedup values between 5 to 9.

4.2. Discussion

The difference in speedup values between the serial and the parallel case, arises from the fact that the communication load for the multigrid solver is higher than that of SOR. Although the amount of data transferred between processors decreases with increasing grid-spacing in the multigrid method, the number of communication attempts and the number of iterations are generally higher than those in the SOR solver.

The communication workload for both the SOR and MG solvers is estimated, assuming that there is a subgrid in the device which has six other neighboring subgrids. In this case, using the average communication startup time and the data transfer rate figures for the nCUBE2 multiprocessor system at Sandia Laboratories, the estimated communication volumes for both the SOR and MG solvers is illustrated in Figure 4.2. From the Figure, it is obvious that the exact solution operation performed on the coarsest level in the MG solver is the main cause for the excessive communication load in the MG solver.

As mentioned in Chapter 3, the computation time of the multigrid solver increases only linearly with respect to the decrease in the convergence threshold.

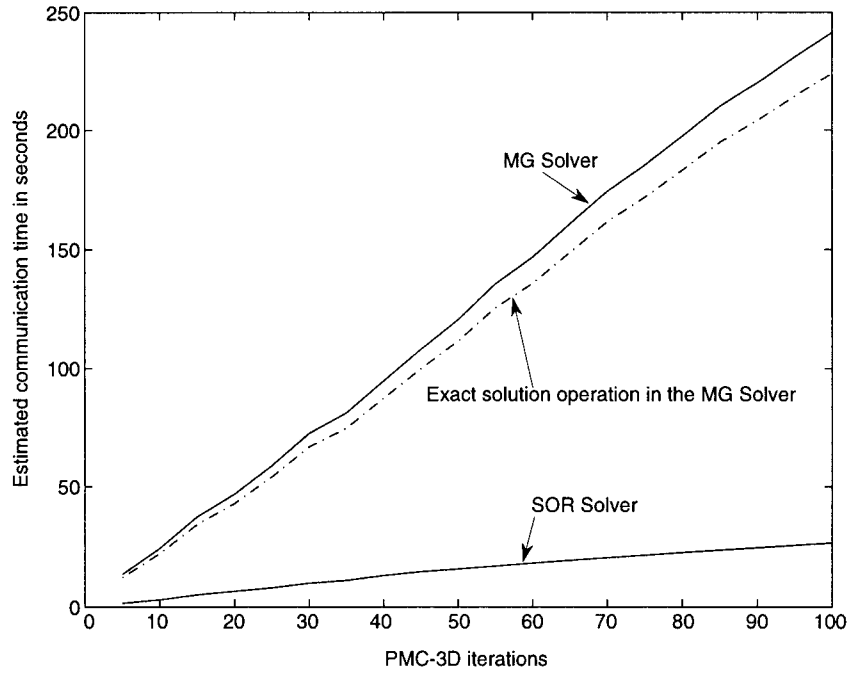


FIGURE 4.2. The communication overhead for the multigrid solver.

However, the computation time of the SOR solver tends to grow exponentially. This effect can be seen in Figure 4.3 and 4.4, in which we plot the computation time as a function of the convergence threshold.

We have presented our experiments in embedding the MG solver in place of the SOR solver for solving the Poisson's equation. We obtained speedups between 6 to 15 for the serial code and 4 to 9 for the parallel code. The simulations were performed on a $129 \times 65 \times 33$ homogenous grid with uniform grid spacings in order to simulate the MESFET structure whose exact dimensions are shown in Figure 4.1. The speedups of the Multigrid Solver developed are comparable with the ones presented by Saraniti et al in [8]. The speedups achieved in the Poisson solver module effectively decreased the total simulation time as predicted in Chapter 2.

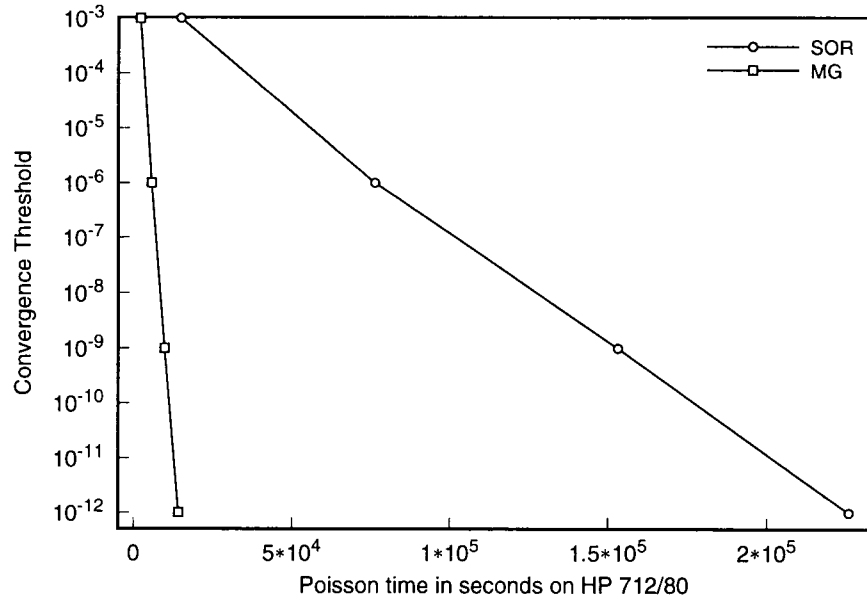


FIGURE 4.3. The computation time of the Poisson solver versus convergence threshold for the serial code running on a HP 712/80, for 100 PMC-3D iterations.

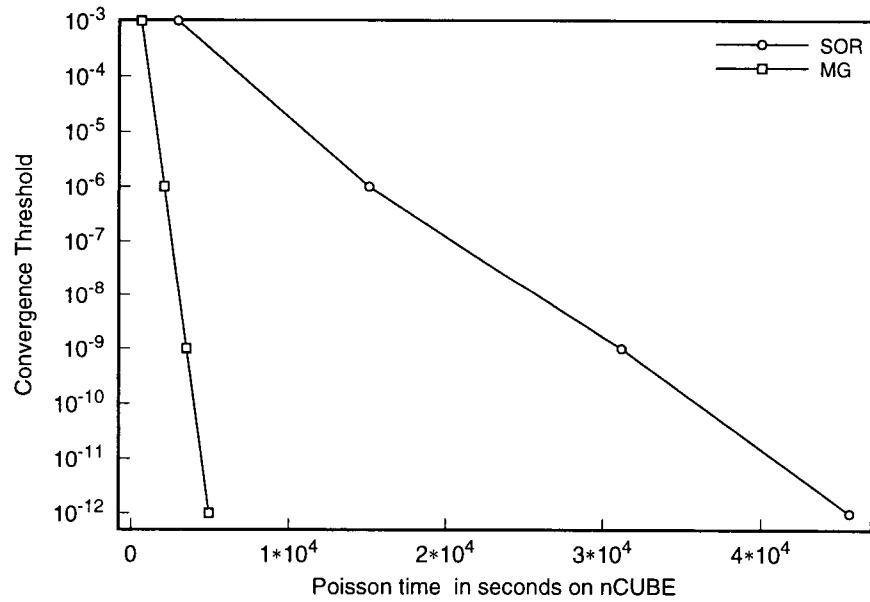


FIGURE 4.4. The computation time of the Poisson solver versus convergence threshold for the serial code running on a 32 node nCUBE multicomputer, for 100 PMC-3D iterations.

4.3. Future Work

The current MG implementation can be improved to serve nonuniform and inhomogeneous grids. As stated in Chapter 3, the requirement for uniform homogeneous grids arises from the fact that pointwise Gauss-Seidel relaxation has a smoothing effect with respect to the “dominant direction” of the operator which is the direction with the smallest grid-size. To achieve good smoothing rates for nonuniform grids, plane and/or line relaxations may be necessary. Plane and line relaxations demand excessive communication volumes in the parallel implementation, however, Thole and Trottenberg demonstrated [21] that a two dimensional multigrid solver can be efficiently used for plane relaxation. Thus, this might be the next development stage for the Poisson solver.

The speedup degradation effect of the excessive communication workload of the exact solution operation in the MG solver is demonstrated in Figure 4.2. One way to decrease the communication workload in the MG solver is, performing the exact solution operation on a single processor instead of a set of processors. Although this approach will introduce some speedup degradation, it will not be as high as the excessive communication workload of the current implementation.

Another problem in parallel multigrid implementations is the number of idle processors in the coarser levels. Chan and Tuminaro address this issue in [20], hence this direction is yet another possibility for improving the current multigrid implementation.

BIBLIOGRAPHY

- [1] C. Jacoboni and P. Lugli. *The Monte Carlo Method for Semiconductor Device Simulation*. Vienna, Austria: Springer-Verlag, 1989.
- [2] C. Moglestue. *Monte Carlo Simulation of Semiconductor Devices*. Chapman & Hall, 1993.
- [3] K. Kometer, G. Zandler, and P. Vogl. Lattice-gas cellular-automaton method for semiclassical transport in semiconductors. *Physics Reviews B*, 46:1382–1394, July 1992.
- [4] U. A. Ranawake, C. Huster, P. M. Lenders, and S. M. Goodnick. PMC-3D: A parallel three-dimensional Monte Carlo semiconductor device simulator. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(6):712–724, 1994.
- [5] S. S. Pennathur and S. M. Goodnick. Monte Carlo investigation of three-dimensional effects in sub-micron GaAs MESFETs. *Inst. Phys. Conf. Ser.*, No 141, Chapter 7, 1995.
- [6] U. A. Ranawake *Cluster partitioning approaches to parallel Monte Carlo simulation on multiprocessors*. PhD thesis, Oregon State University, Oregon, 1992
- [7] S. S. Pennathur *Monte Carlo device modelling applications on parallel computers*. PhD thesis, Oregon State University, Oregon, 1995
- [8] M. Saraniti, A. Rein, G. Zandler, P. Vogl and P. Lugli. An efficient multigrid Poisson solver for device simulations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(2):141–150, 1996.
- [9] A. Brandt. Multigrid techniques: 1984 Guide with applications to fluid dynamics. Monograph 85, Gesellschaft für Mathematik und Datenverarbeitung mbH Bonn, Postfach 1240, D-5205 St. Augustin 1, Germany, 1984.
- [10] A. Brandt. Rigorous quantitative analysis of multigrid, I: Constants coefficients two-level cycle with L_2 -norm. *SIAM Journal on Numerical Analysis*, 31(6):1695–1735, 1994.
- [11] K. Stüben and U. Trottenberg. Multigrid methods: Fundamental algorithms, model problem analysis and applications. In W. Hackbusch and U. Trottenberg, editors, *Multigrid Methods, Proceedings of the Conference*, Lecture Notes in Mathematics, Number: 960, pages 1–176, Köln-Porz, November 23–27, 1981, Berlin: Springer-Verlag.

- [12] A. Brandt. Guide to multigrid development. In W. Hackbusch and U. Trottenberg, editors, *Multigrid Methods, Proceedings of the Conference*, Lecture Notes in Mathematics, Number: 960, pages 220–312, Köln-Porz, November 23–27, 1981, Berlin: Springer-Verlag.
- [13] W. Hackbusch. *Multi-Grid Methods and Applications*. Berlin: Springer-Verlag, 1985.
- [14] J. Kuo and C. Levy. Two-color Fourier analysis of the multigrid method with red-black Gauss-Seidel smoothing. *Applied Mathematics and Computation*, 20:69–87, 1989.
- [15] I. Yavneh. Multigrid smoothing factors for red-black Gauss-Seidel relaxation applied to a class of elliptic operators. *SIAM Journal on Numerical Analysis* 32(4):1126–1138, 1995.
- [16] A. Brandt. Multigrid solvers on parallel computers. In M. H. Schultz, editor, *Elliptic Problem Solvers*, pages 39–84 New York, Academic Press, 1981.
- [17] O. A. McBryan, P. O. Fredericson, J. Linden, A. Schüller, K. Stüben, C. A. Thole and U. Trottenberg. Multigrid methods on parallel Computers - A survey of recent developements. *IMPACT of Computing in Science and Engineering*, 3:1–75, 1991.
- [18] L. R. Matheson and R. E. Tarjan. A critical analysis of multigrid methods on massively parallel computers. Technical Report CWI Tract 103, Center for Mathematics and Computer Science, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, 1993.
- [19] A. Greenbaum. A multigrid method for multiprocessors. *Applied Mathematics and Computation*, 19:75–88, 1986.
- [20] T. F. Chan, R. S. Tuminaro. Design and implementation of parallel multigrid algorithms. Technical Report 87.21, Research Institute for Advanced Computer Science, NASA Ames Research Center Moffet Field, CA 94035, 1987.
- [21] C. A. Thole and U. Trottenberg. Basic smoothing procedures for the multigrid treatment of elliptic 3D operators. *Applied Mathematics and Computation*, 19:333–345, 1986.