NUMERICAL METHODS OF SIMULATING SURFACE PHASE TRANSITIONS ON ATOMISTIC LEVEL

Charun Pidduangkeaw

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Master Degree in Applied Mathematics

Suranaree University of Technology

Academic Year 2011

วิธีเชิงตัวเลขของการจำลองการเปลี่ยนสถานะของพื้นผิวในระดับอะตอม

นายจรัล ปิดดวงแก้ว

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2554

NUMERICAL METHODS OF SIMULATING SURFACE PHASE TRANSITIONS ON ATOMISTIC LEVEL

Suranaree University of Technology has approved this thesis submitted in partial fulfillment of the requirements for a Master's Degree.

Thesis Examining Committed

_____
(Asst. Prof. Dr. Eckart Schulz)
Chairperson

_____
(Prof. Dr. Sergey Meleshko)
Member (Thesis Advisor)

_____
(Assoc. Prof. Dr. Nikolay Moshkin)
Member

_____
(Assoc. Prof. Dr. Kenneth J. Haller)
Member

_____
(Dr. Sayan Kaennakham)
Member

_____
(Prof. Dr. Sukit Limpijumnong)
Vice Rector for Academic Affairs

_____
(Assoc. Prof. Dr. Prapun Manyum)
Dean of Institute of Science

NUMERICAL METHODS OF SIMULATING SURFACE PHASE TRANSITIONS ON ATOMISTIC LEVEL

Charun Pidduangkeaw

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Master Degree in Applied Mathematics

Suranaree University of Technology

Academic Year 2011

จรัล ปิดดวงแก้ว : วิธีเชิงตัวเลขของการจำลองการเปลี่ยนสถานะของพื้นผิวในระดับอะตอม (NUMERICAL METHODS OF SIMULATING SURFACE PHASE TRANSITIONS ON ATOMISIC LEVEL) อาจารย์ที่ปรึกษา : ศาสตราจารย์ ดร.เซอร์เก เมเลชโก, 87 หน้า.

วิทยานิพนธ์นี้ศึกษาและพัฒนาวิธีทางตัวเลข สำหรับศึกษาแบบจำลองการเปลี่ยนสถานะของอะตอม โดยการศึกษาดังกล่าวประกอบด้วยสองขั้นตอนดังนี้คือ ขั้นตอนที่หนึ่งได้แก้ระบบสมการไม่เชิงเส้น ซึ่งได้ทดลองเปรียบเทียบวิธีทางตัวเลขด้วยกันสามวิธี และจากผลการศึกษาและเปรียบแล้ววิธีเกาซ์-ไซเดิลเป็นวิธีที่จะถูกนำไปใช้ในขั้นที่สองต่อไป โดยขั้นที่สองนั้นได้ศึกษาเกี่ยวกับการเคลื่อนที่ของอะตอม และสุดท้ายได้พัฒนาแบบจำลองการเคลื่อนที่ของอะตอมและแสดงในรูปแบบของเอนิเมชัน

สาขาวิชาคณิตศาสตร์ ลายมือชื่อนักศึกษา_____

ปีการศึกษา 2554 ลายมือชื่ออาจารย์ที่ปรึกษา_____

ลายมือชื่ออาจารย์ที่ปรึกษาร่วม_____

CHARUN PIDDUANGKEAW : NUMERICAL METHODS OF SIMULATING SURFACE PHASE TRANSITIONS ON ATOMISTIC LEVEL. THESIS ADVISOR : PROF. SERGEY MELESHKO, Ph.D. 87 PP.

NUMERICAL METHODS/ SYSTEMS OF NONLINEAR EQUATIONS/ ITERATIVE METHODS/ THE GAUSS-SEIDEL METHOD

This thesis is devoted to the development and evaluation of numerical methods for simulation of a phase transition based on a charge compensation model. The algorithm for the study consists of two steps. In the first step, a system of nonlinear equations has to be solved. For solving this system, we explore three iterative methods. The Gauss-Seidel method is chosen as the result of this study. In the second step, the movement of defect atoms is considered. A model of the motion of atoms is developed. The animated results are presented as a wmv-file.

School of Mathematics Student's Signature_____

Academic Year 2011 Advisor's Signature_____

Co-advisor's Signature_____

ACKNOWLEDGEMENTS

"Life is learning" for overcoming the obstacles on our journey. Almost five years at SUT is my great teacher who taught me a lot to walk on and never give up and finally achievement will be reached.

Firstly I would like to express my sincere gratitude and deep appreciation to my advisor Prof. Dr. Sergey Meleshko who told me "go on" "don't give up". His many advices, support, patient help were very valuable and without him this thesis would not have come to a success.

I am profoundly grateful to Assoc. Prof. Dr. Anatoli V. Meleshko, my thesis co-advisor for his collaboration, guidance and advice concerning this thesis.

I would like to express my appreciation to Assoc. Prof. Dr. Prapasri Asawakun, Asst. Prof. Dr. Eckart Schulz, Assoc. Prof. Dr. Nikolay Moshkin, Asst. Prof. Dr. Jessada Tanthanuch, Ass. Prof. Dr. Arjuna Peter Chayasena, Assoc. Prof. Dr. Kenneth J. Haller, for teaching and helping me during the course of my studies at Suranaree Universityy of Technology (SUT).

I would like to thank all my friend for helping, encouraging and supporting me during my five years here. The many experiences together will not vanish from my memory. Every moment of activity will be remembered forever…

Charun Pidduangkeaw

CONTENTS

Page

CONTENTS (Continued)

Page

CONTENTS (Continued)

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

CHAPTER I

INTRODUCTION

One of the definitions of nanotechnology is the ability to manipulate matter atom by atom. However to create microscopic or even macroscopic objects one has to rely on “mass producing” technologies that are based on self-assembly or self-organization phenomena.

Such processes can usually be described as a phase transition from a disordered to an ordered state. There are only a few examples in which such self-organization can be observed on atomic scale. One class of such systems is the class of surfaces of Pb and Sn on Ge(111) and Si(111). Preparation of ultra thin layers of Pb or Sn on these surfaces (1/3 of a monolayer) leads to formation of quasi-two-dimensional binary systems in which atoms from the substrate become substitutional defects.

The development of scanning tunneling microscopy (STM) allows direct observation of atomic arrangements and defect organization. It was discovered in Plummer et al.(2001) that surfaces of Pb/Ge(111) and Sn/Ge(111) undergo a phase transition from $\left(\sqrt{3}\times\sqrt{3}\right)$ to $\left(3\times3\right)$. Such observation was possible by using variable temperature of STMs. Later on, Melechko et al. (2001) and Ottaviano et al. (2000) have observed that substitutional defects undergo a disorder-order phase transition that accompanies $\left(\sqrt{3}\times\sqrt{3}\right)$ to $\left(3\times3\right)$ occurances. These observations can be modeled by lattice mediated defect-defect interaction.

The study of microscopic properties of phase transitions in low-dimensional systems provides an understanding of the fundamental aspects of systems of interacting particles. Phase transitions are strongly affected by defects, especially in systems with lower dimensionality. In quasi-one-dimensional 1D or 2D systems that exhibit a charge density wave (CDW) transition, a small proportion of microscopic disorder can control the global properties due to the collective nature of the phenomena. Defects cause pretransitional effects, inducing the formation of the CDW. It has been speculated that the interaction of mobile defects with the CDW leads to alignment of defects with the CDW, or formation of defect density waves. In this dynamic picture the distribution of defects is neither random nor static; instead defects align their position to optimize the energy of the pinned CDW.

The symmetry lowering phase transition $\left(\sqrt{3}\times\sqrt{3}\right)\Leftrightarrow\left(3\times3\right)$ in Pb/Ge(111), Sn/Ge(111) and similar systems has been a subject of extensive studies. These are quasi-two-dimensional systems composed of an ultra-thin metal film on the surface a semiconductor. At room temperature (RT), one-third of monolayer of Sn is arranged in a $\left(\sqrt{3}\times\sqrt{3}\right)R\ 30°$ structure on Ge(111) [referred to as a $\left(\sqrt{3}\times\sqrt{3}\right)$ structure], with Sn atoms occupying the T_4 sites of the Ge(111) substrate, as shown in Fig. 1. When the temperature is lowered, new $\left(3\times3\right)$ diffraction spots gradually appear in addition to the existing $\left(\sqrt{3}\times\sqrt{3}\right)$ spots in a low-energy electron-diffraction (LEED) pattern. Low-temperature (LT) scanning tunneling microscope (STM) images show $\left(3\times3\right)$ hexagonal (filled states) and honeycomb (empty states) complimentary patterns of bright atoms at biases of opposite sign. The $\left(3\times3\right)$ and $\left(\sqrt{3}\times\sqrt{3}\right)$ unit cells are indicated in Figure 1.1 The STM observations also display the presence of point

defects in these surfaces, the majority of which are substitutional atoms (indicated in Figure 1.1 from the substrate with vacancies constituting the rest. In Sn/Ge(111), the density of the defects is in the range from 2% to 4% due to the preparation procedure, while in Sn/Si(111) and Pb/Ge(111) their density can be varied in a very wide range.

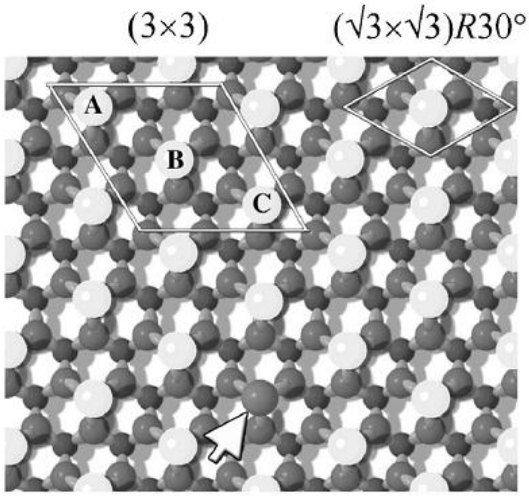


Figure 1.1 Ball model of Sn/Ge(111) structure.

CHAPTER II

MATHEMATICAL FORMULATION OF THE PROBLEM

2.1 Charge Compensation Model (CCM)

The charge-compensation model is based on four assumptions.

The first is that the charge on a lattice site is proportional to the sum of the charges on its nearest neighbors. Consider a two-dimensional triangular lattice as shown in Figure 2.1

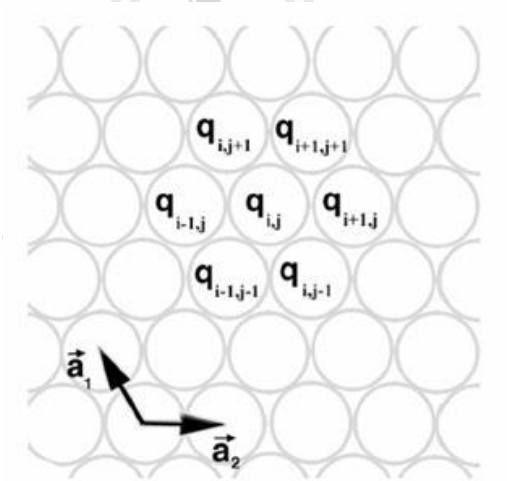


Figure 2.1 Diagram of the charge-compensation model.

The unit vectors displayed in this figure define the $\left(\sqrt{3}\times\sqrt{3}\right)$ periodicity (to be consistent with the experimental observations where the notations for superstructures are chosen with respect to bulk). Suppose that nearest-neighbor (NN) interaction is

such that the charge on site (i, j), indicated in Figure 2.1, is determined by the following equation:

$$q_{i,j} = -R(T) \cdot \sum_{k,l=NN's}^{6} q_{k,l} \ . \tag{1}$$

Here $R(T)$ is the charge-compensation factor (CCF), a free parameter that we assume is a monotonous function of temperature. The summation goes over the six nearest neighbors shown in Figure 2.1 In other words, when a charge is placed on one of the atoms, its nearest neighbors will try to screen or "compensate" for its charge.

The second premise is that the absolute value of charge on any lattice site has a saturation value. Coulomb repulsion should make it increasingly more difficult to add charge to one atom. This can be accounted for by adding a saturation term to Equation(1):

$$q_{i,j} = -R(T) \cdot \sum_{k,l=NN's}^{6} q_{k,l} + s(q_{i,j}) \tag{2}$$

where $s(q)$ must be an odd function. The first nonlinear term in the polynomial that can be used for this purpose is cubic:

$$s(q) = a \cdot q^3 . \tag{3}$$

Here a is a parameter that is assumed to be small and positive $(0 < a \leq 1)$.

The third assumption is that the charge on the lattice site corresponding to a defect (Ge substitutional atom or vacancies) is fixed, the same for all defects of one type (positive for Ge defects and negative for vacancies), and independent of its position in the lattice, the temperature and the defect density.

Finally, *the fourth* and last assumption is that the total charge of the system is always zero.

2.2 Equivalent Formulation of the CCM in the Framework of Ginzburg-Landau Theory

We can approach the structural phase transition from the point of view of the Ginzburg-Landau (G-L) theory of phase transitions. McMillan has applied G-L theory to the CDW phase transitions in transition-metal dichalcogenides. His results have been used for calculations of STM images in layered compounds and comparison with experimental results. In the following we will present similar considerations but restricted to a lattice .

In order to describe the CCM in the framework of G-L theory, Consider a system that has the following free-energy dependence on parameters $\{q_{i,j}\}$, the charge on each lattice site (i, j):

$$F = \sum_{i,j}\left(q_{i,j}^2 + R(T) q_{i,j} \sum_{k,l=NN_{i,j}} q_{k,l} - a\ q_{i,j}^4 \right)$$

$$+ \sum_{i,j\_defect}\left(\left(q_{i,j} + d\right)^2 + R(T) q_{i,j} \sum_{k,l=NN_{i,j}} q_{k,l} - a\ q_{i,j}^4 \right). \qquad (4)$$

The first summation goes over all atoms in the lattice. The internal sum runs over the next-nearest neighbors of the (i, j)th site (six in case of a triangular lattice). We proceed by solving for the order parameters $q_{i,j}$, which minimize the free energy. This can be done by solving a system of equation:

$$\frac{\partial F}{\partial q_{k,l}} = 2q_{k,l} + 2R(T) \sum_{\alpha,\beta=NN_{k,l}} q_{\alpha,\beta} - 4a\ q_{k,l}^3 = 0, \tag{5}$$

$$\frac{\partial F}{\partial q_{k,l\_defect}} = 2\left(q_{k,l\_defect} + d\right) + 2R(T) \sum_{\alpha,\beta=NN_{k,l\_defect}} q_{\alpha,\beta} - 4a\ q_{k,l\_defect}^3 = 0. \tag{6}$$

Essentially, the expressions (5), (6) mean that the CCM described in 2.1 is just a minimization procedure for the free-energy Equation(4).

CHAPTER III

SOLUTION METHODS

There are many numerical methods for solving the above equations such as The Jacobi method, The Gauss-Seidel method and also several ways of iterative methods. They will be used in the thesis for studying the Model of Charge Density Wave (CDW) transition in a 2-dimensional system with defects.

Here these methods for solving linear and nonliner systems of equations are briefly described.

3.1 The Jacobi Method

The Jacobi method is motivated by the following observation. Let A have nonzero diagonal elements. Then the diagonal part D of A is a nonsingular matrix and the system

$$Ax = b \tag{7}$$

can be rewritten as

$$Dx + (L + U)x = b. \tag{8}$$

Consequently,

$$x = D^{-1}\left[(-L - U)x + b\right]. \tag{9}$$

Here L is lower triangular (has elements only the diagonal and below) and U is upper triangular (has elements only the diagonal and above).

Replacing x on the left-hand side by $x^{(n+1)}$ and x on the right-hand side by $x^{(n)}$ leads to the iteration formula of the Jacobi method

$$x^{(n+1)} = -D^{-1}(L+U)x^{(n)} + D^{-1}b. \tag{10}$$

The element-based formula is thus:

$$x_i^{(n+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}x_j^{(n)}\right), \ (i = 1,2,...,N). \tag{11}$$

Note that the computation of $x_i^{(n+1)}$ requires each but the *i*-th component of $x^{(n)}$. Unlike the Gauss-Seidel method, we cannot overwrite $x_i^{(n)}$ with $x_i^{(n+1)}$, as that value will be needed for the rest of the computation. This is the most meaningful difference between the Jacobi and the Gauss-Seidel methods.

The Jacobi method converges if the matrix A is *diagonally dominant*: if in every row of the matrix, the magnitude of the diagonal entry in that row is larger than or equal to the sum of the magnitudes of all the other (non-diagonal) entries in that row, and if in at least one row of the matrix, the magnitude of the diagonal entry in that row is strictly larger than the sum of the magnitudes of all the other (non-diagonal) entries in that row. More precisely, the matrix A is diagonally dominant if

$$|a_{ii}| \geq \sum_{i\neq j} |a_{ij}| \text{ for all } i, \ |a_{ii}| > \sum_{i\neq j} |a_{ij}| \text{ for at least one } i. \tag{12}$$

If the strict inequality is true for all rows (all values of i), then the matrix is called *strictly diagonally dominant*.

The second condition of convergence is the *Spectral Radius* of the iteration matrix:

$$\rho\left(D^{-1}(L+U)\right)<1. \tag{13}$$

3.2 The Gauss-Seidel Method

The Gauss-Seidel method, also known as the Liebmann method or the method of successive displacement, is an iterative method used to solve a linear system of equations. It is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel, and is similar to the Jacobi method.

Analogously to the Jacobi method, we can rewrite system $Ax=b$ as

$$(L+D)x+Ux=b, \tag{14}$$

which further implies

$$x=\left(L+D\right)^{-1}\left[-Ux+b\right]. \tag{15}$$

This leads to the iterative formulation of the Gauss-Seidel method:

$$x^{(n+1)}=-\left(L+D\right)^{-1}Ux^{(n)}+\left(L+D\right)^{-1}b. \tag{16}$$

However, by taking advantage of the triangular form of $\left(L+D\right)$, the elements of $x^{(n+1)}$ can be computed sequentially using forward substitution:

$$x_i^{(n+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{j<i}a_{ij}x_j^{(n+1)}-\sum_{j>i}a_{ij}x_j^{(n)}\right),\quad \left(i=1,2,...,N\right). \tag{17}$$

The computation of $x_i^{(n+1)}$ uses only the elements of $x^{(n+1)}$ that have already been computed, and only the elements of $x^{(n)}$ that have yet to be advanced to iterations $n+1$.

The convergence properties of the Gauss-Seidel method are dependent on the matrix *A*. Namely, the procedure is known to converge if either:

- A is a symmetric *positive-definite*, or
- *A* is strictly or diagonally dominant.

The Gauss-Seidel method sometimes converges even if these conditions are not satisfied.

In this research, solving the nonlinear Equations(1) - (3) requires methods for nonlinear systems. Here we briefly introduce some methods for solving nonlinear systems of equations which are used in the thesis.

3.3 The Jacobi and Gauss-Seidel Method for Nonlinear Systems

Consider a nonlinear system of equations written in the form

$$\begin{aligned} x_1 &= G_1(x_1, x_2, ..., x_N), \\ x_2 &= G_2(x_1, x_2, ..., x_N), \\ &\vdots \qquad\qquad \vdots \\ x_N &= G_N(x_1, x_2, ..., x_N). \end{aligned} \tag{18}$$

This system can be solved similarly to the Jacobi method, where the next iteration requires all value of the last iteration for calculation as equations below:

$$\begin{aligned} x_1^{(n+1)} &= G_1(x_1^{(n)}, x_2^{(n)}, ..., x_N^{(n)}), \\ x_2^{(n+1)} &= G_2(x_1^{(n)}, x_2^{(n)}, ..., x_N^{(n)}), \\ &\vdots \qquad\qquad \vdots \\ x_N^{(n+1)} &= G_N(x_1^{(n)}, x_2^{(n)}, ..., x_N^{(n)}). \end{aligned} \tag{19}$$

For the Gauss-Seidel method, the next iteration requires not all values of the previous iteration but update the previous iteration to calculation as Equations below:

$$\begin{aligned} x_1^{(n+1)} &= G_1(x_1^{(n)}, x_2^{(n)}, ..., x_N^{(n)}) \\ x_2^{(n+1)} &= G_2(x_1^{(n+1)}, x_2^{(n)}, ..., x_N^{(n)}) \\ &\vdots \qquad\qquad \vdots \\ x_N^{(n+1)} &= G_N(x_1^{(n+1)}, x_2^{(n+1)}, ...x_{N-1}^{(n+1)}, x_N^{(n)}) \end{aligned} \tag{20}$$

3.4 The Newton Method

The Newton method is a method for solving a nonlinear system of Equations

$$F(x) = 0 \tag{21}$$

where $F : D \to R^k$ is a continuously differentiable function defined on some open subset $D \subset R^k$.

We begin by considering a function of one variable. Let x_0 be an approximation to a zero of the function f. In a neighborhood of x_0, by Taylor's formula we have that

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) =: g(x). \tag{22}$$

Therefore, we may consider the zero of the affine linear function g as a new approximation to the zero of f and denote it by x_1. From the linear equation

$$f(x_0) + f'(x_0)(x_1 - x_0) = 0, \tag{23}$$

we immediately obtain

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}. \tag{24}$$

Geometrically, the affine function *g* describes the tangent line to the graph of the function *f* at the point x_0.

This consideration can be extended to the case of more than one variable. Given an approximation x_0 to a zero of F, by Taylor's formula we still have the approximation (22), where now

$$F'(x) = \left(\frac{\partial f_j}{\partial x_k}(x) \right)_{j,k=1,\ldots,n}, \tag{25}$$

denote the Jacobian matrix of F.

Again we obtain a new approximation x_1 for the solution of *f(x) = 0* by solving the linearized equation similar to (23), i.e., by

$$x_1 = x_0 - [f'(x_0)]^{-1} f(x_0). \tag{26}$$

Rather than computing the inverse of this matrix (Equation(26)), one can save time by solving the system of linear equations

$$F'(x_n)(x_{n+1} - x_n) = -F(x_n), \tag{27}$$

for the unknown $x_{n+1} - x_n$.

For solving a system of linear equations we use LU decomposition.

3.5 LU Decomposition for Solving Linear System of Equations

Suppose we are able to write the matrix A as a product of two matrices,

$$L \cdot U = A, \tag{28}$$

where L is *lower triangular* (has elements only the diagonal and below) and U is *upper triangular* (has elements only the diagonal and above). For the case of a 4×4 matrix A, for example, Equations(28) are

$$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \cdot \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ 0 & \beta_{22} & \beta_{23} & \beta_{24} \\ 0 & 0 & \beta_{33} & \beta_{34} \\ 0 & 0 & 0 & \beta_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{29}$$

The decomposition (28) can be used for solving the set of linear equations

$$A\cdot x = (L\cdot U)\cdot x = L(U\cdot x) = b\,. \tag{30}$$

First solving for the vector y:

$$L\cdot y = b\ . \tag{31}$$

And then solving the linear system of equations

$$U\cdot x = y\ . \tag{32}$$

What is the advantage of breaking up a system of linear equations into two successive ones?

The advantage is that the solution of a triangular set of equations is quite trivial: Equations(31) can be solved by *forward substitution* as follows,

$$y_1 = \frac{b_1}{\alpha_{11}}\ ,$$

$$y_i = \frac{1}{\alpha_{ii}}\left[b_i - \sum_{j=1}^{i-1} \alpha_{ij} y_j \right], \qquad (i = 2,3,\ldots,N). \tag{33}$$

While (32) can then be solved by *back substitution* as below,

$$x_N = \frac{y_N}{\beta_{NN}} \ ,$$

$$x_i = \frac{1}{\beta_{ii}}\left[y_i - \sum_{j=i+1}^{N} \beta_{ij} x_j \right], \qquad (i = N-1, N-2, \ldots, 1). \tag{34}$$

How then can we solve L and U with a given A? First, we write out the i, j th component of Equation(28) or (29). That component always is a sum beginning with

$$\alpha_{i1}\beta_{1j} + \cdots = a_{ij} \tag{35}$$

The number of terms in the sum depends, however, on whether i or j is the smaller number. We have, in fact, the three cases,

$$i < j: \qquad \alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} + \cdots + \alpha_{ii}\beta_{ij} = a_{ij} \ , \tag{36}$$

$$i = j: \qquad \alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} + \cdots + \alpha_{ii}\beta_{jj} = a_{ij} , \tag{37}$$

$$i > j: \qquad \alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} + \cdots + \alpha_{ij}\beta_{jj} = a_{ij} \ . \tag{38}$$

Equations(36) - (38) are N^2 equations for the $N^2 + N$ unknown $\alpha's$ and $\beta's$ (the diagonal being represented twice). Since the number of unknowns is greater than the number of equations, we can arbitrarily specify N of the unknowns and then try to solve for the others. In fact, as we shall see, it is always possible to take

$$\alpha_{ii} = 1, \qquad (i = 1, \ldots, N). \tag{39}$$

A surprising procedure, now, is *Crout's algorithm*, which quite trivially solves the set of $N^2 + N$ Equations(36) - (39) for all the $\alpha's$ and $\beta's$ by just arranging the equations in a certain order! That order is as follows:

- Set $\alpha_{ii} = 1, \qquad (i = 1, \ldots, N)$ (Equation(39))
- For each $j = 1, 2, 3, \ldots, N$ do these two procedures: First, for

  $i = 1, 2, 3, \ldots, j$ use (36), (37), and (39) to solve for β_{ij}, namely

$$\beta_{ij} = a_{ij} - \sum_{k=1}^{i-1} \alpha_{ik}\beta_{kj} . \tag{40}$$

  when $i = 1$ in (40) the summation term is taken to mean zero. Second, for $i = j+1, j+2, \ldots, N$ use (38) to solve for α_{ij}, namely

$$\alpha_{ij} = \frac{1}{\beta_{ij}} \left(a_{ij} - \sum_{k=1}^{i-1} \alpha_{ik}\beta_{kj} \right) \tag{41}$$

Both procedures must be done before going on to the next j.

CHAPTER IV

COMPARISON OF NUMERICAL METHODS

4.1 Testing the LU Decomposition Method

We verified that LU decompostion is suitable for solving by using the following test :

1) Matrix A and vector X_e are created as input data,

2) Compute vector B by multiplying the matrix A and the vector X_e:

$$B = A \cdot X_e ,$$

3) After getting B use the LU decomposition algorithm to find X_a, solving the linear system of equations

$$A \cdot X_a = B ,$$

4) After using LU decomposition obtain the vector X_a,

5) Finally compare X_e and X_a in the norms below

5.1) $\|A \cdot X_a - B\|$ (42)

5.2) $\|X_e - X_a\|$ (43)

5.3) $\dfrac{\|X_e - X_a\|}{\|X_e\|}$. (44)

The programming Code in C++ and the result of the LU decomposition are listed in Appendix A.

4.2 Application of the Newton Method for Solving the Charge Compensation Model

For applying Newton's method for solving Equations(1) - (3) we need to transform a two-dimensional array into one-dimensional array. For understanding this placement we demonstrate it by using Figure 4.1. Equations(1) - (3) are rewritten as

$$F_m = q_{i,j} + R \cdot \sum_{k.l=NN's}^{6} q_{k.l} - a\left(q_{i,j}\right)^3 = 0, \tag{45}$$

where the relations between m, i and j are defined as it shown in Figure 4.1 (see examples below)

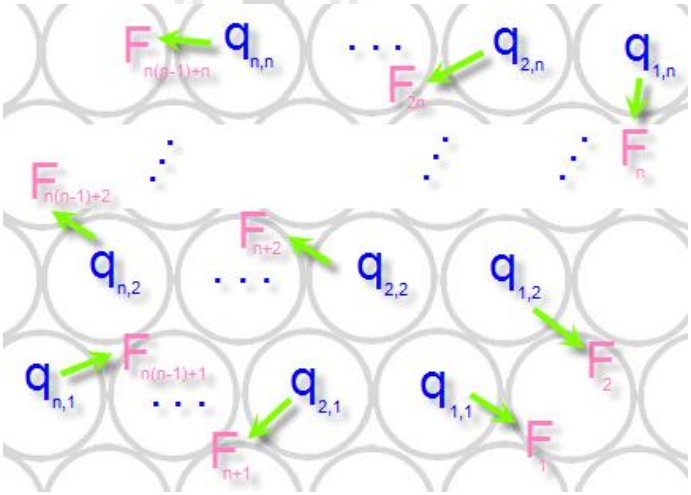


Figure 4.1 Demonstration of the transformation of a two-dimensional array into a one-dimensional array.

We have $n \times n = n^2$ unknowns $q_{i.j}$. This means that the number of equations is n^2.

The values of $q_{i.j}$ for i, j outside of the boundary of the $n \times n$ elements in the lattice are assumed to be zero.

In Equation (45), the term of $\sum_{k.l=NN's}^{6} q_{k.l}$ is equal to the summation of 6 atoms around $q_{i.j}$:

$$\sum_{k.l=NN's}^{6} q_{k.l} = q_{i-1,j-1} + q_{i-1,j} + q_{i,j-1} + q_{i,j+1} + q_{i+1,j} + q_{i+1,j+1}. \quad (46)$$

Substituting (46) into Equation (45), we have

$$F_m = q_{i,j} + R\left(q_{i-1,j-1} + q_{i-1,j} + q_{i,j-1} + q_{i,j+1} + q_{i+1,j} + q_{i+1,j+1}\right) - a\left(q_{i,j}\right)^3 = 0. \quad (47)$$

Here we explain the transformation using $n = 500$.

The first function F_1 is

$$F_1 = q_{1,1} + R\left(q_{1,2} + q_{2,1} + q_{2,2}\right) - a\left(q_{1,1}\right)^3 = 0. \quad (48)$$

Continue along the first row of atoms until

$$F_n = F_{500} = q_{1,500} + R\left(q_{1,499} + q_{2,500}\right) - a\left(q_{1,500}\right)^3 = 0. \quad (49)$$

The next row of atoms begins with

$$F_{n+1} = F_{501} = q_{2,1} + R\left(q_{1,1} + q_{2,2} + q_{3,1} + q_{3,2}\right) - a\left(q_{2,1}\right)^3 = 0. \quad (50)$$

Finally

$$F_{n^2} = F_{250000} = q_{500,500} + R\left(q_{499,499} + q_{499,500} + q_{500,499}\right) - a\left(q_{500,500}\right)^3 = 0. \quad (51)$$

In this case the Jacobi matrix is obtained as follows. For example, the first row of the matrix F' is

$$\frac{\partial F_1}{\partial q_{1,1}} = 1 - 3a\left(q_{1,1}\right)^2, \ \frac{\partial F_1}{\partial q_{1,2}} = R, \ \frac{\partial F_1}{\partial q_{2,1}} = R, \ \frac{\partial F_1}{\partial q_{2,2}} = R.$$

Thus the matrix

$$F'\left(q^{(n)}\right)=A=\left[a_{i,j}\right]_{n^2\times n^2}=\frac{\partial\left(F_1,F_2,\ldots,F_{n^2}\right)}{\partial\left(q_{1,1},q_{1,2},\ldots,q_{n,n}\right)} \tag{52}$$

has the form

$$F'\left(q^{(n)}\right)=\begin{bmatrix} \frac{\partial F_1}{\partial q_{[1][1]}} & \frac{\partial F_1}{\partial q_{[1][2]}} & \frac{\partial F_1}{\partial q_{[1][3]}} & \cdots & \frac{\partial F_1}{\partial q_{[2][1]}} & \frac{\partial F_1}{\partial q_{[2][2]}} & \frac{\partial F_1}{\partial q_{[2][3]}} & \cdots & \frac{\partial F_1}{\partial q_{[3][1]}} & \cdots \\ \frac{\partial F_2}{\partial q_{[1][1]}} & \frac{\partial F_2}{\partial q_{[1][2]}} & \frac{\partial F_2}{\partial q_{[1][3]}} & \cdots & \frac{\partial F_2}{\partial q_{[2][1]}} & \frac{\partial F_2}{\partial q_{[2][2]}} & \frac{\partial F_2}{\partial q_{[2][3]}} & \cdots & \frac{\partial F_2}{\partial q_{[3][1]}} & \cdots \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \cdots \\ \frac{\partial F_n}{\partial q_{[1][1]}} & \frac{\partial F_{500}}{\partial q_{[1][2]}} & \frac{\partial F_{500}}{\partial q_{[1][3]}} & \cdots & \frac{\partial F_{500}}{\partial q_{[2][1]}} & \frac{\partial F_{500}}{\partial q_{[2][2]}} & \frac{\partial F_{500}}{\partial q_{[2][3]}} & \cdots & \frac{\partial}{\partial q_{[3][1]}} & \cdots \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \cdots \end{bmatrix}_{n^2\times n^2}, \tag{53}$$

or after substituting the derivatives

$$F'\left(q^{(n)}\right)=\begin{bmatrix} 1-3a\left(q_{11}\right)^2 & R & 0 & \cdots & R & R & 0 & \cdots & 0 & \cdots \\ R & 1-3a\left(q_{12}\right)^2 & R & \cdots & 0 & R & R & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \cdots \\ R & 0 & 0 & \cdots & 1-3a\left(q_{21}\right)^2 & R & 0 & \cdots & R & \cdots \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \cdots \end{bmatrix}_{n^2\times n^2}. \tag{54}$$

We see that the matrix F' has a particular structure.

Let us explain the algorithm used for programming for obtaining the matrix $A=F'\left(q^{(n)}\right)$ in the Newton method:

$$A=\begin{bmatrix} a_{1,1} & \cdots & a_{1,n^2} \\ \vdots & \ddots & \vdots \\ a_{n^2,1} & \cdots & a_{n^2,n^2} \end{bmatrix}_{n^2\times n^2}=\left[a_{ij}\right]_{n^2\times n^2}. \tag{55}$$

The entries $a_{i,j}$ of the matrix A are defined by the following :

- If $i=j$ then $a_{i,j}=1-3a\left(q_{k.l}\right)^2$

- If $|i-j|=1$ then $a_{i,j}=R$
- If $|i-j|=500$ then $a_{i,j}=R$
- If $|i-j|=501$ then $a_{i,j}=R$
- The remaining entries $a_{i,j}=0$

The vector B is

$$B=\begin{bmatrix} b_1 \\ \vdots \\ b_{n^2} \end{bmatrix}_{1\times n^2} = -F\left(q^{(n)}\right). \tag{56}$$

The C++ code of application of Newton's method for solving Equations(1) - (3) is in Appendix B.

4.3 The Jacobi and Gauss-Seidel Methods for Solving the Charge Compensation Model Equations

In this section iterative methods similar to the Jacobi and Gauss-Seidel method for solving Equation(1) - (3) are presented.

4.3.1 The Jacobi Method

For nonlinear the system of Equations(1) - (3) the Jacobi Method is used in the form

$$q_{i,j}=-R\cdot\sum_{k.l=NN's}^{6} q_{k.l}+a\left(q_{i,j}\right)^3 \tag{57}$$

The iterative method is defined by the formulae

$$q_{i,j}^{(n+1)} = -R \cdot \sum_{k.l=NN's}^{6} q_{k.l}^{(n)} + a\left(q_{i,j}^{(n)}\right)^3. \tag{58}$$

The corresponding code in C++ is presented in Appendix C.

After executing the program for the Jacobi Method, it is obtained that this method is divergent for the parameters of the problem studied.

4.3.2 The Gauss-Seidel Method

The method is defined by the formulae

$$q_{i,j}^{(n+1)} = -R \cdot \sum_{k.l=NN's}^{6} q_{k.l}^{(m(n))} + a\left(q_{i,j}^{(m(n))}\right)^3 \tag{59}$$

where $m(n) = n$ or $m(n) = n+1$.

The function $m(n)$ depends on the way of computing.

At each iteration we compute

$$S(n) = \sum \left(q_{i,j}^{(n+1)} - q_{i,j}^{(n)}\right)^2. \tag{60}$$

The computations are stopped if $S(n) < \varepsilon$. The corresponding code in C++ is presented in Appendix D.

4.4 Comparison of the Newton Method and Gauss-Seidel Method

Newton's method and the LU decomposition method have the limitation that they require a substantial amount of memory of a computer. The maximum number of atoms that computer available could process was 95x95 atoms. In this section we present the results of comparisons of the Newton and Gauss-Seidel methods for n = 95, 75 and 30.

Case 95x95 atoms

The calculations were carried out with the following set of parameters:

a=0.01, d=1.0, R=0.2, M=95, esp=0.001, Nd=800.

The results of calculations are presented in Figure 4.2.

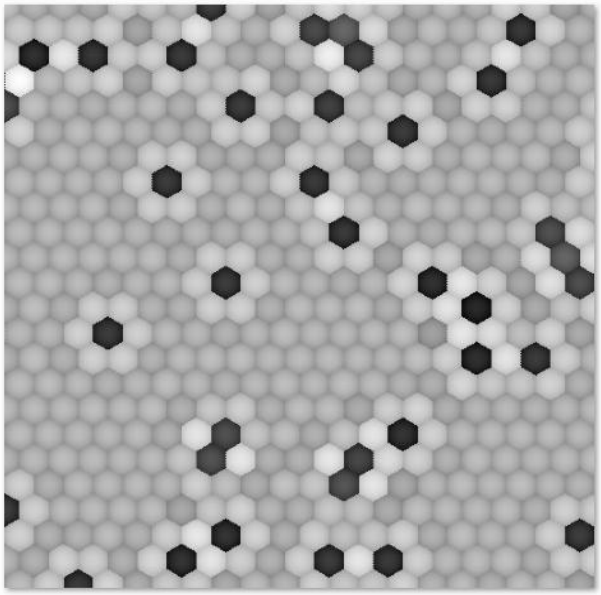


Figure 4.2 Charge of atom.

Here the color depends on the charge of atoms, where the black color corresponds to the defected atoms.

The difference in the norms between the methods is given in Table1.

Table 4.1 Comparison for n = 95.

$\sqrt{\sum\left(q_{i,j}^{n}-q_{i,j}^{g}\right)^{2}}$	$\sqrt{\sum\left(q_{i,j}^{n}\right)^{2}}$	$\dfrac{\sqrt{\sum\left(q_{i,j}^{n}-q_{i,j}^{g}\right)^{2}}}{\sqrt{\sum\left(q_{i,j}^{n}\right)^{2}}}$
0.014869	37.360644	0.000398

Case 75x75 atoms

The calcutions were carried out with the following set of parameters

a=0.01, d=1.0, R=0.2, M=75, esp=0.001, Nd=550.

The results of the comparison of the methods in case 75x75 are given in Table2.

Table 4.2 Comparison for n = 75.

$\sqrt{\sum\left(q_{i,j}^{n}-q_{i,j}^{g}\right)^{2}}$	$\sqrt{\sum\left(q_{i,j}^{n}\right)^{2}}$	$\frac{\sqrt{\sum\left(q_{i,j}^{n}-q_{i,j}^{g}\right)^{2}}}{\sqrt{\sum\left(q_{i,j}^{n}\right)^{2}}}$
0.022319	30.744219	0.000726

Case 30x30 atoms

The calcutions were carried out with the following set of parameters

a=0.01, d=1.0, R=0.2, M=30, esp=0.001, Nd=100.

The results of the comparison of the methods in case 30x30 are given in Table3.

Table4. 3 Comparison for n = 30.

$\sqrt{\sum\left(q_{i,j}^{n}-q_{i,j}^{g}\right)^{2}}$	$\sqrt{\sum\left(q_{i,j}^{n}\right)^{2}}$	$\frac{\sqrt{\sum\left(q_{i,j}^{n}-q_{i,j}^{g}\right)^{2}}}{\sqrt{\sum\left(q_{i,j}^{n}\right)^{2}}}$
0.018343	13.315818	0.001378

Comparison

The convergence of the Gauss-Seidel and Newton's methods was compared, by comparing the values of S(n) at each iteration step, as well as computation time, for various values of R(T).

Comparison in terms of the value of convergence (S(n))

Case 95x95 atoms

Table 4.4 Comparison of S(n) between the methods for n = 95.

n	$S(n)$ *of Gauss-Seidel's Method*	$S(n)$ *of Newton's Method*
1	827.133491	1346.877778
2	91.375741	3.366801
3	14.179667	0.429694
4	2.555106	0.071258
5	0.501284	0.012342
6	0.104474	0.002180
7	0.022813	0.000390
8	0.005152	
9	0.001195	
10	0.000282	

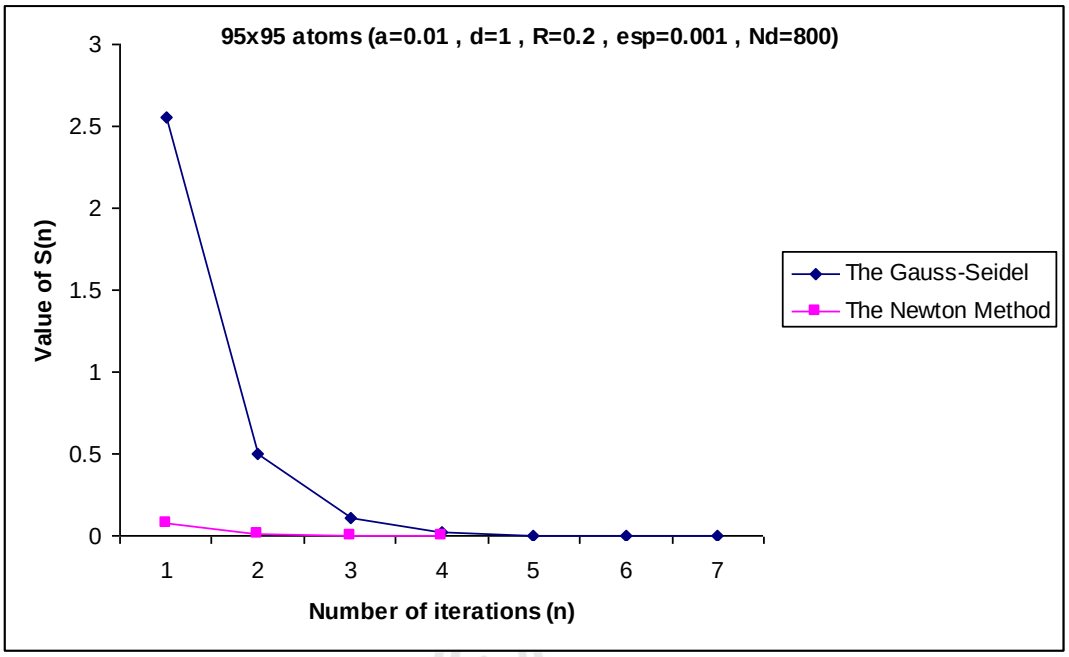


Figure 4.3 Comparison of the values of S(n) between the methods for n = 95.

Case 75x75 atoms

Table 4.5 Comparison of S(n) between the methods for n = 75.

n	*S*(*n*) *of Gauss-Seidel's Method*	*S*(*n*) *of Newton's Method*
1	568.873882	914.104958
2	61.474105	3.537602
3	9.327769	0.423859
4	1.646859	0.066091
5	0.319036	0.011084
6	0.065896	0.001936
7	0.014289	0.000347
8	0.003209	
9	0.000740	

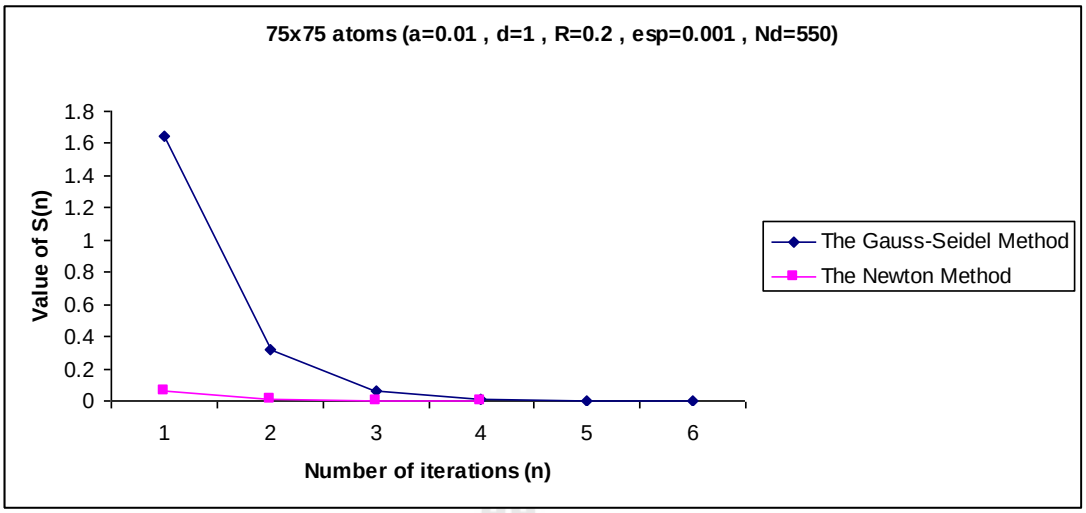


Figure 4.4 Comparison of the values of S(n) between the methods for n = 75.

Case 30x30 atoms

Table 4.6 Comparison of S(n) between the methods for n = 30.

n	$S(n)$ *of Gauss-Seidel's Method*	$S(n)$ *of Newton's Method*
1	104.774614	172.159369
2	11.394245	1.130951
3	1.715829	0.139784
4	0.293744	0.020731
5	0.053735	0.003306
6	0.010592	0.000551
7	0.002217	
8	0.000481	

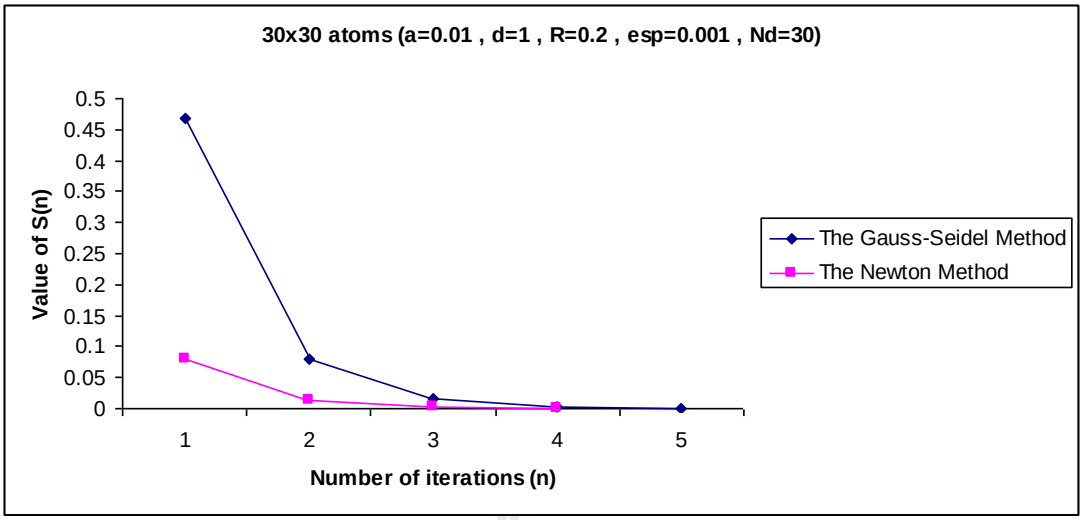


Figure 4.5 Comparison of the values of S(n) between the methods for n = 30.

Comparison in terms of computing time

Table 4.7 Comparison of compution times.

Problem size (nxn) atoms	30x30 atoms	35x35 atoms	40x40 atoms	45x45 atoms	50x50 atoms	55x55 atoms
Calculation time of the Gauss-Seidel Method	3 s	3 s	3 s	3 s	3 s	3 s
Calculation time of the Newton method	22 s	80 s	87 s	190 s	370 s	500 s

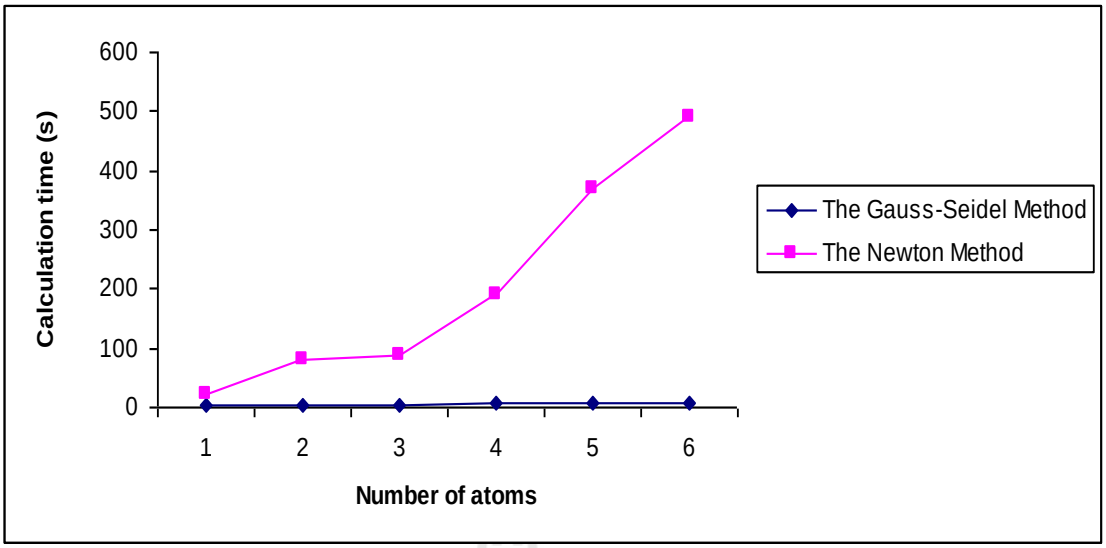


Figure 4.6 Comparison of computation times between the methods.

Negative value of R

We use the case of 30x30 atoms for comparing both methods for negative values of R (dewetting problem).

Case R = -0.140

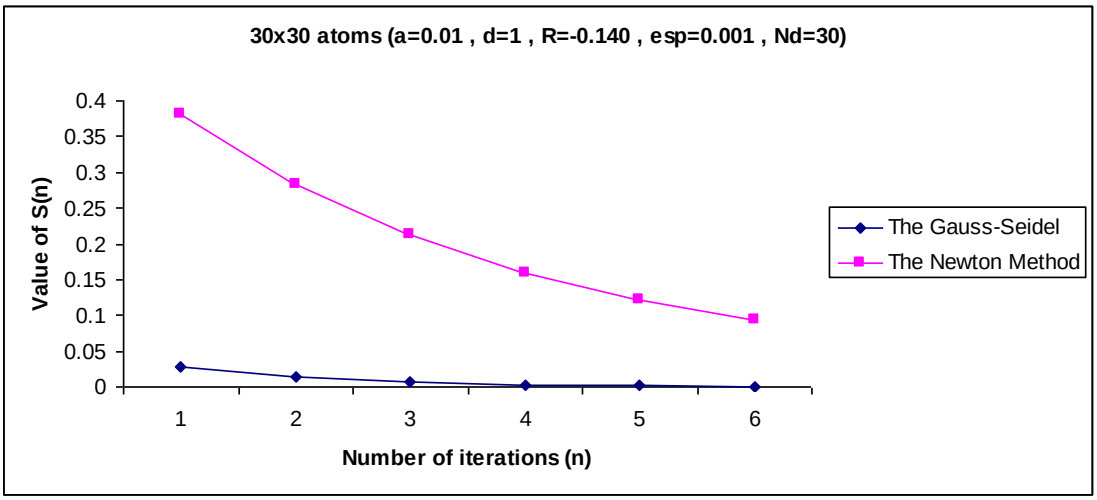


Figure 4.7 Comparison of the values of S(n) between the methods for R= -0.140.

Case R = -0.142

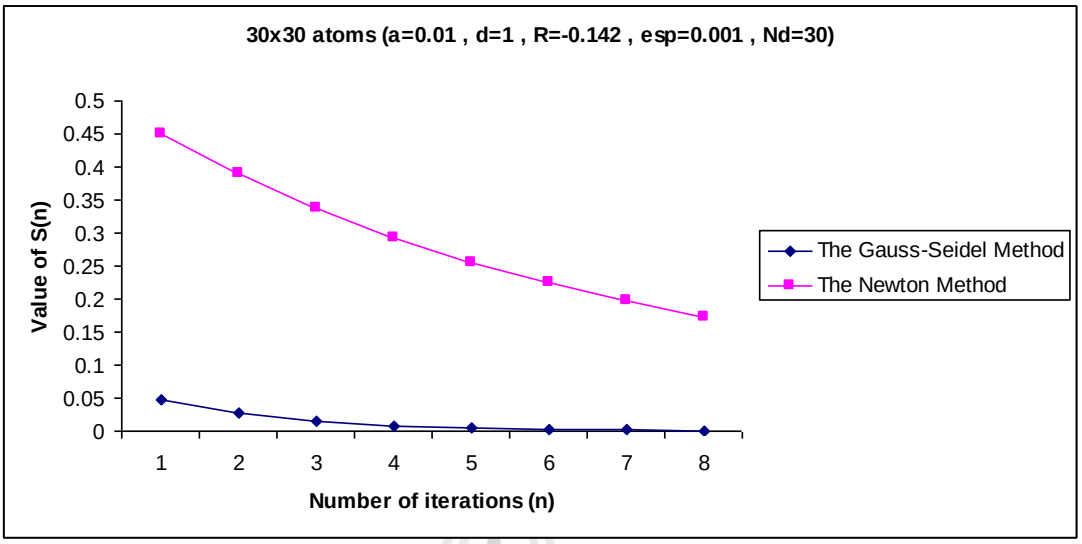


Figure 4.8 Comparison of the values of S(n) between the methods for R= -0.142.

Case R = -0.10

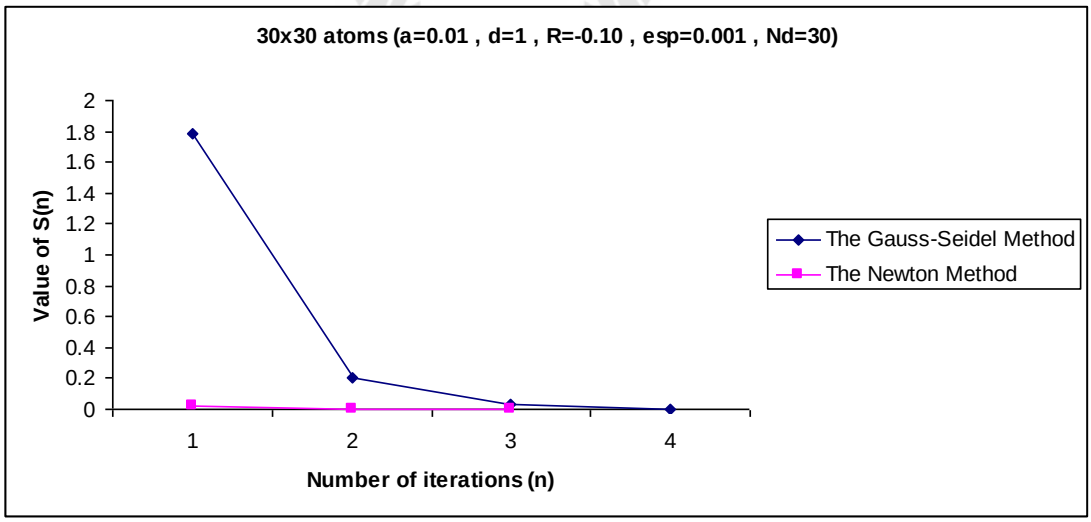


Figure 4.9 Comparison of the values of S(n) between the methods for R= -0.10.

Case R = -0.05

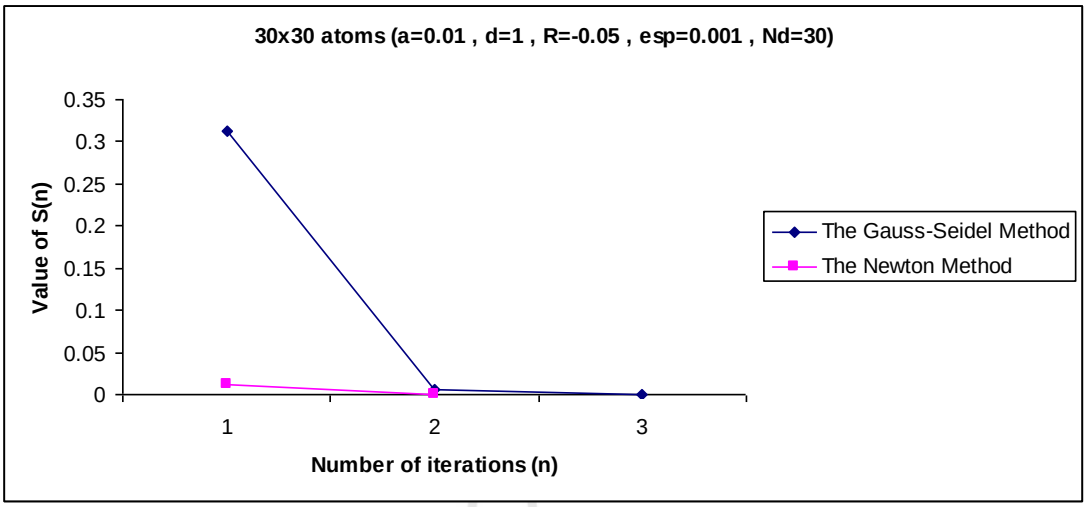


Figure 4.10 Comparison of the values of S(n) between the methods for R= -0.05.

CHAPTER V

MOTION OF ATOMS

5.1 Algorithm for Modeling the motion of the Atoms

Consider a substitutional defect atom $q_{i,j}$ which is surrounded by six neighbor atoms $q_1, q_2, \ldots, q_6$ (Figure 4.16)

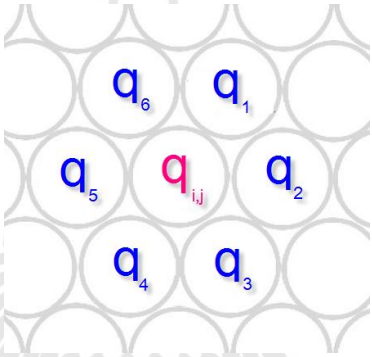


Figure 4.11 Atom positions.

The following criteria were used for the motion of atoms:

1) Randomly choose the direction of the motion into one of these six direction 1 to 6; for example, the direction k.

2) Check that the atom q_k is not substituted by a defective atom, otherwise repeat choosing the direction of the motion.

3) Randomly choose the value in the interval [0.001, 1], for example, it is R_d.

4) Calculate

$$\beta = e^{-\gamma\left(q_{i,j}-\tilde{q}\right)^2}. \tag{61}$$

5) If $\beta \geq R_d$, the substitutional defect atom moves in the k-th direction. If $\beta < R_d$, then the substitutional defect atom $q_{i,j}$ does not move.

6) Consider the next defective atom.

7) After considering all defective atoms, solution of Equations(1) - (3) is computed.

5.2 Results

The results of the calculations using the Gauss-Seidel method with movements of the defective atoms are presented in animated movie.

In the animation the picture is shown for each 10 steps of the motion of atoms.

CHAPTER VI

CONCLUSION

In this thesis the C++ code was developed for solving the Charge Compensation Model.

The used algorithm consists of two steps. In the first step, an iterative method for solving nonlinear Equations(1) - (3) was applied. Three iterative methods were explored: the Jacobi method, the Gauss-Seidel method and the Newton method. It was obtained that the Gauss-Seidel method is preferable.

In the second step, the movement of defected atoms was considered. The model of the motion of atoms was developed. The animated results are presented as a wmv-file.

REFERENCES

REFERENCES

Melechko, A.V., Simkin, M.V., Samatova, N.F., Braun, J.,and Plummer,E.W. (2001) Complex structural phase transition in a defect-populate two-dimensional System. **Physical Review.** 64: 235-424.

Barrett, R. Berry, M. Chan, T. F. Demmel, Donato, Dongarra, J. Eijkhout,V. Pozo R Romine, C. and van der Vorst, H. (1994). **Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods**. Philadelphia, PA: 95-104.

Carpinelli, J. M., Weitering, H. H., Plummer E. W. and Stumpf R.(1996). Direct observation of a surface charge density wave**. Nature 381(6581)**: 398-400.

Plummer, E.W., Ismil, R. Matzdorf, Melechko, A.V. and Jiandi Zhang. (2001). The next 25 years of surface physics. **Progress in Surface Science. 67**: 17-44.

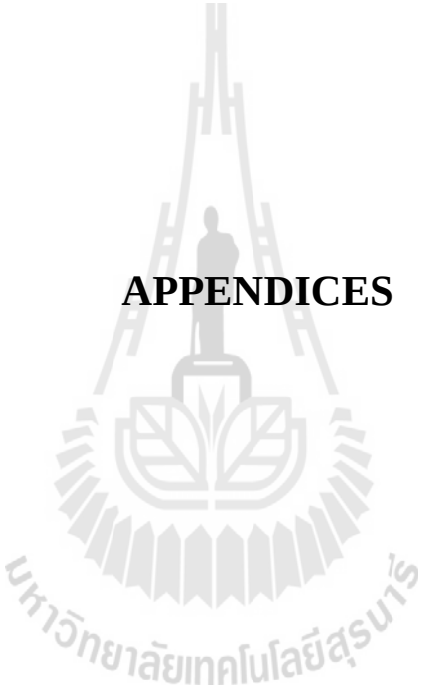
Ottaviano, L., Melechko, A.V. , Santucci, S. and Plummer E.W.(2000). Direct visualization of defect Density Waves in 2D. **Physical Review Letters. 86(9)**

Nikolay P. Moshkin. (2009). **Lecture Notes on Numerical Linear Algebra**. Suranaree University. Nakornratchasima.

Kress Rainer. (1998). **Numerical Analysis**. New York, PA. Springer-Verlag: 101-110.

William H.Press, Saul A. Teukolsky, William T.Vetterling, Brian P. Flannery.(1988) Numerical Recipes in C. The Art of Scientific Computing. Second Edition. Cambridge University Press: 32-49.

APPENDICES



APPENDIX A

LU DECOMPOSITION PROGRAM CODE IN C++

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>

//*****************************
#define  M 920

int i,j,k,nq;

double sum,temH;

double Y[M];
double X[M];
double Xe[M];
double B[M];

double AA[M][M];
double AL[M][M];
double AU[M][M];

void main()
{
        FILE* fa = fopen("dataA2.txt", "r");
        FILE* fxe = fopen("dataXe2.txt", "r");
        FILE* fw = fopen("output2.txt", "w");


//*********************************************************************

// /For read Matrix:A and Xe by files

      nq=5;

                for (i=1;i<=nq;i++)
                {
                        for (j=1;j<=nq;j++)
                        {
```

```
                fscanf(fa,"%lf",&AA[i][j]);
                        }

                        fscanf(fxe,"%lf",&Xe[i]);
                }
//*********************************************************************

// For write Output file!!

fprintf(fw,"\n\n*************************************************\n");
fprintf(fw,"                WELCOME                  ");
fprintf(fw,"\n*************************************************\n");


        for (i=1;i<=nq;i++)
        {
                for (j=1;j<=nq;j++)
                {

                        fprintf(fw,"A[%d][%d]= %lf  ",i,j,AA[i][j]);

                }
                        fprintf(fw,"\n");
        }
fprintf(fw,"\n\n*************************************************\n");

//
        for (i=1;i<=nq;i++)
        {
                        fprintf(fw,"Xe[%d]= %lf    ",i,Xe[i]);
                        fprintf(fw,"\n");
        }
fprintf(fw,"\n\n*************************************************\n");



//*********************************************************************

// For build Matrix:B

      for (i=1;i<=nq;i++)
      {
                        sum = 0;

                        for (j=1; j<=nq; j++)
                        {
           sum = sum + AA[i][j]*Xe[j];
```

```
                        }

                        B[i] = sum;
        }

//*************************************************************************

// For write Output file!!

//
        for (i=1;i<=nq;i++)
        {
                        fprintf(fw,"B[%d]= %lf    ",i,B[i]);
                        fprintf(fw,"\n");
        }

//*************************************************************************

/*....PERFORM DECOMPOSITION [AA] = [AL]*[AU]          */
        for (i=1; i<=nq; i++)
   {
                for (j=1; j<=nq; j++)
      {
                        AL[i][j] = 0;
                        AU[i][j] = 0;
                }
                AU[i][i] = 1;
        }

   for (i=1; i<=nq; i++)
   {
      AL[i][1] = AA[i][1];
   }

   for (j=2; j<=nq; j++)
   {
      AU[1][j] = AA[1][j]/AL[1][1];
   }

   for (j=2; j<=nq-1; j++)
   {
      for (i=j; i<=nq; i++)
      {
         sum = 0;
         for (k=1; k<=j-1; k++)
            sum = sum+AL[i][k]*AU[k][j];
         AL[i][j] = AA[i][j]-sum;
```

```
      }
      for (k=j+1; k<=nq; k++)
      {
         sum = 0;
         for (i=1; i<=j-1; i++)
            sum = sum+AL[j][i]*AU[i][k];
         AU[j][k] = (AA[j][k]-sum)/AL[j][j];
      }
    }

   sum = 0;
   for (k=1; k<=nq-1; k++)
      sum = sum+AL[nq][k]*AU[k][nq];
   AL[nq][nq] = AA[nq][nq]-sum;
//------------------------------------------------------------
/*
// For Monitor!!
         for (i=1;i<=nq;i++)
         {
                  for (j=1;j<=nq;j++)
                  {
                           printf("AL[%d][%d]= %lf    ",i,j,AL[i][j]);

                  }
                           printf("\n");
         }
printf("\n\n*************************************************\n");


//
         for (i=1;i<=nq;i++)
         {
                  for (j=1;j<=nq;j++)
                  {

                           printf("AU[%d][%d]= %lf    ",i,j,AU[i][j]);

                  }
                           printf("\n");
         }

printf("\n\n*************************************************\n");


// For write Output file!!
         for (i=1;i<=nq;i++)
         {
```

```
                for (j=1;j<=nq;j++)
                {

                        fprintf(fw,"AL[%d][%d]= %lf    ",i,j,AL[i][j]);

                }
                        fprintf(fw,"\n");
        }
fprintf(fw,"\n\n*************************************************\n");

//
        for (i=1;i<=nq;i++)
        {
                for (j=1;j<=nq;j++)
                {

                        fprintf(fw,"AU[%d][%d]= %lf    ",i,j,AU[i][j]);

                }
                        fprintf(fw,"\n");
        }

fprintf(fw,"\n\n*************************************************\n");

*/

/*....PERFORM FORWARD PASS TO SOLVE [L][Y] = {B}   */

   Y[1] = B[1]/AL[1][1];
        for (i=2; i<=nq; i++)
   {
           sum = 0.0;
                for (j=1; j<=i-1; j++)
                  {
                        sum = sum+AL[i][j]*Y[j];
        }
                Y[i] = (B[i]-sum)/AL[i][i];
        }
/*....PERFORM BACKWARD PASS TO SOLVE [U][X] = {Y}   */

        X[nq] = Y[nq];
        for (i=nq-1; i>= 1; i--)
   {
           sum = 0.0;
                for (j=i+1; j<=nq; j++)
                        sum = sum+AU[i][j]*X[j];
                X[i] = Y[i]-sum;
```

```
        }

// For Monitor!!
/*
printf("\n\n***********************************************\n");
          for (i=1; i<=nq; i++)
     {
        printf("\n X[%d] = %lf",i,X[i]);
     }
printf("\n\n***********************************************\n");
*/


// For write Output file!!
fprintf(fw,"\n\n***********************************************\n");
          for (i=1; i<=nq; i++)
     {
        fprintf(fw,"\nXa[%d] = %lf",i,X[i]);
     }
fprintf(fw,"\n\n***********************************************\n");


//************************************************************
// Check Error!

   temH = 0;

   for (i=1;i<=nq;i++)
   {
          sum = 0;

          for (j=1; j<=nq; j++)
                {
       sum = sum + AA[i][j]*X[j];
                }

                  temH = temH + (sum-B[i])*(sum-B[i]);
   }

   temH = sqrt(temH);

        fprintf(fw,"Error#1=%lf\n",temH);


//---------------------------
   temH = 0;
```

```
        for (i=1;i<=nq;i++)
        {
    temH = temH + (Xe[i]-X[i])*(Xe[i]-X[i]);
        }

        temH = sqrt(temH);

        fprintf(fw,"Error#2=%lf\n",temH);



//-----------------------------
   sum = 0;

        for (i=1;i<=nq;i++)
        {
    sum = sum + Xe[i]*Xe[i];
        }

   sum = sqrt(sum);

        temH = temH/sum;

        fprintf(fw,"Error#3=%lf\n",temH);


//--------------------------------
        printf("************************************************ \n");
   fprintf(fw,"************************************************ \n");
   printf("          FINISH :) \n");
   fprintf(fw,"FINISH :) \n");
        printf("************************************************ \n");
   fprintf(fw,"************************************************ \n");

//*************************************************************


}
```

This is input of Matrix A: filename: dataA2.text

35.81 12.56 2.98 0.45 3.84 1.73 24.31 9.88 1.25 10.32
5.32 1.59 12.68 45.45 12.11 21.23 54.39 2.18 56.85 99.02
4.77 88.4 67.54 30.55 7.77 98.12 66.89 95.75 7.75 0.32
45.6 79.56 99.11 67.95 8.12 56.99 23.95 15.36 74.18 8.81
1.51 15.4 56.93 20.86 36.34 6.13 34.33 65.98 15.25 98.37
23.84 1.32 24.21 16.45 44.84 56.73 256.35 93.68 7.25 15.62
44.55 21.65 2.19 20.24 34.14 12.33 23.36 91.84 17.28 60.39
88.81 19.56 8.98 90.45 38.86 57.78 44.38 93.87 23.95 12.84
25.2 82.58 4.94 0.40 25.82 86.78 34.33 7.87 4.29 60.36
4.6 7.56 9.11 6.95 8.12 956.99 23.95 75.36 794.18 88.81

This is input of Vector X: filename: dataXe2.txt

34.578
15.3457
24.678
6785.6
2123.467
432.67
13.5646
87.757
39.757
160.37

This is output2.txt

```
*********************************************
              WELCOME
*********************************************
A[1][1]= 35.810000  A[1][2]= 12.560000  A[1][3]= 2.980000  A[1][4]= 0.450000
A[1][5]= 3.840000  A[1][6]= 1.730000  A[1][7]= 24.310000  A[1][8]= 9.880000
A[1][9]= 1.250000  A[1][10]= 10.320000
A[2][1]= 5.320000  A[2][2]= 1.590000  A[2][3]= 12.680000  A[2][4]= 45.450000
A[2][5]= 12.110000  A[2][6]= 21.230000  A[2][7]= 54.390000  A[2][8]= 2.180000
A[2][9]= 56.850000  A[2][10]= 99.020000
A[3][1]= 4.770000  A[3][2]= 88.400000  A[3][3]= 67.540000  A[3][4]= 30.550000
A[3][5]= 7.770000  A[3][6]= 98.120000  A[3][7]= 66.890000  A[3][8]= 95.750000
A[3][9]= 7.750000  A[3][10]= 0.320000
```

A[4][1]= 45.600000 A[4][2]= 79.560000 A[4][3]= 99.110000 A[4][4]= 67.950000
A[4][5]= 8.120000 A[4][6]= 56.990000 A[4][7]= 23.950000 A[4][8]= 15.360000
A[4][9]= 74.180000 A[4][10]= 8.810000
A[5][1]= 1.510000 A[5][2]= 15.400000 A[5][3]= 56.930000 A[5][4]= 20.860000
A[5][5]= 36.340000 A[5][6]= 6.130000 A[5][7]= 34.330000 A[5][8]= 65.980000
A[5][9]= 15.250000 A[5][10]= 98.370000
A[6][1]= 23.840000 A[6][2]= 1.320000 A[6][3]= 24.210000 A[6][4]= 16.450000
A[6][5]= 44.840000 A[6][6]= 56.730000 A[6][7]= 256.350000 A[6][8]= 93.680000
A[6][9]= 7.250000 A[6][10]= 15.620000
A[7][1]= 44.550000 A[7][2]= 21.650000 A[7][3]= 2.190000 A[7][4]= 20.240000
A[7][5]= 34.140000 A[7][6]= 12.330000 A[7][7]= 23.360000 A[7][8]= 91.840000
A[7][9]= 17.280000 A[7][10]= 60.390000
A[8][1]= 88.810000 A[8][2]= 19.560000 A[8][3]= 8.980000 A[8][4]= 90.450000
A[8][5]= 38.860000 A[8][6]= 57.780000 A[8][7]= 44.380000 A[8][8]= 93.870000
A[8][9]= 23.950000 A[8][10]= 12.840000
A[9][1]= 25.200000 A[9][2]= 82.580000 A[9][3]= 4.940000 A[9][4]= 0.400000
A[9][5]= 25.820000 A[9][6]= 86.780000 A[9][7]= 34.330000 A[9][8]= 7.870000
A[9][9]= 4.290000 A[9][10]= 60.360000
A[10][1]= 4.600000 A[10][2]= 7.560000 A[10][3]= 9.110000 A[10][4]= 6.950000
A[10][5]= 8.120000 A[10][6]= 956.990000 A[10][7]= 23.950000 A[10][8]=
75.360000 A[10][9]= 794.180000 A[10][10]= 88.810000

****
Xe[1]= 34.578000
Xe[2]= 15.345700
Xe[3]= 24.678000
Xe[4]= 6785.600000
Xe[5]= 2123.467000
Xe[6]= 432.670000
Xe[7]= 13.564600
Xe[8]= 87.757000
Xe[9]= 39.757000
Xe[10]= 160.376000

****
B[1]= 16362.244148
B[2]= 362897.266957
B[3]= 279110.753964
B[4]= 514260.339122
B[5]= 245698.489928
B[6]= 247318.471944
B[7]= 235845.837141
B[8]= 736718.917260
B[9]= 108356.956564
B[10]= 531718.670722

****

Xa[1] = 34.578000
Xa[2] = 15.345700
Xa[3] = 24.678000
Xa[4] = 6785.600000
Xa[5] = 2123.467000
Xa[6] = 432.670000
Xa[7] = 13.564600
Xa[8] = 87.757000
Xa[9] = 39.757000
Xa[10] = 160.376000

****
Error#1=0.000000
Error#2=0.000000
Error#3=0.000000
****
FINISH :)
****

APPENDIX B

THE NEWTON METHOD FOR SOLVING THE CHARGE COMPENSATION MODEL PROGRAM CODE IN C++

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>

//*****************************
//#define  MM 920
#define  MM 9100

int hn,hm,k,temA,temB;

double sum,temH;

double ff[MM];
double hq[MM];
double Y[MM];
double X[MM];

double AJ[MM][MM];
double AL[MM][MM];
double AU[MM][MM];
double q[MM][MM];
double p[MM][MM];
int i,j,n,rem,M,k1,k2,i1,i2,Nd,t;

double s;
double esp;
double a;
double d;
double R;
double qi;
int alfa[MM][MM];


double Rd;
```

```
double al=0.01;
double B;

int NUM_ROW, NUM_COL;

void Write_atomZfile(char *filename, double atomZ[MM][MM]);

void Swp(double a1, double a2);


void main()
{

        srand(time(NULL));

        n=0;

        FILE* fp = fopen("data95B.txt", "r");  // by only DataR other still P!!
//      FILE* fv = fopen("alfa95B.txt", "w");
        FILE* fw = fopen("output95B2.txt", "w");


        fscanf(fp,"%lf",&a);
   fscanf(fp,"%lf",&d);
        fscanf(fp,"%lf",&R);
        fscanf(fp,"%d",&M);
        fscanf(fp,"%lf",&esp);
        fscanf(fp,"%d",&Nd);
        fscanf(fp,"%lf",&qi);

        fprintf(fw,"a=%lf \n",a);
        fprintf(fw,"d=%lf \n",d);
        fprintf(fw,"R=%lf \n",R);
        fprintf(fw,"M=%d \n",M);
        fprintf(fw,"esp=%lf \n",esp);
        fprintf(fw,"Nd=%ld \n",Nd);
        fprintf(fw,"qi=%lf \n",qi);

        printf("a= %lf,d= %lf,R=%lf,M=%d,Nd=%d\n",a,d,R,M,Nd);


        NUM_ROW = M; // Number of atoms in a row
        NUM_COL = M; // Number of atoms in a colomn

        // initializing charge

        for (i=0;i<=M+1;i++)
```

```
	{
		for (j=0;j<=M+1;j++)
		{
			p[i][j]=0;
			alfa[i][j]=0;

		}
	}

/*	for (i=1;i<=Nd;i++)
	{
tryAgain: // this is a statement label
		k1 = rand() % M + 1;
		k2 = rand() % M + 1;

	printf("defect is [%d][%d] \n",k1,k2);
		fprintf(fv,"defect is [%d][%d] \n",k1,k2);

		printf("alfa[%d][%d]=%d \n",k1,k2,alfa[k1][k2]);
		fprintf(fv,"alfa is [%d][%d]=%d \n",k1,k2,alfa[k1][k2]);

		printf("i=[%d] \n",i);
		fprintf(fv,"i=[%d]  \n",i);

		if (alfa[k1][k2]==1)
			goto tryAgain; // this is the goto statement
		else
		{
			alfa[k1][k2]=1;
			printf("defect is [%d][%d] \n",k1,k2);
			fprintf(fv,"defect is [%d][%d] \n",k1,k2);

		}
	}*/
alfa[17][95]=1;
alfa[79][68]=1;
alfa[43][88]=1;
alfa[89][9]=1;
alfa[55][94]=1;
alfa[65][63]=1;
alfa[84][29]=1;
alfa[40][93]=1;
alfa[78][4]=1;
alfa[49][19]=1;
alfa[31][81]=1;
alfa[30][40]=1;
alfa[71][74]=1;
```

```
alfa[10][75]=1;
alfa[50][66]=1;
alfa[60][34]=1;
alfa[52][33]=1;
alfa[31][26]=1;
alfa[15][24]=1;
alfa[86][25]=1;
alfa[69][52]=1;
alfa[56][72]=1;
alfa[81][13]=1;
alfa[45][38]=1;
alfa[15][37]=1;
alfa[58][16]=1;
alfa[54][43]=1;
alfa[15][65]=1;
alfa[55][31]=1;
alfa[80][6]=1;
alfa[46][86]=1;
alfa[81][92]=1;
alfa[66][52]=1;
alfa[20][66]=1;
alfa[8][70]=1;
alfa[39][86]=1;
alfa[72][74]=1;
alfa[7][94]=1;
alfa[90][80]=1;
alfa[82][86]=1;
alfa[15][39]=1;
alfa[36][69]=1;
alfa[5][72]=1;

//*****************************************************************************


        s=1;
        while (s>esp)
  {
                n++;
                for (i=0;i<=M+1;i++)
                {
                        for (j=0;j<=M+1;j++)
                        {
                                q[i][j]=p[i][j];
                        }
                }
                hn=1;
                for (i=1;i<=M;i++) // Here sill !! q = p
```

```
                {
                        for (j=1;j<=M;j++)
                        {
                                temH    =    q[i+1][j+1]+q[i][j+1]+q[i-1][j]+q[i-1][j-
1]+q[i][j-1]+q[i+1][j];

                                ff[hn]              =              q[i][j]+R*temH-
a*q[i][j]*q[i][j]*q[i][j]+alfa[i][j]*d;

                                ff[hn] = -ff[hn];

            hn++;
                        }
                }
//*****************************************************************

// COMPUTE FOR [AJ]
        hn=1;
        for (i=1; i<=M; i++)      // ! Later fixed nq by M or ?
        {
                    for (j=1; j<=M; j++)
           {
                hq[hn] = q[i][j];  // ! declare hq,hn
                hn++;
           }
        }
        hm=M*M;
        hn=1;
        for (i=1; i<=hm; i++)      // ! Later fixed nq by M or hm?
        {
                    for (j=1; j<=hm; j++)    // check relation bewteen "nq*nq = M =
lastly hn'
           {
                        AJ[i][j] = 0;
//                      ka=nq*(i-1);
//                      kb=nq*(i-1);
                        if (i == j)
             {
                AJ[i][j] = 1+3*a*hq[hn]*hq[hn];
                hn++;
             }
                        temA = fabs(i-j);
                        temB = M+1;
                  if ( (temA == 1)||(temA == M)||(temA == temB) )

             {
                AJ[i][j] = R;
```

```
            }
                    }
             }

//*****************************************************************

/*....PERFORM DECOMPOSITION [AJ] = [AL]*[AU]          */
          for (i=1; i<=hm; i++)
   {
                    for (j=1; j<=hm; j++)
      {
                              AL[i][j] = 0;
                              AU[i][j] = 0;
                    }
                    AU[i][i] = 1;
          }

   for (i=1; i<=hm; i++)
   {
      AL[i][1] = AJ[i][1];
   }

   for (j=2; j<=hm; j++)
   {
      AU[1][j] = AJ[1][j]/AL[1][1];
   }

   for (j=2; j<=hm-1; j++)
   {
      for (i=j; i<=hm; i++)
      {
         sum = 0;
         for (k=1; k<=j-1; k++)
            sum = sum+AL[i][k]*AU[k][j];
         AL[i][j] = AJ[i][j]-sum;
      }
      for (k=j+1; k<=hm; k++)
      {
         sum = 0;
         for (i=1; i<=j-1; i++)
            sum = sum+AL[j][i]*AU[i][k];
         AU[j][k] = (AJ[j][k]-sum)/AL[j][j];
      }
    }

   sum = 0;
   for (k=1; k<=hm-1; k++)
```

```
      sum = sum+AL[hm][k]*AU[k][hm];
   AL[hm][hm] = AJ[hm][hm]-sum;

/*
// For Monitor!!
        for (i=1;i<=hm;i++)
        {
                for (j=1;j<=hm;j++)
                {

                        printf("AL[%d][%d]= %lf    ",i,j,AL[i][j]);

                }
                        printf("\n");
        }
printf("\n\n*********************************************\n");


//
        for (i=1;i<=hm;i++)
        {
                for (j=1;j<=hm;j++)
                {

                        printf("AU[%d][%d]= %lf    ",i,j,AU[i][j]);

                }
                        printf("\n");
        }

printf("\n\n*********************************************\n");


// For write Output file!!
        for (i=1;i<=hm;i++)
        {
                for (j=1;j<=hm;j++)
                {

                        fprintf(fw,"AL[%d][%d]= %lf    ",i,j,AL[i][j]);

                }
                        fprintf(fw,"\n");
        }
fprintf(fw,"\n\n*********************************************\n");

//
```

```
        for (i=1;i<=hm;i++)
        {
                for (j=1;j<=hm;j++)
                {

                        fprintf(fw,"AU[%d][%d]= %lf    ",i,j,AU[i][j]);

                }
                        fprintf(fw,"\n");
        }

fprintf(fw,"\n\n***********************************************\n");

*/

/*....PERFORM FORWARD PASS TO SOLVE [L][Y] = {B}   */

   Y[1] = ff[1]/AL[1][1];
        for (i=2; i<=hm; i++)
   {
           sum = 0.0;
                for (j=1; j<=i-1; j++)
                  {
                        sum = sum+AL[i][j]*Y[j];
        }
                Y[i] = (ff[i]-sum)/AL[i][i];
        }
/*....PERFORM BACKWARD PASS TO SOLVE [U][X] = {Y}   */

        X[hm] = Y[hm];
        for (i=hm-1; i>= 1; i--)
   {
           sum = 0.0;
                for (j=i+1; j<=hm; j++)
                        sum = sum+AU[i][j]*X[j];
                X[i] = Y[i]-sum;
        }

// For Monitor!!
/*
printf("\n\n***********************************************\n");
        for (i=1; i<=hm; i++)
    {
       printf("\n X[%d] = %lf",i,X[i]);
    }
printf("\n\n***********************************************\n");
*/
```

```
// For write Output file!!
/*fprintf(fw,"\n\n*************************************************\n");
          for (i=1; i<=hm; i++)
     {
        fprintf(fw,"\n X[%d] = %lf",i,X[i]);
     }
fprintf(fw,"\n\n*************************************************\n");
*/

//***********************************************************
// For Update the chagre of atom at !n-th round

       hn=1;
       for (i=1; i<=M; i++)      // ! Later fixed nq by M or ?
       {
                    for (j=1; j<=M; j++)
          {
               p[i][j] = q[i][j]+X[hn];
               hn++;
          }
       }

//***********************************************************
// Check Error!
         s=0;
                  for (i=1;i<=M;i++)
                  {
                           for (j=1;j<=M;j++)
                           {
                                    s=s+(p[i][j]-q[i][j])*(p[i][j]-q[i][j]);
                           }
                  }

                  fprintf(fw,"s[%d]=%lf\n",n,s);

                  printf("s(%d)=%.5f\n",n,s);

                  //getch();
         }

//*****************************************************************
*******
//         fclose(fw);
//         fclose(fp);
```

```
	 for (i=1;i<=M+1;i++)
	{
		for (j=1;j<=M+1;j++)
		{

			fprintf(fw,"p[%d][%d]=%lf",i,j,p[i][j]);

		}
			fprintf(fw,"\n");
	}

	Write_atomZfile("Atom_95B.dat",p);



}

void Write_atomZfile(char *filename, double atomZ[MM][MM])
{
	FILE *fp;

	fp=fopen(filename, "w");
	if (fp==NULL)
	{
		perror("Write_atomZfile");
		exit(1);
	}

	for (int icol=1; icol<=NUM_COL; icol++)
	{
		for(int jrow=1; jrow<=NUM_ROW; jrow++)
		{
			fprintf(fp,"%8.3lf\t", atomZ[icol][jrow]);
		}
		fprintf(fp,"\n");
	}
}


void Swp(double a1, double a2)
{
	double temp;
	temp=a1;
	a1=a2;
	a2=temp;
```

APPENDIX C

JACOBI METHOD FOR SOLVING THE CHARE COMPENSATION MODEL PROGRAM CODE IN C++

```
#include<stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>

#define  MM 920

double q[MM][MM];
double p[MM][MM];
int i,j,n,rem,M,k1,k2,i1,i2,Nd,t;

double temH;
double s;
double esp;
double a;
double d;
double R;
double qi;
int alfa[MM][MM];

double Rd;
double al=0.01;
double B;

int NUM_ROW, NUM_COL;

void Write_atomZfile(char *filename, double atomZ[MM][MM]);

void Swp(double a1, double a2);

void main()
{
        srand(time(NULL));

        n=0;
```

```
	FILE* fp = fopen("data30A.txt", "r");
	FILE* fv = fopen("alfa30A.txt", "w");
	FILE* fw = fopen("output30A.txt", "w");


	fscanf(fp,"%lf",&a);
   fscanf(fp,"%lf",&d);
	fscanf(fp,"%lf",&R);
	fscanf(fp,"%d",&M);
	fscanf(fp,"%lf",&esp);
	fscanf(fp,"%d",&Nd);
	fscanf(fp,"%lf",&qi);

	fprintf(fw,"a=%lf \n",a);
	fprintf(fw,"d=%lf \n",d);
	fprintf(fw,"R=%lf \n",R);
	fprintf(fw,"M=%d \n",M);
	fprintf(fw,"esp=%lf \n",esp);
	fprintf(fw,"Nd=%ld \n",Nd);
	fprintf(fw,"qi=%lf \n",qi);

	printf("a= %lf,d= %lf,R=%lf,M=%d,Nd=%d\n",a,d,R,M,Nd);


	NUM_ROW = M; // Number of atoms in a row
	NUM_COL = M; // Number of atoms in a colomn

	// initializing charge

	for (i=0;i<=M+1;i++)
	{
		for (j=0;j<=M+1;j++)
		{
			p[i][j]=0;
			alfa[i][j]=0;

		}
	}

	for (i=1;i<=Nd;i++)
	{
tryAgain: // this is a statement label
		k1 = rand() % M + 1;
		k2 = rand() % M + 1;


		if (alfa[k1][k2]==1)
```

```
                goto tryAgain; // this is the goto statement
        else
        {
                alfa[k1][k2]=1;

                fprintf(fv,"alfa[%d][%d]=1; \n",k1,k2);

        }
}

//alfa[251][250]=1;



        for (i=0;i<=M+1;i++)
        {
                for (j=0;j<=M+1;j++)
                {
                        q[i][j]=p[i][j];
                }
        }

s=1;
while (s>esp)
{
        n++;

        s=0;



        for (i=1;i<=M;i++)
        {
                for (j=1;j<=M;j++)
                {

                        q[i][j]=q[i+1][j+1]+q[i][j+1]+q[i-1][j]+q[i-1][j-
1]+q[i][j-1]+q[i+1][j];


                        p[i][j]=-R*q[i][j]+a*q[i][j]*q[i][j]*q[i][j]-alfa[i][j]*d;


                        s=s+(p[i][j]-q[i][j])*(p[i][j]-q[i][j]);
```

```
                        q[i][j]=p[i][j];
                }
        }


        fprintf(fw,"s[%d]=%lf\n",n,s);
                printf("s(%d)=%.5f\n",n,s);
        }

        for (i=1;i<=M+1;i++)
        {
                for (j=1;j<=M+1;j++)
                {
                        fprintf(fw,"p[%d][%d]=%lf",i,j,p[i][j]);
                }
                        fprintf(fw,"\n");
        }
        Write_atomZfile("Atom_30A.dat",p);

        //Write_atomZfile("AtomMove_ProjectA.dat",p);

}

void Write_atomZfile(char *filename, double atomZ[MM][MM])
{
        FILE *fp;

        fp=fopen(filename, "w");
        if (fp==NULL)
        {
                perror("Write_atomZfile");
                exit(1);
```

```
	}

	for (int icol=1; icol<=NUM_COL; icol++)
	{
		for(int jrow=1; jrow<=NUM_ROW; jrow++)
		{
			fprintf(fp,"%8.3lf\t", atomZ[icol][jrow]);
		}
		fprintf(fp,"\n");
	}
}



void Swp(double a1, double a2)
{
	double temp;
	temp=a1;
	a1=a2;
	a2=temp;

}
```

APPENDIX D

GAUSS-SEIDEL METHOD FOR SOLVING THE CHARE COMPENSATION MODEL PROGRAM CODE IN C++

```
#include<stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>

#define  MM 510

double q[MM][MM];
double p[MM][MM];
int i,j,n,rem,M,k1,k2,i1,i2,Nd,t;
int icount,left,right,top,bottom;
double s;
double esp;
double a;
double d;
double R;
double qi;
int alfa[MM][MM];
double Rd;
double al=0.01;
double B;

int NUM_ROW, NUM_COL;

void Write_atomZfile(char *filename, double atomZ[MM][MM]);
void Write_atomZfile2(int it, double atomZ[MM][MM]);
void Swp(double a1, double a2);

void main()
{

        srand(time(NULL));
        n=0;
```

```
        FILE* fp = fopen("dataZ.txt", "r");
        FILE* fv = fopen("alfaZ.txt", "w");
        FILE* fw = fopen("outputZ.txt", "w");
        FILE* fz = fopen("atomMoveZ.txt", "w");
        FILE* fm = fopen("outputMoveZ.txt", "w");

        fscanf(fp,"%lf",&a);
   fscanf(fp,"%lf",&d);
        fscanf(fp,"%lf",&R);
        fscanf(fp,"%d",&M);
        fscanf(fp,"%lf",&esp);
        fscanf(fp,"%d",&Nd);
        fscanf(fp,"%lf",&qi);

        fprintf(fw,"a=%lf \n",a);
        fprintf(fw,"d=%lf \n",d);
        fprintf(fw,"R=%lf \n",R);
        fprintf(fw,"M=%d \n",M);
        fprintf(fw,"esp=%lf \n",esp);
        fprintf(fw,"Nd=%ld \n",Nd);
        fprintf(fw,"qi=%lf \n",qi);

        printf("a= %lf,d= %lf,R=%lf,M=%d,Nd=%d\n",a,d,R,M,Nd);


        NUM_ROW = M; // Number of atoms in a row
        NUM_COL = M; // Number of atoms in a colomn

        // initializing charge

        for (i=0;i<=M+1;i++)
        {
                for (j=0;j<=M+1;j++)
                {
                        p[i][j]=0;
                        alfa[i][j]=0;
                }
        }

        for (i=1;i<=Nd;i++)
        {
tryAgain: // this is a statement label
                k1 = rand() % M + 1;
                k2 = rand() % M + 1;

        printf("defect is [%d][%d] \n",k1,k2);
                fprintf(fv,"defect is [%d][%d] \n",k1,k2);
```

```
                printf("alfa[%d][%d]=%d \n",k1,k2,alfa[k1][k2]);
                fprintf(fv,"alfa is [%d][%d]=%d \n",k1,k2,alfa[k1][k2]);

                printf("i=[%d] \n",i);
                fprintf(fv,"i=[%d]  \n",i);

                if (alfa[k1][k2]==1)
                        goto tryAgain; // this is the goto statement
                else
                {
                        alfa[k1][k2]=1;
                        printf("defect is [%d][%d] \n",k1,k2);
                        fprintf(fv,"defect is [%d][%d] \n",k1,k2);
                }
        }

//alfa[250][250]=1;

        s=1;
        while (s>esp)
        {
                n++;
                for (i=1;i<=M+1;i++)
                {
                        for (j=1;j<=M+1;j++)
                        {
                                q[i][j]=p[i][j];
                        }
                }
                for (i=1;i<=M;i++)
                {
                        for (j=1;j<=M;j++)
                        {

                                p[i][j]=p[i+1][j+1]+p[i][j+1]+p[i+1][j]+p[i-1][j]+p[i-
1][j-1]+p[i][j-1];
                                //p[i][j]=p[i-1][j]+p[i-1][j+1]+p[i][j-
1]+p[i][j+1]+p[i+1][j-1]+p[i+1][j];
                                p[i][j]=-R*p[i][j]+a*p[i][j]*p[i][j]*p[i][j]-alfa[i][j]*d;


                        //      printf("p[%d][%d]=%lf\n",i,j,p[i][j]);
                        }
                }

        s=0;
```

```
                for (i=1;i<=M;i++)
                {
                        for (j=1;j<=M;j++)
                        {
                                s=s+(p[i][j]-q[i][j])*(p[i][j]-q[i][j]);
                        }
                }

                fprintf(fw,"s[%d]=%lf\n",n,s);

                printf("s(%d)=%.5f\n",n,s);

                //getch();
        }

//      fclose(fw);
//      fclose(fp);

         for (i=1;i<=M+1;i++)
        {
                for (j=1;j<=M+1;j++)
                {

                        fprintf(fw,"p[%d][%d]=%lf",i,j,p[i][j]);

                }
                        fprintf(fw,"\n");
        }

        Write_atomZfile("Atom_E.dat",p);


/*      printf("test\n");
        s=1;
        n=0;
        while (s>esp)
        {
                n++;
                for (i=1;i<=M;i++)
                {
                        for (j=1;j<=M;j++)
                        {
                                q[i][j]=p[i][j];
                        }
                }
```

```
printf("n=%d\n",n);

        if (n%4==1)
        {
                top = M;
                right = M;
                left = 0;
                bottom = 1;

                icount = 0;
                i = 1;

               goto st1;

        }
        else if(n%4==2)
        {
                top = M;
                right = M;
                left = 1;
                bottom = 0;

                icount = 0;
                i = 1;
               goto st2;

        }
        else if(n%4==3)
        {
                top = M;
                right = M+1;
                left = 1;
                bottom = 1;

                icount = 0;
                i = 1;
               goto st3;

        }
        else if(n%4==0)
        {
                top = M+1;
                right = M;
                left = 1;
                bottom = 1;

                icount = 0;
```

```
                i = 1;
                goto st4;

        }

        while (icount < M*M)
        {
//      printf("icount=%d",icount);

st1:
                if (icount < M*M)
                {
                        j = bottom;
                        left++;

                        for (i = left; i <= right; i++)
                        {

p[i][j]=p[i+1][j+1]+p[i][j+1]+p[i+1][j]+p[i-1][j]+p[i-1][j-1]+p[i][j-1];
                                p[i][j]=-
R*p[i][j]+a*p[i][j]*p[i][j]*p[i][j]-alfa[i][j]*d;

                                icount++;
                        }
                }

st2:

                if (icount < M*M)
                {
                        i = right;
                        bottom++;

                        for (j = bottom; j <= top; j++)
                        {

p[i][j]=p[i+1][j+1]+p[i][j+1]+p[i+1][j]+p[i-1][j]+p[i-1][j-1]+p[i][j-1];
                                p[i][j]=-
R*p[i][j]+a*p[i][j]*p[i][j]*p[i][j]-alfa[i][j]*d;

                                icount++;
                        }
                }

st3:

                if (icount < M*M)
```

```
                {
                        j = top;
                        right--;

                        for (i = right; i >= left; i--)
                        {

        p[i][j]=p[i+1][j+1]+p[i][j+1]+p[i+1][j]+p[i-1][j]+p[i-1][j-1]+p[i][j-1];
                                                p[i][j]=-
R*p[i][j]+a*p[i][j]*p[i][j]*p[i][j]-alfa[i][j]*d;

                                                icount++;
                        }
                }

st4:
                if (icount < M*M)
                        {
                        i = left;
                        top--;

                        for (j = top; j >= bottom; j--)
                        {

        p[i][j]=p[i+1][j+1]+p[i][j+1]+p[i+1][j]+p[i-1][j]+p[i-1][j-1]+p[i][j-1];
                                                p[i][j]=-
R*p[i][j]+a*p[i][j]*p[i][j]*p[i][j]-alfa[i][j]*d;

                                                icount++;
                        }
                        }

                }


                s=0;
                for (i=1;i<=M;i++)
                {
                        for (j=1;j<=M;j++)
                        {
                                s=s+(p[i][j]-q[i][j])*(p[i][j]-q[i][j]);
                        }
                }

                fprintf(fw,"s[%d]=%lf\n",n,s);
                printf("s(%d)=%.5f\n",n,s);
```

```
                //getch();
        }

//      fclose(fw);
//      fclose(fp);

        for (i=1;i<=M;i++)
        {
                for (j=1;j<=M;j++)
                {
                        fprintf(fw,"p[%d][%d]=%lf ",i,j,p[i][j]);
                }
                        fprintf(fw,"\n");
        }

        Write_atomZfile("Atom_Z.dat",p);*/
/*      t=0;
   while (t<10)
// check Is atom defect ?
        {
     t++;

        for (i=1;i<=M+1;i++)
        {
                for (j=1;j<=M+1;j++)
                {

                        if (alfa[i][j]==1)
                        {
                                printf("alfa[%d][%d]=%d\n",i,j,alfa[i][j]);
                                fprintf(fz,"alfa[%d][%d]=%d\n",i,j,alfa[i][j]);
                                fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

                                //getchar();



                tryAgain1: // this is a statement label

                                                        i1 = rand() % 6+1;
                                                        i2 = rand() % 1000+1;

                                                                printf("random no. is [%d]
\n",i1);
                                                                fprintf(fz,"random no is [%d]
\n",i1);
```

```
if (i1==1)
{
        if(alfa[i-1][j]==0)
        {
                Rd=double(i2)/1000.0;
B=exp(-al*(p[i][j]-p[i-1][j])*(p[i][j]-p[i-1][j]));
                if (B < Rd)
                {
                        goto newAtom;
                }
                else
                {
fprintf(fz,"p[%d][%d]=%d\n",i-1,j,p[i-1][j]);
                        alfa[i][j]=0     ;
                        alfa[i-1][j]=1   ;
                Swp(p[i][j],p[i-1][j]);
fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);
fprintf(fz,"p[%d][%d]=%d\n",i-1,j,p[i-1][j]);
fprintf(fz,"Rd=%lf\n",Rd);
fprintf(fz,"B=%lf\n",B);
                }
        }
        else
        {
                goto tryAgain1;
        }
}
else if (i1==2)
{
        if(alfa[i-1][j+1]==0)
```

```
                                        {
                                                Rd=double(i2)/1000.0;
                                                B=exp(-al*(p[i][j]-p[i-
1][j+1])*(p[i][j]-p[i-1][j+1]));
                                                if (B < Rd)
                                                {
                                                        goto newAtom;

                                                }
                                                else
                                                {

        fprintf(fz,"p[%d][%d]=%d\n",i-1,j+1,p[i-1][j+1]);
                                                        alfa[i][j]=0    ;
                                                        alfa[i-1][j+1]=1
        ;
                                                Swp(p[i][j],p[i-1][j+1]);

        fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

        fprintf(fz,"p[%d][%d]=%d\n",i-1,j+1,p[i-1][j+1]);

        fprintf(fz,"Rd=%lf\n",Rd);

        fprintf(fz,"B=%lf\n",B);
                                                }
                                        }
                                        else
                                        {
                                                goto tryAgain1;
                                        }
                                }

                                else if (i1==3)

                                {
                                        if(alfa[i][j+1]==0)

                                        {
                                                Rd=double(i2)/1000.0;

                        B=exp(-al*(p[i][j]-p[i][j+1])*(p[i][j]-p[i][j+1]));

                                                if (B < Rd)
                                                {
                                                        goto newAtom;
```

```
                                        }
                                        else
                                        {

fprintf(fz,"p[%d][%d]=%d\n",i,j+1,p[i][j+1]);
                                                alfa[i][j]=0    ;
                                                alfa[i][j+1]=1  ;

Swp(p[i][j],p[i][j+1]);

fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

fprintf(fz,"p[%d][%d]=%d\n",i,j+1,p[i][j+1]);

fprintf(fz,"Rd=%lf\n",Rd);

fprintf(fz,"B=%lf\n",B);
                                        }
                                }
                                else
                                {
                                        goto tryAgain1;
                                }
                        }

                else if (i1==4)

                {
                                if(alfa[i+1][j]==0)

                                {
                                        Rd=double(i2)/1000.0;

        B=exp(-al*(p[i][j]-p[i+1][j])*(p[i][j]-p[i+1][j]));

                                        if (B < Rd)
                                        {
                                                goto newAtom;

                                        }
                                        else
                                        {

fprintf(fz,"p[%d][%d]=%d\n",i+1,j,p[i+1][j]);
                                                alfa[i][j]=0    ;
                                                alfa[i+1][j]=1  ;
```

```
Swp(p[i][j],p[i+1][j]);

fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

fprintf(fz,"p[%d][%d]=%d\n",i+1,j,p[i+1][j]);

fprintf(fz,"Rd=%lf\n",Rd);

fprintf(fz,"B=%lf\n",B);
                                        }
                                }
                                else
                                {
                                        goto tryAgain1;
                                }
                        }

                        else if (i1==5)

                        {
                                if(alfa[i+1][j-1]==0)

                                {
                                        Rd=double(i2)/1000.0;
                                        B=exp(-al*(p[i][j]-
p[i+1][j-1])*(p[i][j]-p[i+1][j-1]));

                                        if (B < Rd)
                                        {
                                                goto newAtom;

                                        }
                                        else
                                        {

fprintf(fz,"p[%d][%d]=%d\n",i+1,j-1,p[i+1][j-1]);
                                                alfa[i][j]=0    ;
                                                alfa[i+1][j-1]=1
;

Swp(p[i][j],p[i+1][j-1]);

fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

fprintf(fz,"p[%d][%d]=%d\n",i+1,j-1,p[i+1][j-1]);

fprintf(fz,"Rd=%lf\n",Rd);
```

```
fprintf(fz,"B=%lf\n",B);
                                        }
                                }
                                else
                                {
                                        goto tryAgain1;
                                }
                        }

                        else if (i1==6)
                        {
                                if(alfa[i][j-1]==0)

                                {
                                        Rd=double(i2)/1000.0;
                B=exp(-al*(p[i][j]-p[i][j-1])*(p[i][j]-p[i][j-1]));
                                        if (B < Rd)
                                        {
                                                goto newAtom;

                                        }
                                        else
                                        {

fprintf(fz,"p[%d][%d]=%d\n",i,j-1,p[i][j-1]);
                                                alfa[i][j]=0    ;
                                                alfa[i][j-1]=1  ;
                                        Swp(p[i][j],p[i][j-1]);

fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

fprintf(fz,"p[%d][%d]=%d\n",i,j-1,p[i][j-1]);

fprintf(fz,"Rd=%lf\n",Rd);

fprintf(fz,"B=%lf\n",B);
                                        }
                                }
                                else
                                {
                                        goto tryAgain1;
                                }
                        }


                fprintf(fz,"alfa[%d][%d]=%lf\n",i,j,alfa[i][j]);
```

```
                                printf("alfa[%d][%d]=%lf\n",i,j,alfa[i][j]);

                        }

                        newAtom: ;

                }

        }

        s=1;
        n=0;
        while (s>esp)
        {
                n++;

                for (i=1;i<=M+1;i++)
                {
                        for (j=1;j<=M+1;j++)
                        {
                                q[i][j]=p[i][j];
                        }
                }
                for (i=1;i<=M;i++)
                {
                        for (j=1;j<=M;j++)
                        {

                p[i][j]=p[i-1][j]+p[i-1][j+1]+p[i][j-1]+p[i][j+1]+p[i+1][j-1]+p[i+1][j];

                p[i][j]=-R*p[i][j]+a*p[i][j]*p[i][j]*p[i][j]-alfa[i][j]*d;


                        //      printf("p[%d][%d]=%lf\n",i,j,p[i][j]);
                        }
                }

        s=0;
                for (i=1;i<=M;i++)
                {
                        for (j=1;j<=M;j++)
                        {
                                s=s+(p[i][j]-q[i][j])*(p[i][j]-q[i][j]);
                        }
                }

                fprintf(fm,"s[%d]=%lf\n",n,s);
```

```
            printf("s(%d)=%.5f\n",n,s);

            //getch();
        }

        fclose(fw);
        fclose(fp);


        //Write_atomZfile("AtomMove_K.dat",p);

        fprintf(fz,"************************************************** \n");

         Write_atomZfile2(t,p);



}*/
        t=0;
   while (t<10)
// check Is atom defect ?
        {
     t++;

        for (i=1;i<=M+1;i++)
        {
            for (j=1;j<=M+1;j++)
            {

                if (alfa[i][j]==1)
                {
                    printf("alfa[%d][%d]=%d\n",i,j,alfa[i][j]);
                    fprintf(fz,"alfa[%d][%d]=%d\n",i,j,alfa[i][j]);
                    fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

                    //getchar();



            tryAgain1: // this is a statement label

                                    i1 = rand() % 6+1;
                                    i2 = rand() % 1000+1;

                                    printf("random no. is [%d] \n",i1);
```

```
                    fprintf(fz,"random no is [%d] \n",i1);


                    if (i1==1)

                    {
                        if(alfa[i+1][j+1]==0)

                        {
                            Rd=double(i2)/1000.0;

                B=exp(-al*(p[i][j]-p[i+1][j+1])*(p[i][j]-p[i+1][j+1]));

                            if (B < Rd)
                            {
                                goto newAtom;

                            }
                            else

                            {

fprintf(fz,"p[%d][%d]=%d\n",i-1,j,p[i+1][j+1]);
                                alfa[i][j]=0     ;
                                alfa[i-1][j]=1   ;

                              Swp(p[i][j],p[i-1][j]);

fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

fprintf(fz,"p[%d][%d]=%d\n",i-1,j,p[i+1][j+1]);

fprintf(fz,"Rd=%lf\n",Rd);

fprintf(fz,"B=%lf\n",B);

                            }

                        }
                        else
                        {
                            goto tryAgain1;
                        }
                    }

                    else if (i1==2)
```

```
{
        if(alfa[i][j+1]==0)

        {
                Rd=double(i2)/1000.0;

B=exp(-al*(p[i][j]-p[i][j+1])*(p[i][j]-p[i][j+1]));

                if (B < Rd)
                {
                        goto newAtom;

                }
                else
                {

fprintf(fz,"p[%d][%d]=%d\n",i-1,j+1,p[i][j+1]);
                        alfa[i][j]=0     ;

                        alfa[i-1][j+1]=1
                        ;
                Swp(p[i][j],p[i-1][j+1]);

fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

fprintf(fz,"p[%d][%d]=%d\n",i-1,j+1,p[i][j+1]);

fprintf(fz,"Rd=%lf\n",Rd);

fprintf(fz,"B=%lf\n",B);
                }
        }
        else
        {
                goto tryAgain1;
        }
}

else if (i1==3)

{
        if(alfa[i-1][j]==0)

        {
                Rd=double(i2)/1000.0;
B=exp(-al*(p[i][j]-p[i-1][j])*(p[i][j]-p[i-1][j]));
                if (B < Rd)
```

```
                                                {
                                                        goto newAtom;

                                                }
                                                else
                                                {

fprintf(fz,"p[%d][%d]=%d\n",i,j+1,p[i-1][j]);
                                                        alfa[i][j]=0     ;
                                                        alfa[i][j+1]=1  ;

Swp(p[i][j],p[i][j+1]);

fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

fprintf(fz,"p[%d][%d]=%d\n",i,j+1,p[i-1][j]);

fprintf(fz,"Rd=%lf\n",Rd);

fprintf(fz,"B=%lf\n",B);
                                                }
                                        }
                                        else
                                        {
                                                goto tryAgain1;
                                        }
                                }

                                else if (i1==4)

                                {
                                        if(alfa[i-1][j-1]==0)

                                        {
                                                Rd=double(i2)/1000.0;

                        B=exp(-al*(p[i][j]-p[i-1][j-1])*(p[i][j]-p[i-1][j-1]));
                                                if (B < Rd)
                                                {
                                                        goto newAtom;

                                                }
                                                else
                                                {

fprintf(fz,"p[%d][%d]=%d\n",i+1,j,p[i-1][j-1]);
                                                        alfa[i][j]=0     ;
```

```
                                                alfa[i+1][j]=1  ;
Swp(p[i][j],p[i+1][j]);
fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);
fprintf(fz,"p[%d][%d]=%d\n",i+1,j,p[i-1][j-1]);
fprintf(fz,"Rd=%lf\n",Rd);
fprintf(fz,"B=%lf\n",B);
                                        }
                                }
                                else
                                {
                                        goto tryAgain1;
                                }
                        }
                        else if (i1==5)
                        {
                                if(alfa[i][j-1]==0)
                                {
                                        Rd=double(i2)/1000.0;
                                B=exp(-al*(p[i][j]-p[i][j-1])*(p[i][j]-p[i][j-1]));
                                        if (B < Rd)
                                        {
                                                goto newAtom;
                                        }
                                        else
                                        {
fprintf(fz,"p[%d][%d]=%d\n",i+1,j-1,p[i][j-1]);
                                                alfa[i][j]=0     ;
                                                alfa[i+1][j-1]=1
;
Swp(p[i][j],p[i+1][j-1]);
fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);
fprintf(fz,"p[%d][%d]=%d\n",i+1,j-1,p[i][j-1]);
```

```
fprintf(fz,"Rd=%lf\n",Rd);

fprintf(fz,"B=%lf\n",B);
                                        }
                                }
                                else
                                {
                                        goto tryAgain1;
                                }
                        }

                        else if (i1==6)
                        {
                                if(alfa[i][j-1]==0)

                                {
                                        Rd=double(i2)/1000.0;

                        B=exp(-al*(p[i][j]-p[i][j-1])*(p[i][j]-p[i][j-1]));

                                if (B < Rd)
                                        {
                                                goto newAtom;

                                        }
                                        else
                                        {

fprintf(fz,"p[%d][%d]=%d\n",i,j-1,p[i][j-1]);
                                                alfa[i][j]=0    ;
                                                alfa[i][j-1]=1  ;
                                        Swp(p[i][j],p[i][j-1]);

fprintf(fz,"p[%d][%d]=%d\n",i,j,p[i][j]);

fprintf(fz,"p[%d][%d]=%d\n",i,j-1,p[i][j-1]);

fprintf(fz,"Rd=%lf\n",Rd);

fprintf(fz,"B=%lf\n",B);
                                        }
                                }
                                else
                                {
                                        goto tryAgain1;
                                }
```

```
                                        }

                                        fprintf(fz,"alfa[%d][%d]=%lf\n",i,j,alfa[i][j]);
                                        printf("alfa[%d][%d]=%lf\n",i,j,alfa[i][j]);

                        }

                        newAtom: ;
                }

        }

        s=1;
        n=0;
        while (s>esp)
        {
                n++;

                for (i=1;i<=M+1;i++)
                {
                        for (j=1;j<=M+1;j++)
                        {
                                q[i][j]=p[i][j];
                        }
                }
                for (i=1;i<=M;i++)
                {
                        for (j=1;j<=M;j++)
                        {

                                p[i][j]=p[i+1][j+1]+p[i][j+1]+p[i+1][j]+p[i-1][j]+p[i-
1][j-1]+p[i][j-1];
                                //p[i][j]=p[i-1][j]+p[i-1][j+1]+p[i][j-
1]+p[i][j+1]+p[i+1][j-1]+p[i+1][j];
                                p[i][j]=-R*p[i][j]+a*p[i][j]*p[i][j]*p[i][j]-alfa[i][j]*d;


                        //      printf("p[%d][%d]=%lf\n",i,j,p[i][j]);
                        }
                }

        s=0;
                for (i=1;i<=M;i++)
                {
                        for (j=1;j<=M;j++)
                        {
                                s=s+(p[i][j]-q[i][j])*(p[i][j]-q[i][j]);
```

```
                    }
                }
                fprintf(fm,"s[%d]=%lf\n",n,s);

                printf("s(%d)=%.5f\n",n,s);

                //getch();
        }

        fclose(fw);
        fclose(fp);


        //Write_atomZfile("AtomMove_ProjectA.dat",p);

        fprintf(fz,"************************************************** \n");

         Write_atomZfile2(t,p);

}
}


void Write_atomZfile(char *filename, double atomZ[MM][MM])
{
        FILE *fp;

        fp=fopen(filename, "w");
        if (fp==NULL)
        {
                perror("Write_atomZfile");
                exit(1);
        }

        for (int icol=1; icol<=NUM_COL; icol++)
        {
                for(int jrow=1; jrow<=NUM_ROW; jrow++)
                {
                        fprintf(fp,"%8.3lf\t", atomZ[icol][jrow]);
                }
                fprintf(fp,"\n");
        }
}
void Write_atomZfile2(int it, double atomZ[MM][MM])
{
        char *filename;
        char buffer[20];
```

```
        char str1[20];

        FILE *fp;

    itoa(it,buffer,10);
         strcpy(str1,"AtomMove_E");
         filename=strcat(str1,buffer);

        fp=fopen(filename, "w");
        if (fp==NULL)
        {
                perror("Write_atomZfile");
                exit(1);
        }

        for (int icol=0; icol<NUM_COL; icol++)
        {
                for(int jrow=0; jrow<NUM_ROW; jrow++)
                {
                        fprintf(fp,"%8.3lf\t", atomZ[icol][jrow]);
                }
                fprintf(fp,"\n");

        }

}

void Swp(double a1, double a2)
{
        double temp;
        temp=a1;
        a1=a2;
        a2=temp;

}
```

CURRICULUM VITAE

NAME: Mr. Charun Pidduangkeaw

DATE OF BIRTH: July 24, 1979 (Thai)

E-MAIL: runma111@hotmail.com, runma123@yahoo.com

EDUCATIONAL BACKGROUND:

Bachelor Degree of Engineering in Electronics Engineering, King Mongkut's Institute of Technology Ladkrabang (KMITL), Bankok, Thailand, 2000.

WORK EXPERIENCE:

HITACHI CONSUMER PRODUCT(THAILAND), Engineer, Prachinburi, Thailand, 2000.

TELECOMASIA CORPORATION PUBLIC CO.,LTD. (TRUE CORPORATION), Engineer, Section of Public Phone Engineering, Bangkok, Thailand, 2001-2005.

SEAGATE TECHNOLOGY THAILAND, Engineer, Department of Test Engneering, Nakhonratchasima, Thailand, 2005-2008.

RATCHADA ACADEMIC CENTER (RAC), Manager, Nakhonratchasima and Phetchaboon, Thailand, 2008-2010.

MATH GENIUS, Manager and President, Nakhonratchasima, Thailand, 2010-present.