STRUCTURE FORMATION ON A QUASICRYSTALLINE SUBSTRATE

by

Burak Bilki

B.S., Mechanical Engineering, Boğaziçi University, 2001

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Physics

Boğaziçi University

2006

ACKNOWLEDGEMENTS

I would like to put here a few sentences such that I will recall how a scientist -both as an instructor and a researcher- must be and how must not be, when I glance at these pages several years later. This thesis has been a hard work. Yet the masters program as a whole has been harder. With "hard", I specifically mean personal difficulties I encountered rather than intellectual. Although I sometimes think that this period has cost me much more than it deserves, it has a significant contribution to my life.

First, I would like to thank the Department of Mechanical Engineering. There, I had the opportunity to meet many instructors I will always admire not only for their extraordinary aptitude for engineering but also for their attitude to life in general.

I would like to thank the Physics Department for employing me as a research assistant and endeavoring to supply the necessary conditions for my study to the possible extent. I would like to present my very special thanks to Prof. Avadis Hacınlıyan and Prof. Haluk Beker who turned me into a real physicist.

I am grateful to our collaborators in ETHZ, specifically Prof. Mehmet Erbudak for the invaluable research experience he provided and for his kind hospitality in Zurich. I would like to thank Prof. Teoman Turgut (Director) in the name of Feza Gürsey Institute for supplying me a user account in the computer cluster Gilgamesh at which almost all of the calculations were performed.

Finally, I would like to thank my family but the words will not be enough to express how lucky I am having such a family. They provided a vital amount of encouragement throughout this study and they deserve the degree as much as I do.

ABSTRACT

STRUCTURE FORMATION ON A QUASICRYSTALLINE SUBSTRATE

In recent Low Energy Electron Diffraction (LEED) Experiments by Flückiger et al. on periodic Al films grown on a quasicrystalline substrate, it was observed that depending on the thickness of the Al film grown, the structure of the Al superlattice consists of domains of hexagonal symmetry that are oriented with respect to each other at angles commensurate with the underlying decagonal structure of the substrate [1].

The structural properties of a periodic film grown on a quasi-periodic substrate was studied using molecular dynamics simulations with simulated annealing. The simulations were set up using the atomic coordinates of the quasicrystalline structure as obtained from the diffraction experiments by Saitoh *et al.* [2].

It is found that the structure formed exhibits translational and rotational effects resulting from the underlying quasicrystalline substrate. The effects of changing the parametrization in simulations were also investigated and no significant change in the characteristics of the final structure formed was encountered. The Fourier investigations are in compliance with the diffraction experiment results of [1].

ÖZET

KUAZİKRİSTAL SUBSTRAT ÜZERİNDE YAPI OLUŞUMU

Yakın geçmişte, kuazikristal substrat üzerinde geliştirilen periyodik Al filmleri üzerine Flückiger ve diğerleri tarafından gerçekleştirilen düşük enerjili elektron saçılması (LEED) deneylerinde, geliştirilen Al filmin kalınlığına balı olarak, Al kafesinin yapısında, birbirlerine göre, alt katmandaki substratın onlu yapısıyla uyumlu açılar yapacak şekilde yönlenmiş altılı simetrik alanlar bulunduğu gözlenmiştir [1].

Kuaziperiyodik bir substrat üzerinde geliştirilen periyodik bir filmin yapısal özellikleri, simüle edilmiş tavlamalı moleküler dinamik simülasyonları kullanılarak çalışılmıştır. Simülasyonlar, Saitoh ve diğerlerinin saçılma deneylerinde elde ettikleri kuazikristal yapıların atomik koordinatları kullanılarak oluşturulmuştur [2].

Oluşan yapının, alt katmandaki kuazikristal substrattan kaynaklanan öteleme ve dönme etkileri sergilediği anlaşılmıştır. Simülasyonlardaki parametrizasyonun değişmesinin etkileri de araştırılmış ve oluşan nihai yapının karakteristiklerinde belirgin bir değişikliğe rastlanmamıştır. Fourier incelemeleri, [1]'taki saçılma deneyi sonuçları ile uyumludur.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . x
LIST OF SYMBOLS/ABBREVIATIONS . . . xi
1. INTRODUCTION . . . 1
2. QUASICRYSTALLINE STRUCTURE . . . 3
2.1. Quasicrystal Geometry . . . 3
2.1.1. Theory of Functions . . . 3
2.1.2. Tiles and Tilings . . . 4
2.2. One Dimensional Quasicrystals . . . 5
2.3. Higher Dimensional Quasicrystals . . . 9
3. SIMULATIONS AND PARAMETERS . . . 11
3.1. Quasicrystalline Substrate . . . 11
3.2. Particle Dynamics . . . 12
4. COMPUTATIONAL ANALYSIS AND RESULTS . . . 17
4.1. Adsorbate Layer Parameters . . . 17
4.2. Substrate Layer . . . 18
4.3. Numerical Implementation . . . 18
4.4. Simulation Results . . . 21
4.4.1. Translational Effect . . . 21
4.4.2. Rotational Effects . . . 22
4.4.3. Multiple Domain Results . . . 24
5. CONCLUSION AND DISCUSSION . . . 26
APPENDIX A: MD SIMULATIONS WITH SIMULATED ANNEALING . . . 31
A.1. Definition . . . 31
A.2. Simulation Flow . . . 32
A.2.1. Preliminary Analysis . . . 32

A.2.1.1. Boundary Conditions 32
A.2.1.2. Initial Conditions 34
A.2.1.3. Conservation Laws 34
A.2.2. Simulation Algorithm 35
A.2.2.1. Initialization 35
A.2.2.2. Computation of Forces 36
A.2.2.3. Integration 36
A.2.2.4. Measurements And Control 37
A.3. Test Code For Pure Adsorbate Layer 38
A.4. Simulating A Simple System 48
A.5. Simulation LJ Potential 50
APPENDIX B: SURFACE POTENTIAL OF THE SUBSTRATE LAYER . . 52
APPENDIX C: FOURIER TRANSFORM 54
APPENDIX D: DOMAIN STRUCTURE ANALYSIS 60
REFERENCES 65

LIST OF FIGURES

Figure 2.1. The three dimensional representation of the unit cell used to create a periodic crystal in five dimensions (a) and the resultant projection in two dimensions (b) which shows quasiperiodic structure 4

Figure 2.2. Two rhombi with equal side length but different opening angles of 72° ("fat" rhombus) and 36° ("thin" rhombus). Arrow decorations encode the matching rules 5

Figure 2.3. An example of a rhombic Penrose tiling formed by appropriate matching of fat and thin rhombi 6

Figure 2.4. The chair is substituted by four self similar chairs. These four prototiles are congruent but differ in orientation 9

Figure 3.1. Plot of the original substrate layers. Layer 1 is on the surface of the specimen and layer 2 is 0.5 Å deeper. The unit cell cut from this configuration is denoted by the solid square 12

Figure 3.2. Unit cell cut from the original configuration to be used for simulations: The surface layer (a) and its diffraction pattern (b); the second monolayer that is 0.5 Å deeper from the surface (c) and its diffraction pattern (d); the two layers plotted together (e) (red denotes the surface layer and green denotes the deeper layer) and their diffraction pattern (f) 13

Figure 4.1. Contour plot of the potential due to the substrate layer (a) and the simultaneous plot of the potential contours and the atomic coordinates of the substrate layer atoms 19

Figure 4.2. Overall potential of the substrate layer 20

Figure 4.3. Two simulation outputs [(a) and (b)], their superposition exhibiting the translational effect (c), and the corresponding diffraction pattern (d) 22

Figure 4.4. Two simulation outputs [(a) and (b)], their superposition exhibiting the rotational effect by 12° (c), and the corresponding diffraction pattern (d) 23

Figure 4.5. Two simulation outputs [(a) and (b)], their superposition exhibiting the rotational effect by 2.5° (c), and the corresponding diffraction pattern (d) 24

Figure 4.6. The sample domain structure plot for a simulation output 25

Figure 5.1. Superposition of the atomic configurations exhibiting translational effect (see Fig. 4.3c) plotted on the surface potential due to the substrate layer 28

Figure 5.2. Superposition of the atomic configurations exhibiting rotational effect by 12° (see Fig. 4.4c) and by 2.5° (see Fig. 4.5c) plotted on the surface potential due to the substrate layer 29

Figure 5.3. The diffraction pattern of a combined structure 29

Figure A.1. The flowchart of the MD simulations with simulated annealing . . 35

Figure A.2. Standard LJ potential with the parameters adjusted for the simulations 50

LIST OF TABLES

Table A.4. Sample simulation run output for a pure adsorbate system. 49

LIST OF SYMBOLS/ABBREVIATIONS

$\vec{a}_i$	Acceleration vector of atom i
A	Amplitude of the diffracted wave
E	Total energy
f_i	ith Fibonacci number
$\vec{F}$	Interatomic force
$F(\vec{k})$	Structure factor
$I(\vec{k})$	Intensity distibution/Diffraction pattern
$\vec{k}$	Reciprocal space unit vector
k	Boltzman's constant
K	Kinetic energy
L	Size of the simulation unit cell (square)
m_i	Mass of atom i
M	Matrix generator of the Fibonacci sequence
N_a	Number of adsorbate atoms
N_s	Number of substrate atoms
$\vec{r}_i$	Position vector of atom i
r_{ij}	Distance between atom i and atom j
r_x	x-component of the interatomic displacement vector $\vec{r}$
r_y	y-component of the interatomic displacement vector $\vec{r}$
r_{xi}	x-component of the position vector of atom i
r_{yi}	y-component of the position vector of atom i
$\ddot{\vec{r}}$	Acceleration vector
T	Temperature
$\vec{v}_i$	Velocity vector of atom i
v_i	Speed of atom i
V	Interatomic potential
$\vec{\Delta}$	Shift vector
Δt	Time stepsize

ϵ	Strength of LJ interaction
σ	Range of LJ interaction
τ	The Golden Mean
C-IC	Commensurate-incommensurate
LEED	Low Energy Electron Diffraction
LJ	Lennard Jones
MD	Molecular dynamics

1. INTRODUCTION

The concept of an elastic lattice interacting with a rigid substrate lattice producing commensurate and incommensurate transitions when the periodicities of the two lattices match or mismatch is found in numerous condensed matter systems, including atoms adsorbed on surfaces [3], layered superconductors and superconducting networks and Josephson-junction arrays [4]. The elastic lattice imposes a super-periodicity with respect to the rigid substrate. If this super-periodicity is a simple rational multiple of the basic unit cell, one speaks of the commensurate (lock-in) state. In the incommensurate state, the ratio of the super-periodicity repeat distance to the basic lattice repeat distance is irrational and may show continuous variation [5]. When the density of such condensed matter systems changes, the system commonly undergoes phase transitions between the commensurate and incommensurate phases controlled by the relative strengths of pinning and interaction [6]. Such commensurate-incommensurate (C-IC) transitions have been studied numerically for several different conditions ([7], [8]).

Phase transformation phenomena for quasi-two-dimensional "disordered" phases adsorbed on solid surfaces has been an area of intense research. When weakly incommensurate and commensurate phases of quasi-two-dimensional adsorbates on periodiclattices are inspected, it is found that the weakly incommensurate phase is characterized by large variations of separation and orientation between domain walls, a broad distribution of commensurate islands in both size and relative location to one another, and the lack of any long range atomic correlation [7]. The incommensurate phase consists of commensurate islands separated by an interconnecting network of incommensurate domain walls, the structure of this network being a sensitive function of temperature and coverage [8]. For example, if graphite is used as the periodic substrate and Kr is used as the quasi-two-dimensional adsorbate, the domain walls may be in three different directions because of the hexagonal substrate [3]. The incommensurate phase is mainly characterized by the honeycomb network of domain walls.

The structure formation on corrugated substrates has also been investigated. An example of this kind would be N_2 adsorbed on highly corrugated, anisotropic $Cu(110)$ surface [9]. The resulting superstructure is a remarkably stable, high-order commensurate phase with an oblique unit cell. The molecules are oriented such that the centers of mass are arranged along the substrate troughs in a seven sublattice pinwheel structure, demonstrating the strong influence of the substrate corrugation and anisotropy on the orientational ordering.

In this thesis we will study the structure formation on a quasi-periodic substrate. Quasicrystals have no definite unit cell in two dimensions and no long range translational order but self similar repeating structures with different orientations [10], the adsorbate is a periodic structure if it crystallizes by itself. The problem is treated with molecular dynamics simulations with simulated annealing.

2. QUASICRYSTALLINE STRUCTURE

A perfect crystal is considered to be constructed by the infinite regular repetition in space of identical structural units or building blocks. The properties of periodic arrangements of atoms are of central importance in solid state physics. However, the world of solid state science was startled by the announcement of the discovery of a "metallic phase with long-range orientational order and no translational symmetry". The material in question was an alloy of Aluminum and Manganese. Its diffraction images showed icosahedral symmetry, long believed to be impossible for matter in the crystalline state [10]. Since translational symmetry and icosahedral symmetry are incompatible, the new metallic phase was something new and strange. This section deals with the basic properties and a brief glossary of these strange crystal structures: Quasicrystals.

The discovery of alloys with long range orientational order and sharp diffraction images of non-crystallographic symmetry has initiated an intensive investigation of the possible structures and physical properties of such systems. In order to understand the crystallographic structure and properties of quasicrystals, two important mathematical tools that had already been completed before quasicrystals were discovered and became upmost important while investigating the quasiperiodic structure must be mentioned.

2.1. Quasicrystal Geometry

2.1.1. Theory of Functions

Quasiperiodic functions had already been studied by P. Bohl in 1893 and N. Escanglon in 1919 [11]. They are distinguished from non-periodic functions by having a finite number of linearly independent basis vectors which span its Fourier module (the reciprocal lattice in the case of a crystal). The number of independent basis vectors needed defines the rank of the Fourier module. For a periodic function, this rank is equal to the dimension of the lattice. A parallel net-like arrangement of points is by

definition a lattice, provided the environment about any particular point is in every way the same as about any other point [12]. For a quasiperiodic function, the rank of the Fourier module is larger and it defines the dimension of the (higher dimensional) space, also often called the superspace, in which the quasiperiodic function can be embedded as a periodic function in each of its coordinates. This mathematical tool enables us to use the projection method which first creates five-dimensional superlattice (a completely periodic hypercubic structure) and then projects the points in the latice (according to some geometric constraints) on a two-dimensional slice. The resulting two-dimensional structure is a quasicrystal. An example of the geometric tool and the output of the projection method is shown in Figure 2.1.

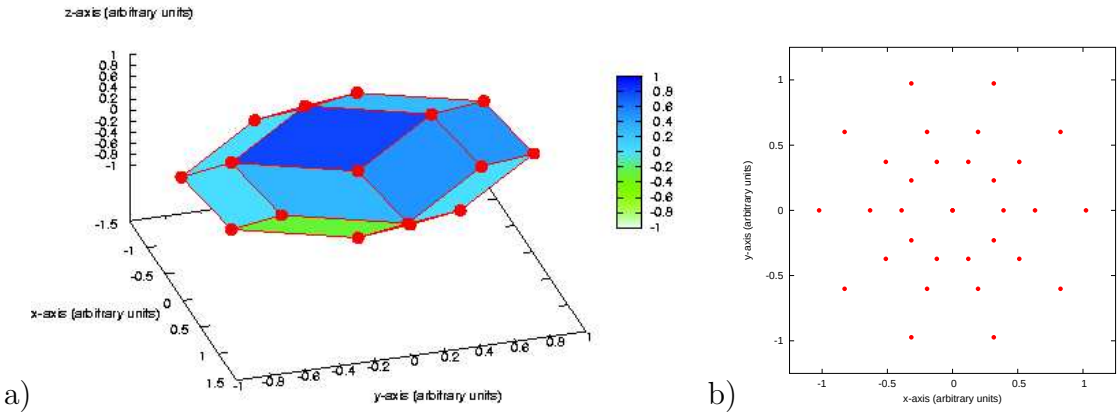

Figure 2.1. The three dimensional representation of the unit cell used to create a periodic crystal in five dimensions (a) and the resultant projection in two dimensions (b) which shows quasiperiodic structure

Figure 2.1 (a) shows the rhombic icosahedron, which is the canonical projection window of the projection from a five dimensional hypercube to a two dimensional surface. The rhombic icosahedron is the three dimensional representation of the unit cell of a periodic lattice in five dimensions. A sample from the final output of the projection is presented in Figure 2.1 (b).

2.1.2. Tiles and Tilings

The question to answer in this section is "How many tiles are needed at least to tile the Euclidean space (primarily two-dimensional) in a purely aperiodic manner

without overlapping tiles or empty spaces between them?". The keywords in this question are "aperiodic" and "Euclidean" since for periodic lattices one unit cell is sufficient and in curved space the answer to this question is completely different. The final answer to this question was given by Roger Penrose in 1974 as "two tiles" which can be seen in Figure 2.2. He used two rhombi with equal side length but different opening angles of 72° ("fat" rhombus) and 36° ("thin" rhombus) respectively, implying five-fold symmetry and hence non-periodic tiling. The resulting tiling is non-periodic, it has long range bond orientational order with five-fold symmetry and it is self similar, i.e. one can produce new tilings by inflation or deflation of the existing tiling (see [11] and [10] for further information). It consists of an infinite repetition of the same local structure in an always slightly different surrounding and it is much more difficult to build up than one would expect. An example of the two dimensional Penrose tiling is shown in Figure 2.3. The periodic stackings of this tiling could be extended to decagonal quasicrystals when the vertices of the the two rhombi are considered as atomic sites as indicated in the figure [13]. The tiling method was extended to two dimensional quasicrystal generation methods such as grid method, inflation and deflation.

Figure 2.2. Two rhombi with equal side length but different opening angles of 72° ("fat" rhombus) and 36° ("thin" rhombus). Arrow decorations encode the matching rules

2.2. One Dimensional Quasicrystals

The model of a one dimensional quasiperiodic lattice is the Fibonacci chain. The quasiperiodic Fibonacci chain can be generated according to several schemes. The usual way to introduce the Fibonacci sequence employs an inflation-like procedure, a substitution rule, starting from an initial segment, say "0". After defining another kind of segment "1", one generates a series of new chains using the following substitution

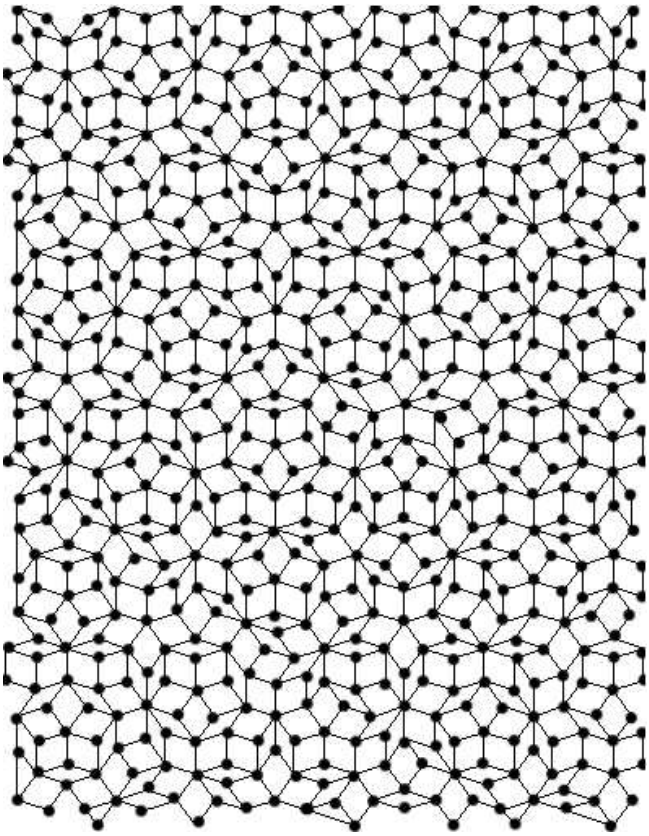

Figure 2.3. An example of a rhombic Penrose tiling formed by appropriate matching of fat and thin rhombi

rules: $0 \rightarrow 1$ and $1 \rightarrow 1+0$. Each 0 will be replaced by a 1, and each 1 will be replaced by a $1+0$. First few members of the Fibonacci chain are

$$
\begin{aligned}
&0\\
&1\\
&1{+}0\\
&1{+}0{+}1\\
&1{+}0{+}1{+}1{+}0\\
&1{+}0{+}1{+}1{+}0{+}1{+}0{+}1\\
&1{+}0{+}1{+}1{+}0{+}1{+}0{+}1{+}1{+}0{+}1{+}1{+}0\\
&1{+}0{+}1{+}1{+}0{+}1{+}0{+}1{+}1{+}0{+}1{+}1{+}0{+}1{+}0{+}1{+}1{+}0{+}1{+}0{+}1\\
&.\\
&.\\
&.
\end{aligned}
\tag{2.1}
$$

If f_n denotes the sum in the nth row in Eqn. 2.1, one obtains Fibonacci numbers f_n as

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ..., \tag{2.2}$$

defined by the recursive relation

$$f_{n+1} = f_n + f_{n-1}, \tag{2.3}$$

with initial conditions $f_0 = 0$ and $f_1 = 1$. The ratio of successive Fibonacci numbers converges to the golden ratio as the chain grows to infinite length

$$\tau = \lim_{n\to\infty} \frac{f_{n+1}}{f_n} = \frac{1+\sqrt{5}}{2}. \tag{2.4}$$

This can be proved in another way: What we are concerned with in the Fibonacci chain is the number of 0s and 1s. The substitution rule given above is a linear transformation with matrix

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \tag{2.5}$$

where the initial population is

$$m_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{2.6}$$

representing the number of 0s in the first row and the number of 1s in the second row. Therefore, in the first generation we have

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{2.7}$$

and in the second we have

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tag{2.8}$$

representing one 0 and one 1 in the chain. And as a final check if we apply M^6 to m_0 we get

$$M^6 m_0 = \begin{pmatrix} 5 & 8 \\ 8 & 13 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 5 \\ 8 \end{pmatrix}, \tag{2.9}$$

which is the correspondent matrix of the seventh row in Eqn. 2.1, five 0s and eight 1s adding up to the sixth Fibonacci number: 8. The recursion relation in Eqn. 2.3 can be written in matrix form as

$$m_n = M^n m_0 \quad n = 1, 2, 3, ... \tag{2.10}$$

The characteristic equation of M matrix is

$$\begin{vmatrix} -\lambda & 1 \\ 1 & 1-\lambda \end{vmatrix} = 0 \Longrightarrow \lambda^2 - \lambda - 1 = 0. \tag{2.11}$$

The solutions to Eqn. 2.11 which are the eigenvalues of M are τ and $-1/\tau$. Since τ is an irrational number, we have proved that the Fibonacci sequence is aperiodic.

Some useful identities that can be derived from Eqn. 2.11 are

- $\tau^2 = \tau + 1$
- $\tau - \tau^{-1} = 1$
- $\tau^3 - 2\tau^2 = -1$.

2.3. Higher Dimensional Quasicrystals

The construction of two dimensional quasicrystals can briefly be described as follows:

- Substitution: A method similar to its analogue in one dimensional case. But this time, a geometrical large scale tiling is substituted by its self similar prototiles. Figure 2.4 shows an example of producing "chair tiling" by substitution [10].

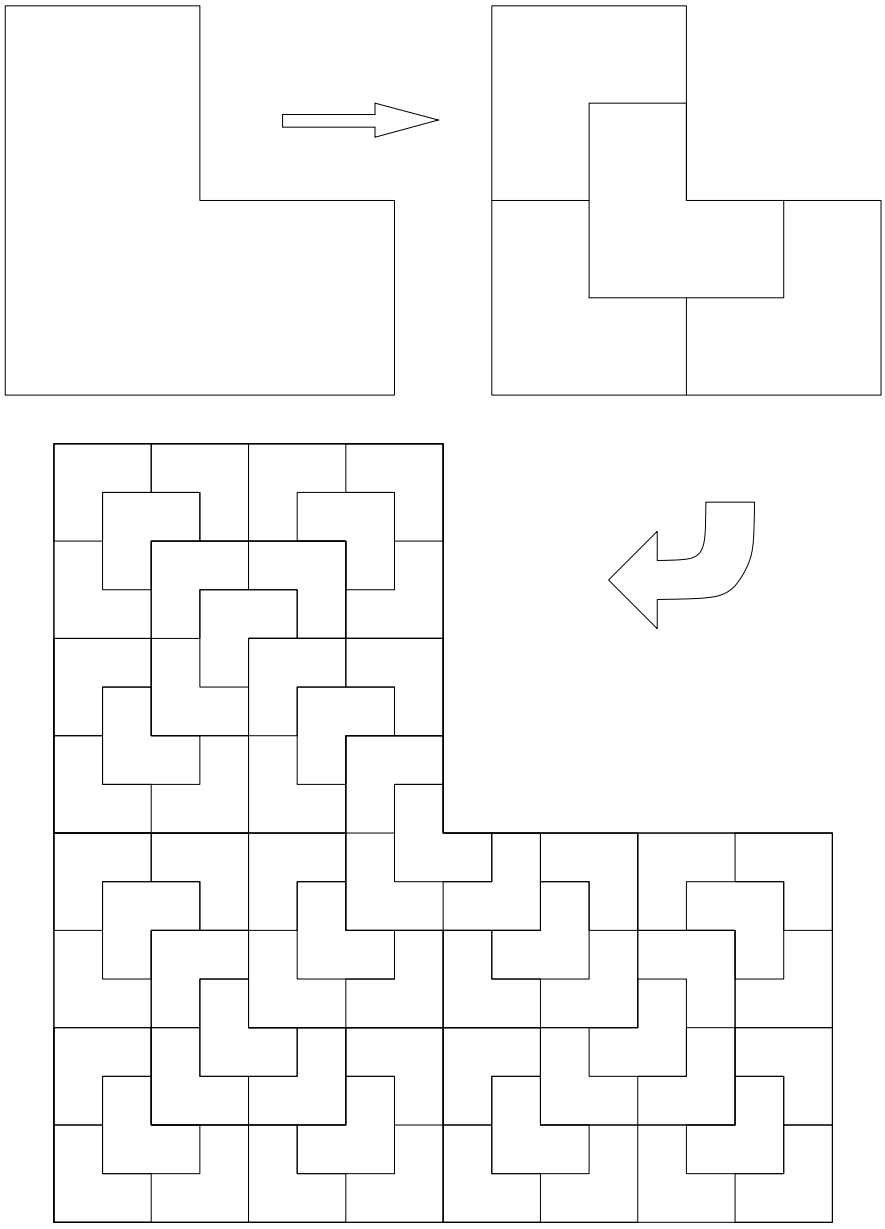

Figure 2.4. The chair is substituted by four self similar chairs. These four prototiles are congruent but differ in orientation

- Multigrid Method: The two dimensional space is decorated by families of equally spaced parallel lines. Each family is called a grid and the configuration is called a multigrid. After following certain algebraic steps, an orthogonal dual of the multigrid is constructed. If five families of grid lines are used in construction of the orthogonal dual one gets the Penrose rhomb tiling (see Figure 2.3).
- Projection: This is a powerful technique for constructing non-periodic patterns and tilings. In this method, a subset of a point lattice is projected onto a plane. As mentioned earlier, the projection of a five dimensional hypercubic lattice onto a plane results in two dimensional Penrose tiling (see Figure 2.3).

The next chapter consists of a detailed description of a three dimensional quasiperiodic structure which is actually the atomic coordinates of an experimental sample of a quasicrystalline surface. The structural and symmetry properties are discussed in detail and it will serve as the substrate layer for the simulations.

3. SIMULATIONS AND PARAMETERS

The computational tool that will be used to treat the problem of structure formation on a quasicrystalline substrate is molecular dynamics (MD) simulations with simulated annealing. The necessary information about MD simulations and simulated annealing can be found in Appendix A. The simulation parameters are explained in detail in the following sections.

3.1. Quasicrystalline Substrate

The quasicrystalline subsrate layer for the simulations was constructed from the atomic coordinates given by Saitoh *et al* [2]. It consists of two layers of atoms that are 0.5 Å apart. The complete real space picture of the quasicrystalline substrate is presented in Figure 3.1. A 50 Å by 50 Å unit cell is cut from this atomic configuration to serve as the unit cell for the simulations. The real space coordinates of the two layers of the unit cell together with their diffraction patterns are presented in Figure 3.2. The surface layer consists of 610 atoms giving a density of 0.244 atoms/Å^2 and layer two has 594 atoms giving a density of 0.238 atoms/Å^2. Minimum interatomic distance in layer one is 1.506 Å and for layer two is 1.503 Å.

The diffraction patterns exhibit the 10-fold symmetric quasicrystalline structure. The overall real space picture of the substrate layer that will be used for the simulations -namely the unit cell- is shown in Figure 3.2 (e) together with its diffraction pattern (f). The two layers may or may not have atoms on identical coordinates and the compensation of the two layers for each other to form a 10-fold symmetric structure should be noted. The diffraction pattern calculated for the two layers together has more clear view. This diffraction pattern is the same as for a two dimensional Penrose tiling [10].

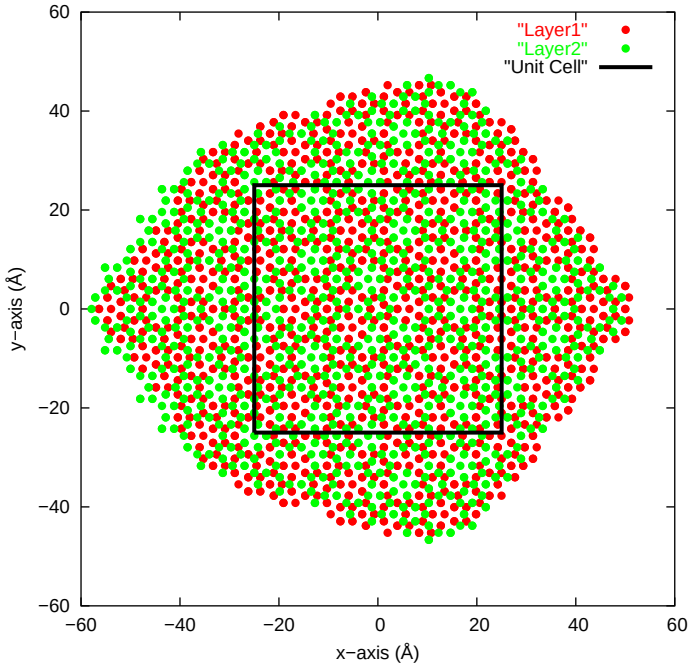

Figure 3.1. Plot of the original substrate layers. Layer 1 is on the surface of the specimen and layer 2 is 0.5 Å deeper. The unit cell cut from this configuration is denoted by the solid square

3.2. Particle Dynamics

The simulations are carried out in a two dimensional unit cell of the same size as the substrate layers (50 Å by 50 Å) at an elevation h above the substrate surface layer. The particles (called atoms interchangeably) are considered to be spherical particles with their centers on the surface of the unit cell. The interactions occur between pairs of atoms. The role of these interactions is to have a repulsive character for small interatomic distances and to have an attractive character at large interatomic distances. We have used the Lennard-Jones (LJ) potential which is given in Equation 3.1 for a pair of atoms i and j located at $\vec{r_i}$ and $\vec{r_j}$.

$$V\left(r_{ij}\right) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{3.1}$$

where $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ and $r_{ij} = |\vec{r}_i - \vec{r}_j|$. The parameter ϵ is related to the strength of the interaction and σ is a parameter related to the range of the interaction. The

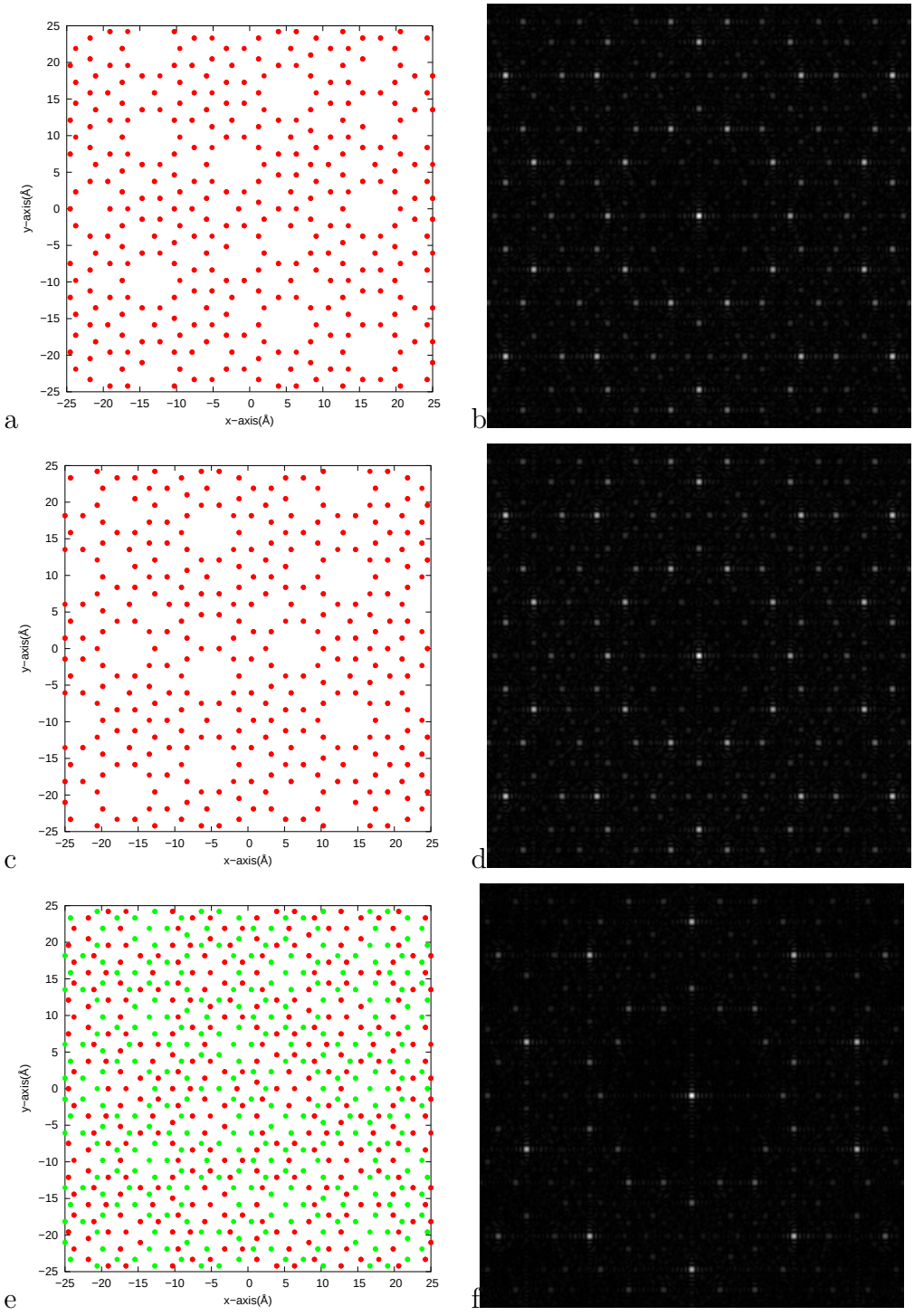

Figure 3.2. Unit cell cut from the original configuration to be used for simulations: The surface layer (a) and its diffraction pattern (b); the second monolayer that is 0.5 Å deeper from the surface (c) and its diffraction pattern (d); the two layers plotted together (e) (red denotes the surface layer and green denotes the deeper layer) and their diffraction pattern (f)

interaction calculations are performed for each pair of atoms in the unit cell. The force between each pair can then be calculated using $\vec{F} = -\vec{\nabla}V(r)$ as

$$\vec{F}_{ij} = \left(\frac{24\epsilon}{r_{ij}}\right)\left[2\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]\hat{r}_{ij} \tag{3.2}$$

where $\hat{r}_{ij} = \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|}$ which is the unit vector on the line joining the atoms i and j and directed towards atom i. Using Newton's second law $\vec{F}_i = m_i\ddot{\vec{r}}_i$ to get the equations of motion for each atom i, we have

$$\vec{F}_i = \sum_{\substack{j=1 \\ j\neq i}}^{N_a} \vec{F}_{ij} + \sum_{\alpha=1}^{N_s} \vec{F}_{i\alpha} = m_i\ddot{\vec{r}}_i \tag{3.3}$$

where N_a is the number of atoms in the adsorbate layer, N_s is the number of atoms in the substrate layers and m_i is the mass of atom i.

Therefore, the net acceleration of atom i can be expressed as

$$\begin{aligned}\ddot{\vec{r}}_i = &\sum_{\substack{j=1 \\ j\neq i}}^{N_a}\left(\frac{24\epsilon_a}{m_i r_{ij}}\right)\left[2\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]\hat{r}_{ij} \\ &+\sum_{\alpha=1}^{N_s}\left(\frac{24\epsilon_s}{m_i r_{i\alpha}}\right)\left[2\left(\frac{\sigma}{r_{i\alpha}}\right)^{12} - \left(\frac{\sigma}{r_{i\alpha}}\right)^{6}\right]\hat{r}_{i\alpha}\end{aligned} \tag{3.4}$$

where the first sum represents the interaction between the adsorbate atoms and the second sum represents the interaction between the adsorbate and substrate atoms. ϵ_a and ϵ_s are the strengths of the adsorbate-adsorbate and adsorbate-substrate interactions respectively. $r_{i\alpha}$ is the interatomic seperation between the adsorbate atom i and the substrate atom α and $\hat{r}_{i\alpha}$ is the unit vector from the substrate atom α to the adsorbate atom i. Equation 3.4 is the differential equation to be integrated numerically (details can be found in Appendix A.2.2.3).

Now, we have to set some MD units in order to make the simulations more efficient, but still keeping the results interpretable by physical reasoning. The scaling

scalars could best be arranged such that they do not interfere with the output directly and could easily be replaced by their real observable value. As can be seen from Equation 3.4 the mass of the atom m_i is the same for all the adsorbate atoms and occurs as a multiplicative constant for the differential equation to be solved. Since we are interested in the final configuration of the adsorbate layer at the end of a single run of the simulation, the mass of each atom could be taken to be unity.

The basic measurements during a run of the simulations is the mean kinetic, potential and total energy of the system. The mean potential energy of the dynamic adsorbate layer is calculated as

$$V_M = \frac{1}{N_a} \sum_{i}^{N_a} \left\{ \sum_{\substack{j=1 \\ j \neq i}}^{N_a} 4\epsilon_a \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] + \sum_{\alpha=1}^{N_s} 4\epsilon_s \left[\left(\frac{\sigma}{r_{i\alpha}} \right)^{12} - \left(\frac{\sigma}{r_{i\alpha}} \right)^{6} \right] \right\}. \quad (3.5)$$

The mean kinetic energy of the adsorbate atoms is

$$K_M = \frac{1}{N_a} \sum_{i=1}^{N_a} \frac{v_i^2}{2} \quad (3.6)$$

where v_i is the speed of the atom i. And the mean total energy of the adsorbate layer is

$$E_M = V_M + K_M. \quad (3.7)$$

We are using MD simulations with simulated annealing to let the system freeze gradually and the adsorbate layer get ordered with the most probable structural configuration. The crucial parameter in these calculations is the measurement and control of the temperature of the system. We know from kinetic theory that each translational degree of freedom contributes $\frac{kT}{2}$ to the average kinetic energy where k is Boltzman's constant. Since our system has two degrees of freedom, the average kinetic energy of

our system will be kT. Setting $k = 1$ in our simulations results in

$$T = K_M = \frac{1}{N_a} \sum_{i=1}^{N_a} \frac{v_i^2}{2} \tag{3.8}$$

where we have used Equation 3.5. Therefore, this result completes the picture: Reducing the temperature gradually, is obtained by gradually decreasing the kinetic energy of the system, namely reducing the speed of each atom regularly during the run of the simulation.

4. COMPUTATIONAL ANALYSIS AND RESULTS

4.1. Adsorbate Layer Parameters

As a first attempt to treat the problem of structure formation on a quasicrystalline substrate, we tried to form an adsorbate structure that is similar in density to the substrate layers in order that the adsorbate-adsorbate and the adsorbate-substrate interactions are comparable. This would let us change and test other parameters that may effect the system behavior and structure.

As presented in Section 3.1, the atomic densities of the first and second substrate layers are 0.244 atoms/Å^2 and 0.2376 atoms/Å^2 respectively where the first layer represents the surface layer. In order for the adsorbate layer to have a relevant density, the number of atoms in the adsorbate layer was set to 600 giving an atomic density of 0.24 atoms/Å^2 which is almost midway between the densities of the two substrate layers.

The problem now turns out to find the necessary conditions for this density value to be applicable to our simulations. The parameters ϵ and σ in Equation 3.1 are to be determined. Since ϵ is a parameter related to the strength of the interactions, we set it equal to $0.25eV$ for reasons of simplicity in calculations. In order to find the value for σ to fill the unit cell with the least amount of defects possible, structure factor calculations have been made. The range of the interactions should be arranged such that the structure formed fills the unit cell completely and as regularly as possible. With constant initial conditions, the simulation -with cooling on- is run for various σ, and the final atomic orientation is investigated by Fourier analysis (see Appendix C for details). The value of σ is selected as 3.81Åfor which the adsorbate layer fills the unit cell as a regular hexagonal structure with the least amount of defects possible. Therefore, this value of σ is used in the investigations of the relative orientations of simulation outputs since any deviation in this hexagonal structure is a direct consequence of the effects of the substrate layers.

4.2. Substrate Layer

The substrate layer properties are mentioned in detail in Section 3.1. When LJ potential is used as the interatomic interaction, the potential of the substrate layer is calculated as presented in Figure 4.1.

Figure 4.1 indicate that many five-fold symmetric contour lines of the substrate layer potential. The substrate atoms in these regions exhibit five-fold or ten-fold symmetric circular structures. At the very center of these constant potential "roses" a single substrate atom rest indicating a center of symmetry. Minimum energy contours carry the labels -8 and -7 in Figure 4.1 (a). The second minimum potential contours are observed to exhibit interesting five-fold and ten-fold symmetric decorations in the plot. The contour plot is obtained by the code presented in Appendix B.

Figure 4.2 shows the overall potential pattern of the substrate layer. The fact that the global minima are on the two fold symmetry sites is remarkable.

4.3. Numerical Implementation

The numerical calculations are carried out in a square unit cell corresponding to a 50×50 Å^2 substrate area with periodic boundary conditions. The top and bottom layers of the substrate contain 610 and 594 atoms, respectively. It is assumed that the artificial periodicity induced by the boundary conditions is large enough to be negligible with respect to the smaller length scales of structures in the quasi-crystalline substrate. ϵ is set to 0.25 eV for practical purposes. Note that in the absence of a substrate, the adatoms cannot form a fully ordered, i. e., defect-free, triangular lattice, since this is incommensurate with the square unit cell chosen. The unit cell thus prevents the adatom lattice from locking into an orientation consistent with itself. For our purposes this is desirable, since we are interested in the possible formation of local domain structures with orientations that should not be favored by boundary effects.

Calculations with two sets of parameters are performed: Set (i) has 600 adatoms

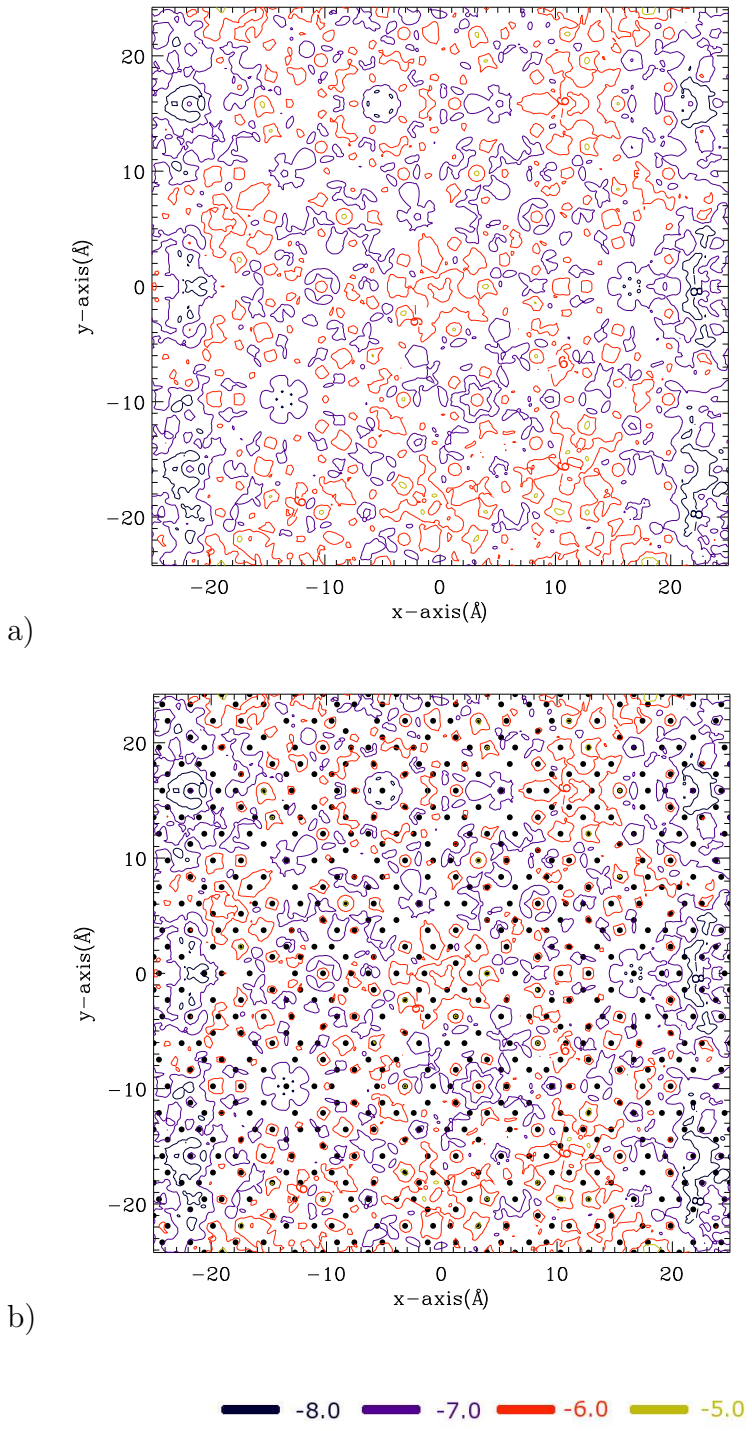

Figure 4.1. Contour plot of the potential due to the substrate layer (a) and the simultaneous plot of the potential contours and the atomic coordinates of the substrate layer atoms

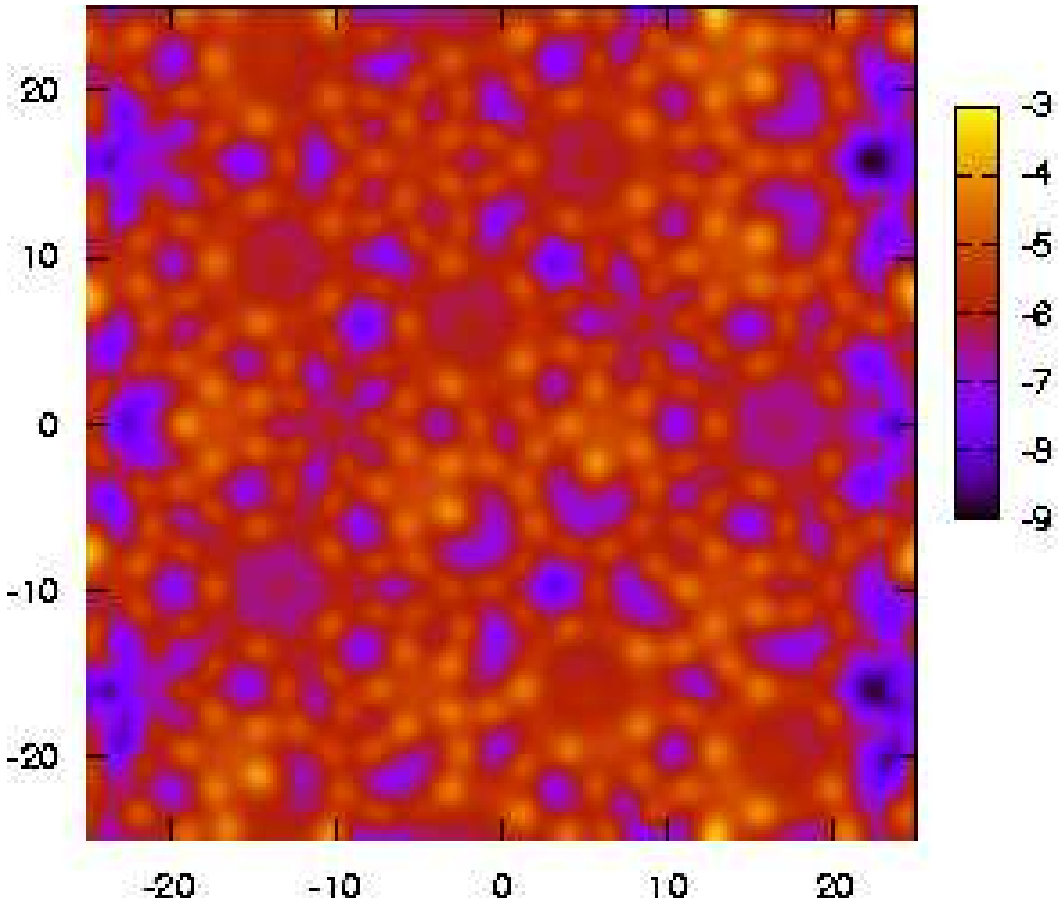

Figure 4.2. Overall potential of the substrate layer

with $\sigma = 3.81$ Å and $h = 3.5$ Å, corresponding to $\bar{r}/\sigma = 0.54$, where $\bar{r}$ is the average adatom-adatom distance. Set (ii) has 1600 adatoms with $\sigma = 2.55$ Å and $h = 2.34$ Å, so that $\bar{r}/\sigma = 0.49$ (σ is the Al-Al interaction hard-core radius and h is the bulk interplanar distance of Al(111) that were used in [1]). Thus for set (ii) the average force between adatoms is roughly 3.25 times stronger than for set (i). In set (i), the adatom layer has a density comparable to any one of the substrate layers and σ is estimated such that this layer can fill the unit cell with the least amount of defects possible when simulations in the absence of the substrate layers are considered. We used set (i) in order to obtain single domain structures and set (ii) for multiple domain structures as simulation outputs.

The simulations are carried out as follows: For each run the adatoms are placed at random, such that two adatoms are not closer to each other than 0.37σ. This condition

is imposed in order to avoid unnecessarily large repulsive forces during the initial phase of the simulations. It is verified that this constraint did not alter the simulation results. The initial velocities of the adatoms were chosen from a Gaussian distribution and the equations of motion, Eq. (3.4), were integrated using a Verlet algorithm with fixed time step. Annealing was simulated by reducing the velocities of all adsorbate atoms by 1% every 120 time steps. The time step was chosen such that during evolution without cooling the total energy of the system remained constant within a precision of $10^{-5}\bar{E}$, where $\bar{E}$ is the average energy. (Details of the implementation are presented in Appendix A.)

4.4. Simulation Results

The final configurations for the simulations using set (i) have hexagonal structures with a single rotational alignment and contain dislocations. Different final atomic configurations of the adsorbate layer are found for different initial conditions. However, all observed configurations are related to each other. For the purpose of comparison, some of these configurations are plotted with identical scaling and the basic differences and similarities are studied. In the following, we present the effects on the final structures resulting from the underlying quasicrystalline layers in two groups.

4.4.1. Translational Effect

Some configurations have the same rotational alignment but are translated relative to each other. An example is shown in Fig. 4.3.

The displacement applied to the configuration shown in Fig. 4.3b in order to obtain a minimum mismatch with the configuration displayed in Fig. 4.3a is 2.29 Å. The significance of this numerical value is that it is $\tau\sqrt{2}$ where τ is the golden mean implying that the displacement vector between the two configurations is $\vec{\Delta} = \tau\,(1, -1)$. We observe two more vectors of the same length that make 72° and 144° with $\vec{\Delta}$ in counterclockwise direction. It is also remarkable that the diffraction pattern in Fig. 4.3d for the mixed configuration presented in Fig. 4.3c has a hexagonal structure

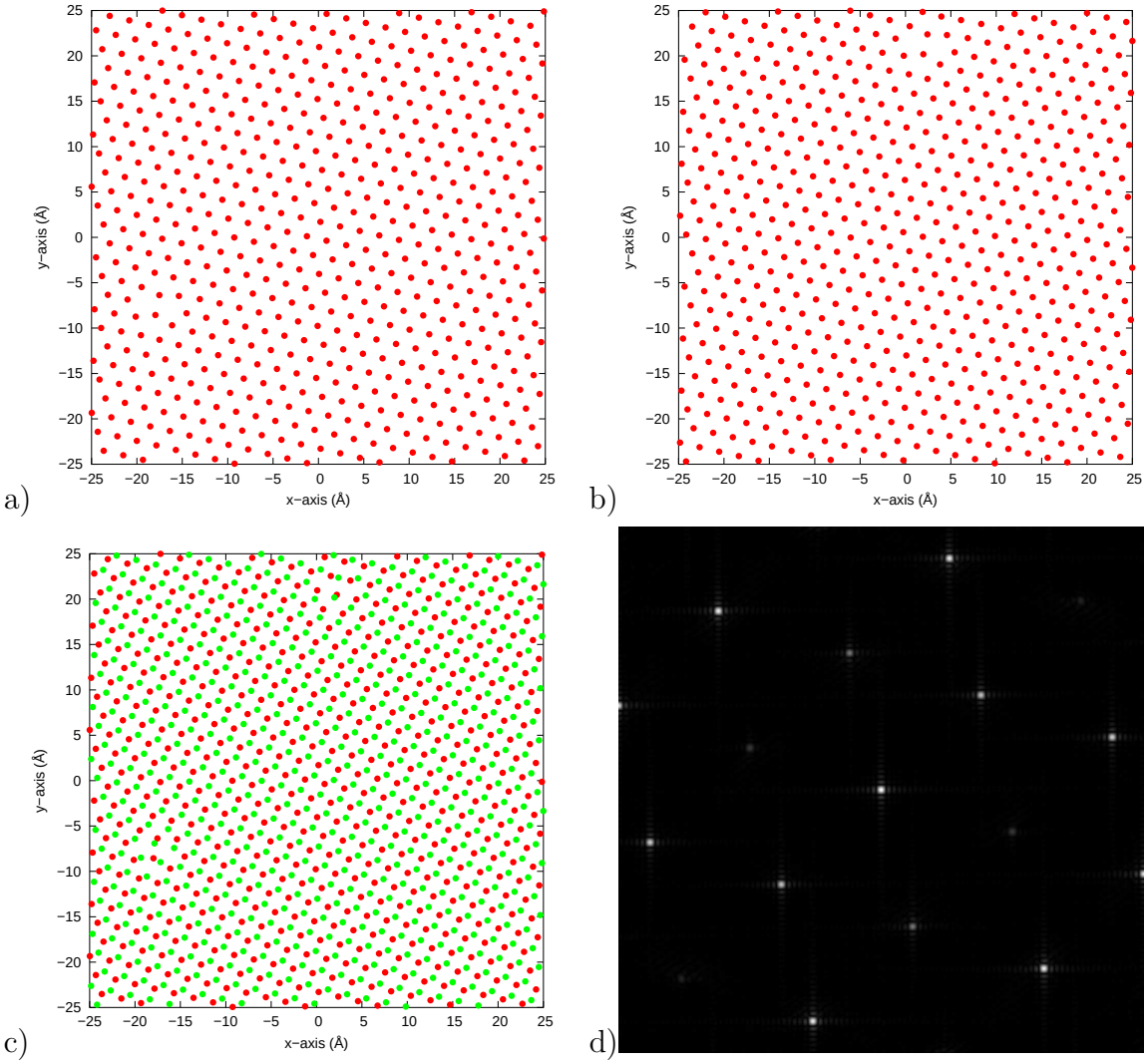

Figure 4.3. Two simulation outputs [(a) and (b)], their superposition exhibiting the translational effect (c), and the corresponding diffraction pattern (d)

with unequal intensities of the diffraction spots.

4.4.2. Rotational Effects

Other common effects that are observed in the final output configurations of the adsorbate layer are the rotational effects. Frequently, the final configurations have almost the same characteristics except that one is rotated by either 12° or 2.5° with respect to the other. An example is shown in Fig. 4.4. If we combine the two configurations shown in Figs. 4.4a and 4.4b, we obtain Fig. 4.4c. The diffraction pattern concerning the combined configuration shows two hexagonal structures that are rotated

by 12° with respect to each other.

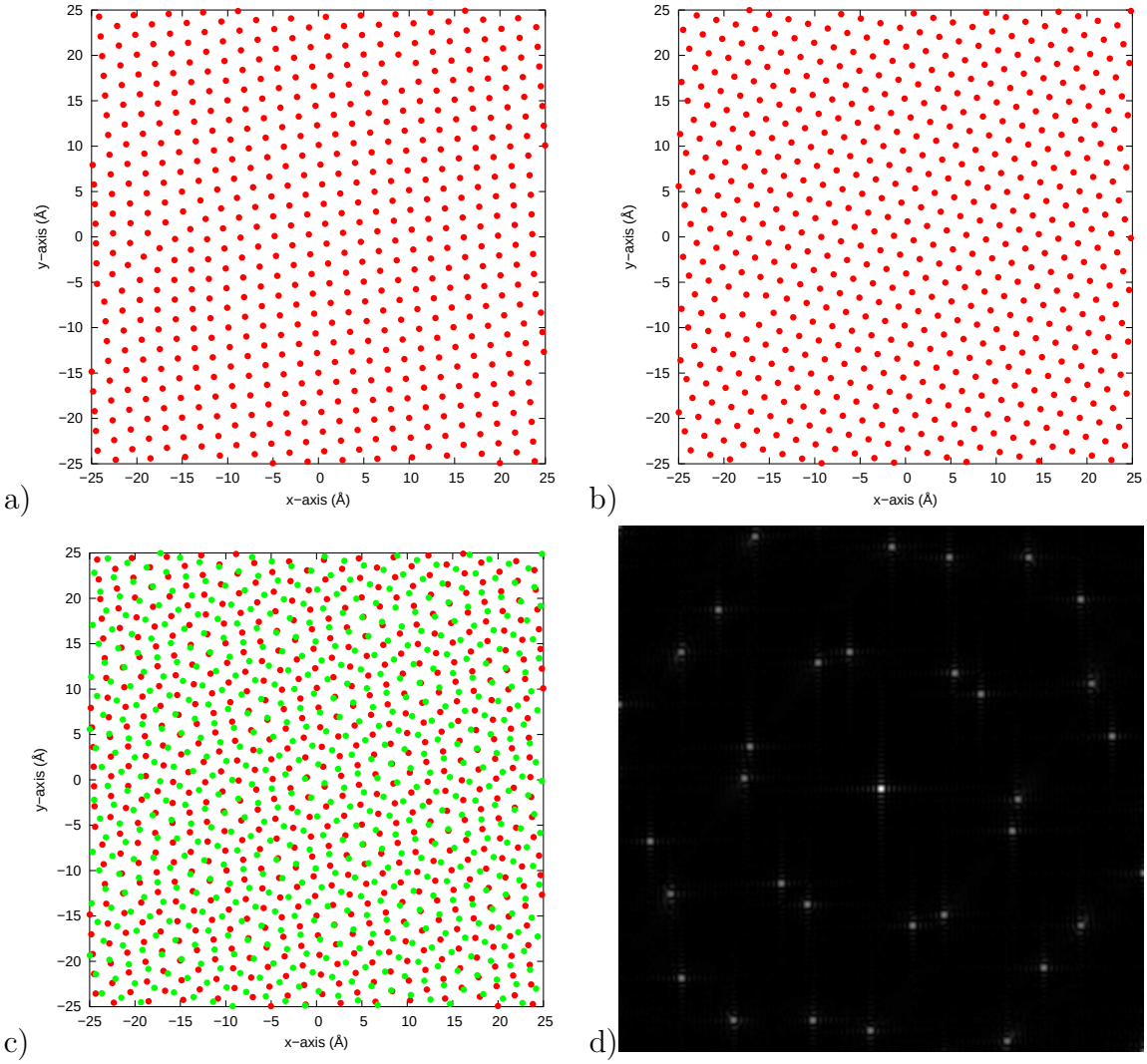

Figure 4.4. Two simulation outputs [(a) and (b)], their superposition exhibiting the rotational effect by 12° (c), and the corresponding diffraction pattern (d)

Some of the output configurations are related to each other so that they are pinned at some point which acts as a vertex around which the entire structure seems to be rotated. An example is shown in Fig. 4.5. The two configurations of Figs. 4.5a and 4.5b are pinned at a special location where they share a common hexagonal structure that can be identified in Fig. 4.5c, where the superposition of these two configurations are plotted. The diffraction pattern of this combined configuration is shown in Fig. 4.5d, which contains two hexagonal structures rotated by 2.5°.

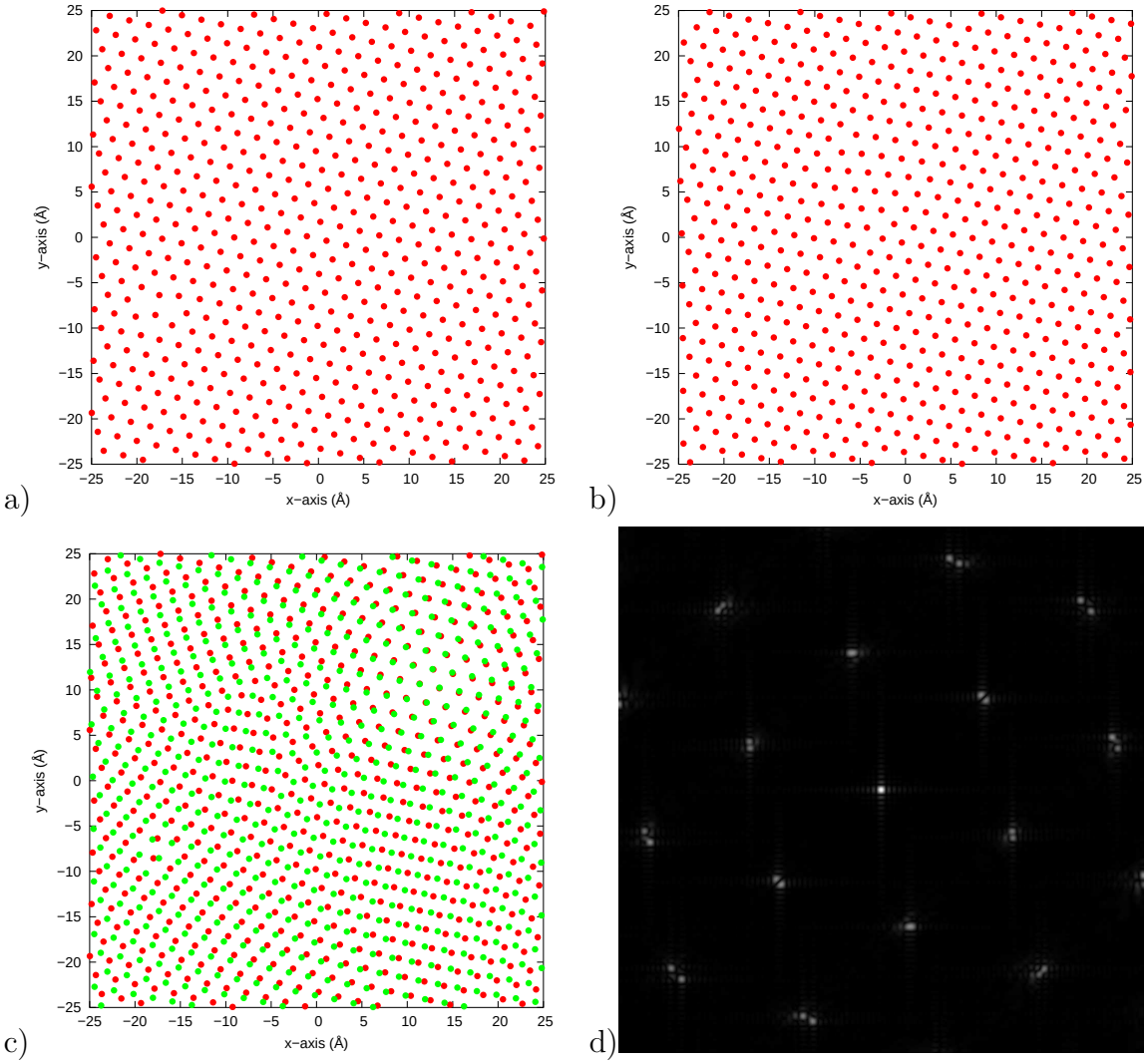

Figure 4.5. Two simulation outputs [(a) and (b)], their superposition exhibiting the rotational effect by 2.5° (c), and the corresponding diffraction pattern (d)

4.4.3. Multiple Domain Results

We used the calculation set (ii) to study the dependence of the results on the simulation parameters and the domain formation during cooling. The final configurations are not single hexagonal structures. They are composed of several hexagonal domains aligned in different orientations. In order to investigate the domain structure of the final configuration, the code in Appendix D written in IDL is used. An example plot of this kind is shown in Figure 4.6.

As can be seen from the figure, the structure contains four different domains

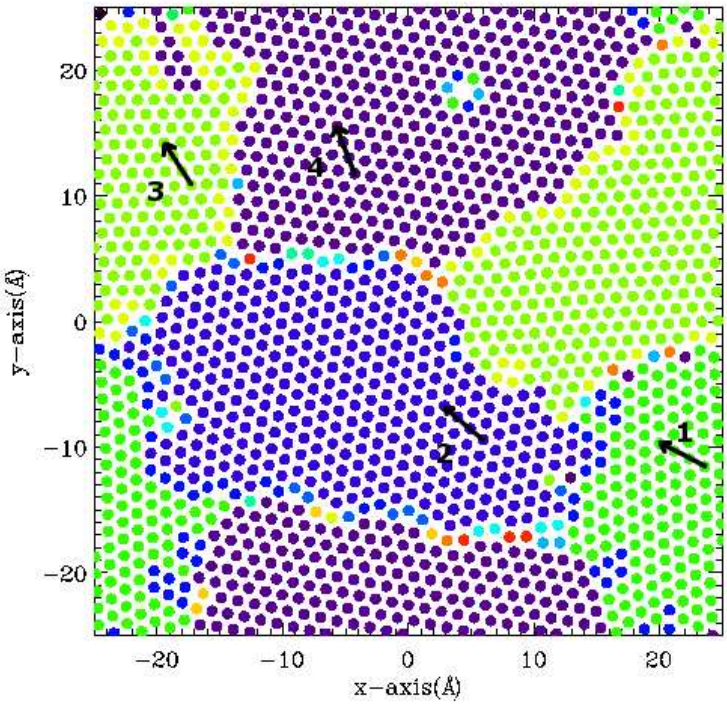

Figure 4.6. The sample domain structure plot for a simulation output

indicated with different colors and numbers 1 through 4. The domain boundaries and the defects are also indicated with varying colors. We observe at most four domains in a single configuration with domain sizes of approximately 25 Å. Also in this case, the angular mismatch at the domain boundaries is either 12° or 2.5°. We find more defects in the final configurations compared to the calculations using set (i).

5. CONCLUSION AND DISCUSSION

At the interface between a quasicrystal and an ordinary crystal, the structural misfit leads to the formation of self-size-selected, nano-scale crystalline domains. The aperiodic substrate further acts as a structural template to align these domains along quasicrystallographic directions. In order to mimic these observations numerically with molecular-dynamics simulations, we have introduced foreign atoms on a model quasicrystalline surface. We have observed that different surface configurations emerged resulting from the misfit between a quasicrystal and a crystal.

As in a Wigner crystal, it is the repulsive interaction between the adatoms which forces them to assemble in a crystalline matrix with sixfold symmetry [16]. This symmetry is not reinforced by the chosen square unit cell. Then, any small deviation from this configuration is due to the quasicrystalline substrate. We have observed different configurations in the resulting surface adlayer which we can treat in two different groups.

In the first group, we observe a translational effect between sets of final configurations. The magnitude of the displacements are related to the golden mean. We have found three vectors corresponding to the observed displacements belonging to the quasicrystalline group, i. e., vectors that are 72° apart. The vectors are shifted by 45° relative to the symmetry lines of the quasicrystalline substrate, carrying the signature of the aperiodic substrate. When we inspect the potential figure of the quasicrystalline surface at the elevation where the simulations are performed (see Fig. 4.2), we observe that the absolute minima locations occur at twofold symmetry sites which is in complete accordance with the energy calculations in [1]. In order to investigate the effect of periodic boundary conditions that could interfere with the potential calculations, the periodic boundary conditions are removed and the surface potential is re-calculated for the simulation unit cell but this time using the complete layers of the substrate (see Fig. 3.1). The basic difference between the two figures is the shrinking of the near-boundary absolute potential minima regions. However, the relative locations of these

potential wells are preserved. We also note that the observed translation directions are the relative twofold symmetry axes for these potential minima. For the parameters presently used, other two complementary pentagonal directions could not be observed. We encounter the displacement amount as a common interatomic distance in the final adsorbate configurations. We conjecture that hexagonal adatom lattices and the boundary conditions play a role in selecting some preferable displacement directions. Figure 5.1 demonstrates the translational effect together with the surface potential due to the substrate layer. The figure shows the absolute potential minima at the two-fold symmetry sites with squares. It is also remarkable that the orientation of one edge of the small pentagons indicated with a dark circular region on the potential pattern frequently coincides with the displacement direction. Since these pentagons are formed by relatively high potential regions at their vertices, we conjecture that the translational effect occurs along the two-fold symmetry directions of the absolute potential minima in a way that the translated structures are relatively symmetric with respect to local energy maxima. The displacement amount also corresponds to the center-to-vertex distance of these pentagons. It is also notable that these pentagons are on two-fold symmetric sites.

In the second group, we identify pairs of configurations which represent just two specific rotations by 12° and 2.5°. In actual crystal growth on a quasicrystalline substrate, the adatoms are first attracted to energetically favorable locations of the substrate which serve as a seed for further growth. In the calculations, we observe vortex formations around some singular points. We identify these points as energetically favorable sites on the quasicrystalline surface. The separation of these sites around which the crystalline regions are assembled, determines the size of the final domains. In the experiment, the domain size was estimated as 3 nm [1]. The computations suggest domains of similar dimensions. It is interesting to note that the azimuthal displacements in the domain orientations were also experimentally observed [1]. Figure 5.2 shows the superposed structures that exhibit rotational effects together with the potential pattern. In the figures, the black dots indicate local potential minima which are the energetically favorable locations for domain formation. The green dot in Fig. 5.2a indicates a local potential maximum although one would expect it to be a local

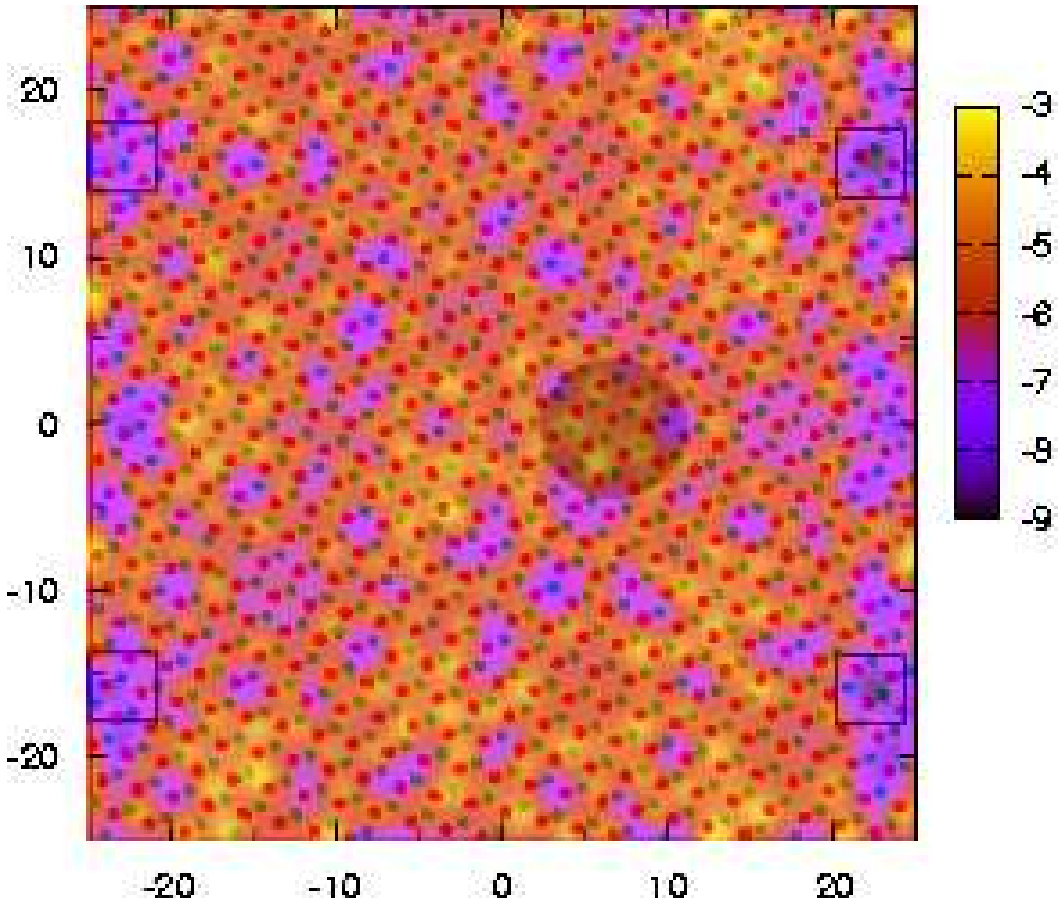

Figure 5.1. Superposition of the atomic configurations exhibiting translational effect (see Fig. 4.3c) plotted on the surface potential due to the substrate layer

minimum since it is a popular atomic site. However, this is not surprising since the adsorbate and the substrate structures are incommensurate and the simulations take place in a finite size.

During growth of a metal on a quasicrystalline substrate, all these effects play a role and the final adlayer carries their traces. In order to realize this situation, we use one of the final configurations of the calculations with set (i) to construct a combined configuration. We first add a 2.5°-rotated twin to the initial configuration. Then we repeatedly rotate this structure by 12° and superimpose each one of the configurations to a combined result. The diffraction pattern for this construct is presented in Fig. 5.3 which shows two sets of diffraction spots separated by 2.5° and azimuthally repeats every 12°. This pattern is consistent with experimental findings based on LEED [1].

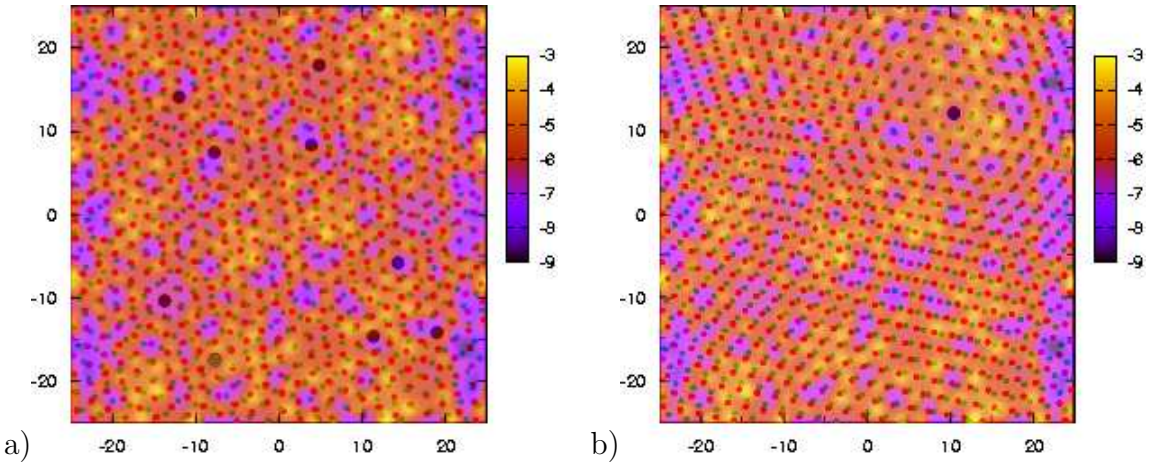

Figure 5.2. Superposition of the atomic configurations exhibiting rotational effect by 12° (see Fig. 4.4c) and by 2.5° (see Fig. 4.5c) plotted on the surface potential due to the substrate layer

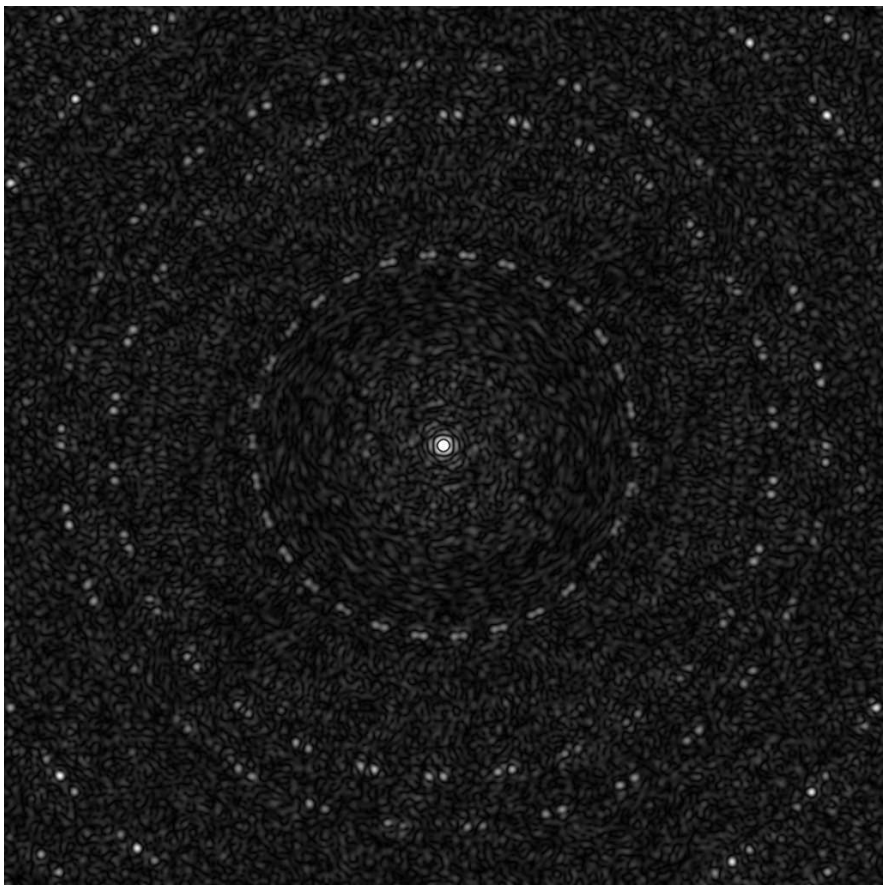

Figure 5.3. The diffraction pattern of a combined structure

Metal atom adsorbate-quasicrystal interactions have shown effects of intermixing and alloying [17, 18]. The computations presented here naively neglects real interac-

tions and assumes a hard-sphere Lennard-Jones type environment for all the involved species. Nevertheless, some salient features of the quasicrystalline order is detected in the formation of the adlayer.

APPENDIX A: MD SIMULATIONS WITH SIMULATED ANNEALING

A.1. Definition

MD simulations is an efficient, reliable and flexible method of visualizing the dynamics of small and medium scale atomic and molecular structures. It is based on the calculation of interatomic forces for all the constituents and application of well known dynamical rules in order to evaluate the position and velocity of each particle at later times. Therefore the primary parameter in the performance and the efficiency of MD simulations is the size of the system simulated, which is characterized by the total number of atomic particles for which the interatomic forces are calculated.

MD simulations are ideal for keeping track of every single particle from the beginning to the end of the simulations. At any instant, the properties of the system can be monitored and can be altered on demand. Furthermore, the whole system can be turned into a real-time simulation environment giving an opportunity to visualize any significant observable phenomena.

One of the most important parameters in MD simulations is temperature since the kinetic energy of the particles is related to temperature statistically. The problem we are concerned with is a structure formation phenomena. The physical importance of the problem resides in the fact that the natural way this occurs is the cooling of the molten material on the quasicrystalline substrate while the atoms are moving towards the locations that they will be most stable having the minimum energy possible. In order to simulate this phenomenon, our MD simulations employ simulated annealing. The simulated annealing algorithm was derived from statistical mechanics. Kirkpatrick et. al. [19] proposed an algorithm which is based on the analogy between the annealing of solids and the problem of solving combinatorial optimization problems.

Annealing is the physical process of heating up a solid and then cooling it down

slowly until it crystallizes. The atoms in the material have high energies at high temperatures and have more freedom to arrange themselves. As the temperature is reduced, the atomic energies decrease. A crystal with regular structure is obtained at the state where the system has minimum energy. Simulated annealing is motivated by an analogy to annealing in solids.

The algorithm in this thesis simulates the cooling of a molten periodic film on a quasiperiodic substrate. The simulation is started in a highly energetic state where collisions between particles is frequent and rapid, and then by lowering the velocities of the atoms, the system is cooled down and the system is again set free in order to get a new equilibrium where the particles move to energetically favorable locations on the substrate surface. The final structure is the frozen adsorbate layer with atoms being on the energetically favorable locations.

A.2. Simulation Flow

A.2.1. Preliminary Analysis

A.2.1.1. Boundary Conditions. The system simulated acts in a finite region which probably has rigid boundaries where the atoms collide during interactions. However, in systems of macroscopic size, only a very small fraction of atoms is close enough to a wall to experience these collisions. Thus, the simulation will fail to capture the typical state of an interior atom. Since our goal is not the study of behavior near real walls, walls are eliminated. Periodic boundary conditions are used instead. Periodic boundary conditions are formed by filling the space with identical unit cells of the simulation in the form of infinite-space filling arrays. The effect is as if the simulation unit cell is surrounded by eight more unit cells on the circumference. The consequences of the periodicity mentioned are:

- An atom that leaves the simulation region from one bounding face immediately enters the region from the opposite face,
- An atom that is close enough to a boundary will interact with atoms in an

adjacent copy of the system, namely with atoms near the opposite boundary.

The second consequence of the periodic boundary conditions is known as the wraparound effect in literature and the simulation region topologically but not spatially resembles a torus [14].

The first item above should be taken into consideration after each integration of the equations of motion. If we denote the size of the square unit cell as L and let x and y coordinates be defined to lie between $-\frac{L}{2}$ and $\frac{L}{2}$, then we should do the following:

$$\left\{ \begin{array}{lcl} r_{xi} \geq \frac{L}{2} & \Rightarrow & r_{xi} \longrightarrow r_{xi} - L \\ r_{yi} \geq \frac{L}{2} & \Rightarrow & r_{yi} \longrightarrow r_{yi} - L \end{array} \right\} \tag{A.1}$$

$$\left\{ \begin{array}{lcl} r_{xi} < -\frac{L}{2} & \Rightarrow & r_{xi} \longrightarrow r_{xi} + L \\ r_{yi} < -\frac{L}{2} & \Rightarrow & r_{yi} \longrightarrow r_{yi} + L \end{array} \right\} \tag{A.2}$$

where r_{xi} and r_{yi} denote the x and y coordinates of the atom i.

The wraparound consequence of the periodic boundary conditions is implied on the interaction computations. The interactions are pair interactions as mentioned previously and they are calculated based on the interatomic distances r. Therefore the interatomic distances should be checked and corrected according to the following:

$$\left\{ \begin{array}{lcl} r_{x} > \frac{L}{2} & \Rightarrow & r_{x} \longrightarrow r_{x} - L \\ r_{y} > \frac{L}{2} & \Rightarrow & r_{y} \longrightarrow r_{y} - L \end{array} \right\} \tag{A.3}$$

$$\left\{ \begin{array}{lcl} r_x < -\frac{L}{2} & \Rightarrow & r_x \longrightarrow r_x + L \\ r_y < -\frac{L}{2} & \Rightarrow & r_y \longrightarrow r_y + L \end{array} \right\} \tag{A.4}$$

where r_x and r_y are the x and y components of the displacement vector from atom i to atom j. The interactions are then calculated based on those new interatomic distances imposed by the wraparound effect.

A.2.1.2. Initial Conditions. The power of an MD simulation resides on the fact that it represents the real experimental findings independent of the initial conditions the system may have. Based on this fact, the atoms are assigned random initial positions on the simulation unit cell obeying the closure constraints imposed by the interaction potential (any two particles should not be closer to each other than a prescribed distance to avoid undesired numerical explosions in calculations). The initial velocities are also assigned random initial directions and a fixed magnitude based on temperature. They are also adjusted to ensure that the center of mass of the system is at rest.

A.2.1.3. Conservation Laws. For an MD simulation algorithm without the simulated annealing part, the linear momentum and energy should be conserved for all the steps. The conservation of linear momentum is intrinsic to the algorithm, and can only be violated by some error in the algorithm. Energy conservation is sensitive to the choice of the integrator for the equations of motion and the time stepsize of that integrator.

In the case of an MD simulation with simulated annealing, the linear momentum is still conserved, however, energy is not. Since the kinetic energy of the system is reduced gradually, the total energy of the system is not observed to be constant during the whole simulation. Although it is conserved during all the integration steps between the cooling steps, energy is not conserved in the overall run of the simulation.

In any simulation where the periodic boundary conditions are implied, angular momentum is not conserved. Because of the periodic boundaries, the rotational

invariance needed for angular momentum conservation is not applicable.

A.2.2. Simulation Algorithm

The simulation flowchart is presented in Figure A.1.

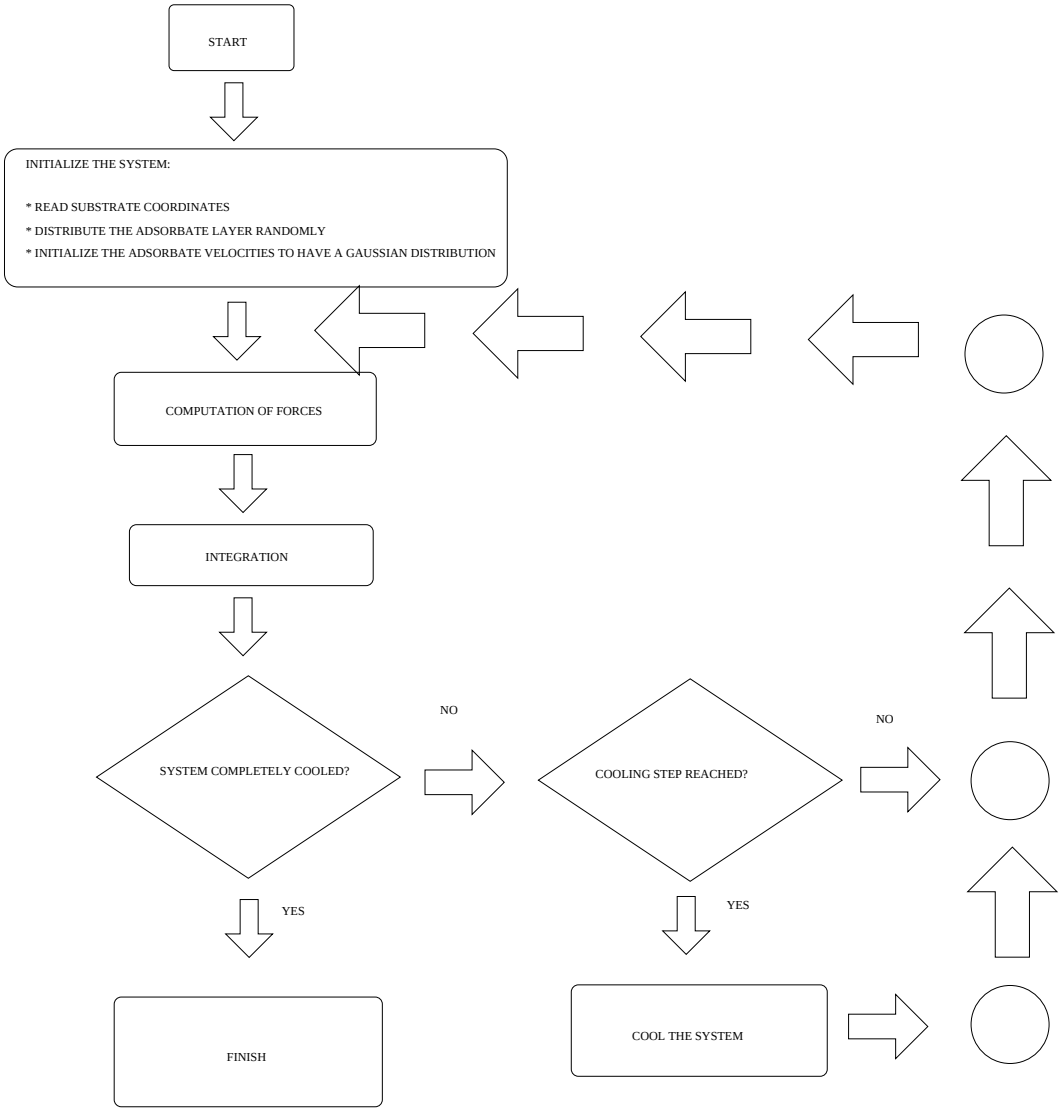

Figure A.1. The flowchart of the MD simulations with simulated annealing

A.2.2.1. Initialization. In this step of the simulations, the system is initialized according to the issues mentioned in the previous sections. The atomic coordinates of the substrate layer atoms are read from a file. The adsorbate atom coordinates are

distributed randomly over the simulation unit cell obeying the minimum separation constraint implied by the interatomic interactions, and their velocities are randomly oriented with a Gaussian speed distribution. The acceleration of all the adsorbate layer atoms is initialized as 0 in all directions.

A.2.2.2. Computation of Forces. Force calculations are performed according to Eqn. 3.2 in Section 3.2 giving rise to the acceleration of a single atom as in Eqn. 3.4. The force calculations are performed for every single pair in the system once since the force on atom i resulting from atom j will be equal in magnitude and opposite in direction to the force on atom j due to atom i. This consequence drawn out of Newton's third law, $\vec{F}_{ij} = -\vec{F}_{ji}$, puts Equation 3.4 in this form.

$$\begin{aligned}\ddot{\vec{r}}_i = &\sum_{j>i}^{N_a} \left(\frac{24\epsilon_a}{m_i r_{ij}}\right) \left[2\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \hat{r}_{ij} \\ &+ \sum_{\alpha=1}^{N_s} \left(\frac{24\epsilon_s}{m_i r_{i\alpha}}\right) \left[2\left(\frac{\sigma}{r_{i\alpha}}\right)^{12} - \left(\frac{\sigma}{r_{i\alpha}}\right)^{6}\right] \hat{r}_{i\alpha}\end{aligned} \tag{A.5}$$

A.2.2.3. Integration. In our MD simulations with simulated annealing, one of the simplest methods of numerical integration techniques, the Verlet algorithm is used. It is iterated as follows:

$$\vec{r}_i\left(t + \Delta t\right) = \vec{r}_i\left(t\right) + \vec{v}_i\left(t\right)\Delta t + \vec{a}_i\left(t\right)\frac{\left(\Delta t\right)^2}{2} \tag{A.6}$$

$$\vec{v}_i\left(t + \Delta t\right) = \vec{v}_i\left(t\right) + \left(\vec{a}_i\left(t\right) + \vec{a}_i\left(t + \Delta t\right)\right)\frac{\Delta t}{2}. \tag{A.7}$$

where $\vec{r}_i$ is the position, $\vec{v}_i$ is the velocity and $\vec{a}_i$ is the acceleration of the atom i. Δt is the integration time stepsize which is an important parameter from the point of view of energy conservation. The force computation is performed once more using the

new atomic coordinates of the adsorbate layer and then using the new accelerations, the new velocities are calculated as in Equation A.7. After these two easy steps the new coordinates together with the new velocities of the atoms of the subsrate layer are obtained. The newly calculated accelerations are saved and they serve as the old accelerations for the next step in the simulation.

If we put Equation A.7 in a different form and sum over all the particles, it is obvious that the linear momentum is conserved.

$$\sum_{i}^{N_a} \left[\vec{v}_i\left(t+\Delta t\right) - \vec{v}_i\left(t\right)\right] = \sum_{i}^{N_a} \left[\left(\vec{a}_i\left(t\right) + \vec{a}_i\left(t+\Delta t\right)\right)\frac{\Delta t}{2}\right] = 0 \qquad \text{(A.8)}$$

The right hand side contains the sum of acelerations in two different time steps, namely the forces in the adsorbate layer. Since the only forces are interatomic forces and no external forces are exerted on the system, the linear momentum of the system is conserved.

Annealing is applied at the second stage of the Verlet integrations regularly after a predefined number of time steps. The newly calculated velocities are multiplied by a reducing factor so that all the atoms in the system are slowed down and the kinetic energy of the system is reduced. As mentioned in Eqn. 3.8, this means a reduction in the temperature of the system, thus annealing. These iterations are performed until the system is cooled down to a level in which the particle motion, which is basically the oscillation about an equilibrium point, is much lower than the interatomic seperation. Towards this final step, the adsorbate atoms are almost located at the most favorable locations on the simulation unit cell for themselves, the average kinetic energy of the system is almost zero, and the atoms are slightly vibrating about their final equilibrium position.

A.2.2.4. Measurements And Control. During the run of the simulation mean kinetic, potential and total energies of the system are measured after each time step. Between cooling sessions, the system is set free to let the adsorbate layer atoms to come to a

new equilibrium, and the conservation of linear momentum and energy is checked in every time step until a new cooling step is reached. If there exists any inconsistency with the conservation laws, the simulation aborts. The most probable times that this situation might occur are the first few steps of the simulation (easily detectable and recoverable) and the few pre-cooling steps of the first 10% of the annealing process (which could be time consuming).

The simulation produces a tabular output informing about the completed percentage of the annealing process and mean kinetic, potential and total energies. If the simulation is run in the real-time mode, the cooling time steps could be observed together with the behavior of the atoms prior to and after the cooling step.

A.3. Test Code For Pure Adsorbate Layer

A demonstration of the simulation used to treat the problem of structure formation on a quasicrystalline substrate is presented and explained below. Only the interactions between the adsorbate layer atoms are calculated. The demonstration, as well as the whole simulation, is prepared to run under Linux systems with Gnuplot installed. The real time behavior of the system is displayed in a separate Gnuplot window which displays the atomic coordinates of the adsorbate layer atoms and updates after each time step -or after a predefined number of time steps.

In order to add the required user interface and preferences to the simulation, the following shell script (again written and can be run in Linux systems) is prepared:

```
1   #!/bin/sh
2   echo "Please enter the type of simulation : "
3   echo -e "(1) low density (gas) simulation \n
4   (2) medium density (liquid) simulation \n
5   (3) high density (solid) simulation \n"
6   read a1
7   echo "Would you like to perform annealing? (1) - Yes (0) - No"
8   read a2
9   if [ $a1 = 1 ]
10  then
11    gcc -D Nadd=50 -D hhh=2.34 -D Stepsize=0.0000025 -o ljpure ljpure.c -lm
12    ./ljpure out1.txt $a2 | gnuplot -noraise -persist
```

```
13    elif [$a1 = 2]
14    then
15      gcc -D Nadd=120 -D hhh=1.75 -D Stepsize=0.00000025 -o ljpure ljpure.c -lm
16      ./ljpure out1.txt $a2 | gnuplot -noraise -persist
17    elif [$a1 = 3]
18    then
19      gcc -D Nadd=400 -D hhh=1. -D Stepsize=0.000000025 -o ljpure ljpure.c -lm
21      ./ljpure out1.txt $a2 | gnuplot -noraise -persist
22    fi
```

The script starts with reading the type of the simulation in terms of the density of the adsorbate layer. The user has three choices of systems to simulate in terms of densities of the adsorbate layer: A low density, a medium density and a high density system. Relevant natural analogues could be a gas, a liquid and a solid system (only from the point of view of densities, no mechanical or structural properties are taken into consideration). The script then asks for the applicability of simulated annealing. After getting the required parameters from the user, the script compiles the source code for the required parameters. And then executes the code just compiled according to the adjustments made by the user.

The source code is written in C programming language and compiled and tested under systems running Linux. The three inputs from the user through the above script are the number of atoms in the adsorbate layer (which is the parameter that controls the atomic density of the adsorbate layer), the minimum allowable initial separation between any pair in the unit cell (which is necessary since the interaction exhibits infinite values for low interatomic distances) and the time step size of the Verlet integrator explained in Section A.2.2.3.

The source code for this simulation "ljpure.c" (denoting LJ interaction for pure adsorbate layer i.e. no substrate interaction) is presented below:

```
1    #include < stdio.h >
2    #include < math.h >
3    #include < stdlib.h >
4    #include < time.h >
5
6    #define Size 10.
```

```
7   #define Tol 0.01
8   #define initvel 1
9   #define iter 120
10  #define n 10
11  #define sigma 2.55
12  #define eps 0.25
13  #define Temp 10000.
14  #define pi 4. * atan(1.)
15  #define sqr(x)((x) * (x))
16
17  long double ra [Nadd] [2] = {0.}, va [Nadd] [2] = {0.}, aa [Nadd] [2] = {0.}, aa1 [Nadd] [2] = {0.}, vv, vm, dx,
    dy, dz, V, K, Km, E, Em;
18  double T, T1, cut;
19  int cool;
20  char * fname;
21
22  long double FitIn(long double a);
23  long double power(long double a, int c);
24  void AddDist();
25  void Init();
26  void Boundary();
27  void Energy();
28  void ComputeForces();
29  void Verlet();
30  void Initialize();
31  void WriteFile(char * filename);
32  void CoolOn();
33  void CoolOff();
34
35  long double FitIn(long double a)
36  {
37    if(a > Size/2.){a = a − Size; }
38    if(a < −Size/2.){a = a + Size; }
39    return a;
40  }
41
42  long double power(long double a, int c)
43  {
44    int i;
45    long double tt;
46    tt = 1.;
47    for(i = 1; i <= c; i + +)
48    {
49      tt = tt * a;
50    }
51    return(tt);
52  }
53
```

```
54  void AddDist()
55  {
56    int i, j, d, c;
57    long double r;
58    fprintf(stderr, "Distribution of the adsorbate layer.....");
59    ra[0][0] = (long double)Size * rand()/(RAND_MAX);
60    ra[0][1] = (long double)Size * rand()/(RAND_MAX);
61    c = 0;
62    for(i = 1; i < Nadd; i++)
63    {
64      d = 1;
65      while(d == 1)
66      {
67        d = 0;
68        ra[i][0] = (long double)Size * rand()/(RAND_MAX);
69        ra[i][1] = (long double)Size * rand()/(RAND_MAX);
70        for(j = 0; j < i; j++)
71        {
72          dx = FitIn(ra[i][0] - ra[j][0]);
73          dy = FitIn(ra[i][1] - ra[j][1]);
74          r = sqrt(sqr(dx) + sqr(dy));
75          if(r < hhh * 0.4){d = 1;}
76        }
77        c+ = 1;
78      }
79      fprintf(stderr,"%5d/%5d.....\b \ b \ b \ b \ b \ b \ b \ b \ b \ b \ b \ b \ b \ b \ b \ b \ b",i+1,Nadd);
80    }
81    fprintf(stderr, "%5d/%5d.....done \ n", i, Nadd);
82  }
83
84  void Init()
85  {
86    int i;
87    long double v, vsumx, vsumy, ang;
88    AddDist();
89    v = sqrt(2. * (1. - 1./Nadd) * Temp);
90    vsumx = 0.;
91    vsumy = 0.;
92    vv = 0.;
93    K = 0.;
94    for(i = 0; i < Nadd; i++)
95    {
96      ang = (long double)2. * pi * rand()/RAND_MAX;
97      va[i][0] = v * cos(ang);
98      va[i][1] = v * sin(ang);
99      aa[i][0] = 0.;
100     aa[i][1] = 0.;
101     aa1[i][0] = 0.;
```

```
102       aa1[i][1] = 0.;
103       vsumx = vsumx + va[i][0];
104       vsumy = vsumy + va[i][1];
105     }
106     vsumx = vsumx/Nadd;
107     vsumy = vsumy/Nadd;
108     for(i = 0; i < Nadd; i + +)
109     {
110       if(initvel == 1)
111       {
112         va[i][0] = va[i][0] − vsumx;
113         va[i][1] = va[i][1] − vsumy;
114         vv = sqr(va[i][0]) + sqr(va[i][1]);
115         K = K + vv;
116       }
117       else
118       {
119         va[i][0] = 0.;
120         va[i][1] = 0.;
121       }
122     }
123     K = K/2.;
124   }
125
126   void Boundary()
127   {
128     if((dx) > Size/2.){dx = dx − Size; }
129     if((dy) > Size/2.){dy = dy − Size; }
130     if((dx) < −Size/2.){dx = dx + Size; }
131     if((dy) < −Size/2.){dy = dy + Size; }
132   }
133
134   void Energy()
135   {
136     E = V + K;
137     Em = E/Nadd;
138     Km = K/Nadd;
139     vm = vm/Nadd;
140   }
141
142   void ComputeForces()
143   {
144     int i, j, s, c, a, b;
145     long double rra, rrs, kv, lv, kf, lf;
146     for(i = 0; i <= Nadd − 1; i + +)
147     {
148       aa[i][0] = 0.;
149       aa[i][1] = 0.;
```

```
150     }
151     V = 0.;
152     for(i = 0; i <= Nadd − 2; i + +)
153     {
154       for(j = i + 1; j < Nadd; j + +)
155       {
156         dx = ra[i][0] − ra[j][0];
157         dy = ra[i][1] − ra[j][1];
158         Boundary();
159         rra = sqrt(sqr(dx) + sqr(dy));
160         kv=(long double) 4.*eps*(power((sigma/rra),12)-power((sigma/rra),6));
161         kf=(long double) 24.*eps *(2.*power((sigma/rra), 12)-power((sigma/rra), 6)) *power((1./rra), 2);
162         aa[i][0] = aa[i][0] + kf ∗ dx;
163         aa[j][0] = aa[j][0] − kf ∗ dx;
164         aa[i][1] = aa[i][1] + kf ∗ dy;
165         aa[j][1] = aa[j][1] − kf ∗ dy;
166         V = V + kv;
167       }
168     }
169   }
170
171   void Verlet()
172   {
173     long double aad, bad;
174     int i, j;
175     for(i = 0; i <= Nadd − 1; i + +)
176     {
177       aad = Stepsize ∗ va[i][0] + (1./2.) ∗ sqr(Stepsize) ∗ aa[i][0];
178       bad = Stepsize ∗ va[i][1] + (1./2.) ∗ sqr(Stepsize) ∗ aa[i][1];
179       ra[i][0] = ra[i][0] + aad;
180       ra[i][0] = ra[i][0] − Size ∗ floor(ra[i][0]/Size);
181       ra[i][1] = ra[i][1] + bad;
182       ra[i][1] = ra[i][1] − Size ∗ floor(ra[i][1]/Size);
183       aa1[i][0] = aa[i][0];
184       aa1[i][1] = aa[i][1];
185     }
186     ComputeForces();
187     K = 0.;
188     vm = 0.;
189     for(i = 0; i < Nadd; i + +)
190     {
191       va[i][0] = (va[i][0] + (1./2.) ∗ (aa[i][0] + aa1[i][0]) ∗ Stepsize) ∗ T;
192       va[i][1] = (va[i][1] + (1./2.) ∗ (aa[i][1] + aa1[i][1]) ∗ Stepsize) ∗ T;
193       K = K + (sqr(va[i][0]) + sqr(va[i][1]))/2.;
194       vm = vm + va[i][0] + va[i][1];
195     }
196   }
197
```

```
198  void Initialize()
199  {
200    int i;
201    Init();
202    ComputeForces();
203    Energy();
204    T = 1.;
205  }
206
207  void WriteFile(char * filename)
208  {
209    FILE * id;
210    int j;
211    id = fopen(filename, "w");
212    for(j = 0; j <= Nadd - 1; j + +)
213    {
214      fprintf(id, "%12.10Lf%12.10Lf \ n", ra[j][0], ra[j][1]);
215    }
216    fclose(id);
217  }
218
219  void CoolOn()
220  {
221    int i, j;
222    fprintf(stderr, "Annealing is set to"On" \ n");
223    while(T >= 0.)
224  {
225      T1 = T;
226      T = 1.;
227      for(i = 1; i <= iter; i + +)
228      {
229        Verlet();
230        printf("set nokey \ n");
231        printf("plot[0 : %f][0 : %f] \ " - \" \ n", Size, Size);
232        for(j = 0; j < Nadd; j + +)
233        {
234          printf("%Lf%Lf \ n", ra[j][0], ra[j][1]);
235        }
236        printf("e \ n");
237      }
238      WriteFile(fname);
239      T = T1 - Tol;
240      Verlet();
241      Energy();
242      fprintf(stderr,"%6.2f%% <V>= %15.8Lf <K>= %15.8Lf <E>= %15.8Lf \ n",100* (1.-
     T),V/Nadd,Km,Em);
243    }
244  }
```

```
245
246  void CoolOff()
247  {
248    int i, j;
249    fprintf(stderr, "Annealing is set to"Off" \ n");
250    T = 1.;
251    for(i = 1; i <= 10000; i + +)
252    {
253      Verlet();
254      printf("set nokey \ n");
255      printf("plot[0 : %f][0 : %f] \ " − \" \ n", Size, Size);
256      for(j = 0; j < Nadd; j + +)
257      {
258        printf("%Lf%Lf \ n", ra[j][0], ra[j][1]);
259      }
260      printf("e \ n");
261      if((i%100) == 0)
262      {
263        Energy();
264        fprintf(stderr,"%5d/%5d| <V>= %10.2Lf| <K>= %10.2Lf| <E>= %10.2Lf| <v>= %6.4Lf \
     n",i,10000,V/Nadd,Km,Em,vm);
265      }
266    }
267    WriteFile(fname);
268  }
269
270  main(int argc, char ∗ ∗argv)
271  {
272    srand((unsigned)time(NULL));
273    fname = argv[1];
274    cool = atoi(argv[2]);
275    Initialize();
276    if(cool == 1){CoolOn(); }
277    else{CoolOff(); }
278    WriteFile(fname);
279  }
```

The explanation of basic variables and procedures in the code is as follows:

- The first four lines in the code include the general C header files required by the program.
- Lines $6 - 15$ are the definitions of some constants and macros used during the run of the program.

- Lines $17-20$ are the declarations of the variables used in the program together with their types and sizes that should be allocated in memory. The whole simulation is based on keeping track of the variables *ra* (position of each atom), *va* (velocity of each atom) and *aa* (acceleration of each atom). The lines $22-33$ are the function prototypes together with their arguments and the type of their outputs.
- Functions *FitIn* (lines $35-40$) and *Boundary* (lines $126-132$) are responsible for supplying periodic boundary conditions. They are called at different times and by different subroutines during the run.
- Function *power* (lines $42-52$) just raises its one input argument to the power given by its other input argument.
- Function *AddDist* (lines $54-82$) is responsible for the distribution of the adsorbate layer atoms in the unit cell randomly. The consraint for this distribution is that any two atoms should not be closer to each other than a predefined distance (0.37σ). Since LJ interaction approaches infinite values as the interatomic distance approaches zero, this is just a precaution against undesired numerical inflation of some variables in the program.
- Function *Init* (lines $84-124$) triggers *AddDist* function, and then initializes each atom's velocity. The velocities are randomly oriented where the distribution of the speeds is Gaussian. The velocities sum up to zero indicating the initial linear momentum of the system being zero.
- *Energy* (lines $134-140$) function just evaluates the mean values of the energies.
- Function *ComputeForces* (lines $142-169$) is responsible for calculation of interactions between every single pair in the adsorbate layer. It first initializes the accelerations of each atom, and then calculate the interatomic forces for each pair once, and then sums them up to get the net aceleration of each atom in the current atomic orientation of the adsorbate layer.
- Function *Verlet* (lines $171-196$) is the Verlet integrator decribed in Appendix A.2.2.3.
- Function *Initialize* (lines $198-205$) initializes the system systematically and performs the first interatomic interaction and energy calculations.
- *WriteFile* (lines $207-217$) is the function that writes the current atomic co-

ordinates of the adsorbate layer atoms into a file which is supplied to it as an argument.

- Function *CoolOn* (lines 219−244) is responsible for managing the whole run of the simulation if the simulated annealing option is turned on. It gradually decreases the speeds of the atoms each time a predefined number of iterations have been performed. It is also responsible for real time visualization of the system as well as informing the user about the present stage of annealing and invoking *Write File* function regularly.
- Function *CoolOff* (lines 246 − 268) has similar responsibilities as the function *CoolOn* has except that this time the temperature of the system is not adjusted, rather the system is simulated for a predefined number of timesteps in order to catch the long time behavior of the adsorbate layer together with its equilibrium schema. This mode is extrtemely useful when deciding the correct stepsize for the integrator for a completely new set of parameters since the conservation principles are best observable in the long run.
- The classical *main* (lines 270 − 279) function is where the job is distributed to related subroutines. The final atomic configuration of the adsorbate layer is sent to function *WriteFile* for the last time by function *main*.

When the substrate layer is taken into consideration, the function *ComputeForces* is modified as follows:

```
1   void ComputeForces()
2   {
3     int i, j, s, c, a, b;
4     long double rra, rrs, kv, lv, kf, lf;
5     for(i = 0; i <= Nadd − 1; i + +)
6     {
7       aa[i][0] = 0.;
8       aa[i][1] = 0.;
9     }
10    V = 0.;
11    for(i = 0; i <= Nadd − 2; i + +)
12    {
13      for(j = i + 1; j < Nadd; j + +)
14      {
15        dx = ra[i][0] − ra[j][0];
16        dy = ra[i][1] − ra[j][1];
17        Boundary();
```

```
18          rra = sqrt(sqr(dx) + sqr(dy));
19          kv=(long double) 4.*eps*(power((sigma/rra),12)-power((sigma/rra),6));
20          kf=(long double) 24.*eps *(2.*power((sigma/rra), 12)-power((sigma/rra), 6)) *power((1./rra), 2);
21          aa[i][0] = aa[i][0] + kf * dx;
22          aa[j][0] = aa[j][0] - kf * dx;
23          aa[i][1] = aa[i][1] + kf * dy;
24          aa[j][1] = aa[j][1] - kf * dy;
25          V = V + kv;
26        }
27      for(i = 0; i <= Nadd - 1; i + +)
28      {
29        for(j = 0; j < Nsubs; j + +)
30        {
31          dx = ra[i][0] - rs[j][0];
32          dy = ra[i][1] - rs[j][1];
33          dz = rs[j][2] + h;
34          Boundary();
35          rrs = sqrt(sqr(dx) + sqr(dy) + sqr(dz));
36          lv=(long double) 4.*eps*(power((sigma/rrs),12)-power((sigma/rrs),6));
37          lf=(long double) 24.*eps *(2.*power((sigma/rrs), 12)-power((sigma/rrs), 6)) *power((1./rrs), 2);
38          aa[i][0] = aa[i][0] + lf * dx;
39          aa[i][1] = aa[i][1] + lf * dy;
40          V = V + lv;
41        }
42      }
43    }
```

The second part of this function (lines $27 - 42$) calculates the interatomic forces between the adatoms and the substrate atoms. The atomic coordinates of the substrate layer are input from a file. While calculating the interatomic distances, the distances between the two substrate layers and between the adsorbate layer and the substrate surface layer are taken into consideration (line 33).

A.4. Simulating A Simple System

The simulations can be performed in a real-time visual environment where the atomic configuration of the adsorbate layer is plotted on the computer screen after each time step so that the simulations can be investigated for whether they produce reasonable output or not. If the behavior of the system is observed to be inconsistent with any of the constraints and conditions mentioned in the previous chapters, the

Table A.4. Sample simulation run output for a pure adsorbate system.

Step Number	$\bar{V}$	$\bar{K}$	$\bar{E}$	$\lvert\sum\vec{v}\rvert$
100/10000	34798.35	14094.72	48893.07	0.0000
200/10000	28008.16	20884.89	48893.05	0.0000
300/10000	22276.17	26616.88	48893.05	0.0000
400/10000	18417.91	30475.14	48893.06	0.0000
500/10000	16123.41	32769.66	48893.06	0.0000
600/10000	14872.43	34020.64	48893.07	0.0000
700/10000	14241.16	34651.91	48893.07	0.0000
800/10000	13998.81	34894.26	48893.07	0.0000
900/10000	14081.84	34811.23	48893.07	0.0000
1000/10000	14443.28	34449.79	48893.07	0.0000
2000/10000	13254.84	35638.22	48893.07	0.0000
3000/10000	11627.67	37265.40	48893.07	0.0000
4000/10000	13446.64	35446.41	48893.06	0.0000
5000/10000	14019.06	34874.01	48893.07	0.0000
7500/10000	12973.88	35919.19	48893.07	0.0000
10000/10000	17250.16	31642.90	48893.06	0.0000

parameters and the algorithms are re-investigated and revised if necessary.

The software used for the simulations offers the choice of annealing as described above. If annealing is not turned on, the program runs for a user defined number of timesteps. This allows the user to check the algorithm and its compliance with the conservation laws and the conditions it should fulfill. The program also outputs the mean kinetic, potential and total energies together with the net linear momentum of the system which allows the visualization of the fluctuations of the energies and the compatibility and the efficiency of the time step size chosen. A sample output of a run without annealing is shown in Table A.4.

It is obvious that the energy and the linear momentum are conserved. The time

step size of the Verlet integrator could be tested by means of such an analysis. If the energy is not conserved - yet this is still a matter of judgment, throughout this research mean total energy fluctuations in one per cent range are assumed to obey energy conservation - the stepsize is lowered by some percentage and the code is tested again.

A.5. Simulation LJ Potential

The LJ potential used in the simulations is plotted in Figure A.2.

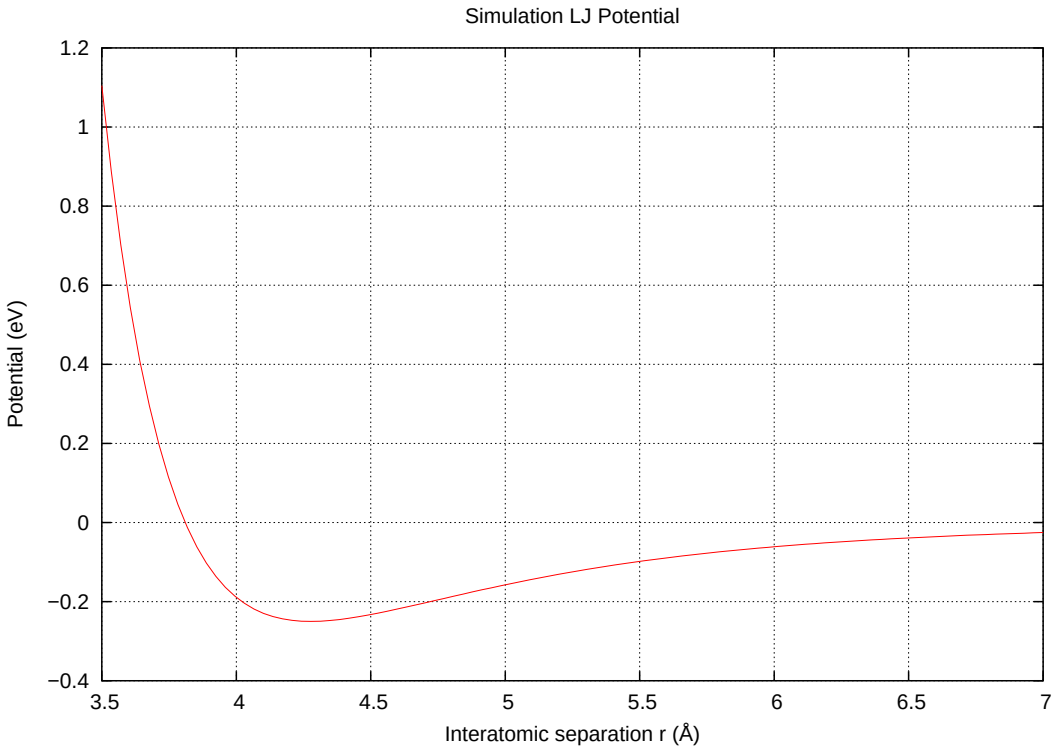

Figure A.2. Standard LJ potential with the parameters adjusted for the simulations

The potential together with the simulation runtime parameters has the following properties:

- $\epsilon = 0.25eV$ which controls the strength of the LJ interaction
- $\sigma = 3.81$Å which defines the range of the interaction
- The adsorbate layer is placed 3.5Å above the substrate surface layer.
- The adsorbate atoms are not allowed to have initial atomic coordinates that are

closer to each other less than 40 percent of 3.5Å in order to prevent undesired numerical inflation in the calculations.

- The initial velocities of the adsorbate layer atoms are adjusted in random directions and giving an overall Gaussian speed distribution.
- The time step size of the integrator is adjusted so that the conservation laws are not violated for a simulation without annealing.
- Annealing is applied by reducing the velocities of the adsorbate atoms by amounts gradually increasing by one per cent every cooling step after 120 time steps has elapsed. The system is let reach a new equilibrium in between.

APPENDIX B: SURFACE POTENTIAL OF THE SUBSTRATE LAYER

The code for obtaining the contour plot of the potential schema of the substrate layer under the LJ potential is written in IDL.

```
pro surface_plot
openr, 10,' result_381.txt'
aa = fltarr(501, 501)
readf, 10, aa
close, 10
openr, 11,' saitoh_50.txt'
bb = fltarr(4, 665)
readf, 11, bb
close, 11
openr, 12,' results381.txt'
cc = fltarr(3, 501)
readf, 12, cc
close, 12
rrr = 0.5

loadct, 13
device, decomposed = 0, retain = 2
surface, aa, cc[0, *], cc[0, *]
plots, aa[0, *], aa[1, *], psym = 1
triangulate, aa[0, *], aa[1, *], triangle, boundaryPts
shade_surf, aa, cc[0, *], cc[0, *], /save
contour, aa, /t3d, /noerase, zvalue = 3.0, /noclip
show3, aa, esurface
show3, aa
writejpeg,' contour.jpg', tvrd()

set_plot,' x'
device, decomposed = 0, retain = 2
plot,bb[0,*],bb[1,*],/nodata,/isotropic,xtitle='x-axis',ytitle='y-axis', &
xstyle=1,ystyle=1,xgridstyle=0,ygridstyle=0,clip=[-25.,-25.,25.,25.], &
psym = circ(rad = rrr, /fill), symsize = 1.0
oplot, bb[0, *], bb[1, *], psym = circ(rad = rrr, /fill), symsize = 1.0, &
color = 50 * 50 * 100
contour, aa, cc[0, *], cc[0, *], /isotropic, /noerase, zvalue = 0.0, /follow, /overplot

set_plot,' ps'
device, /encapsulated, /color, filename =' contours.eps'
plot,bb[0,*],bb[1,*],/nodata,/isotropic,xtitle='x-axis ( A)',ytitle='y-axis ( A)', &
```

```
39  xstyle=1,ystyle=1,xgridstyle=0,ygridstyle=0,clip=[-25.,-25.,25.,25.], &
40  psym = circ(rad = rrr, /fill), symsize = 1.0
41  oplot, bb[0, *], bb[1, *], psym = circ(rad = rrr, /fill), symsize = 1.0, &
42  color = 5 * 5 * 1
43  contour, aa, cc[0, *], cc[0, *], /isotropic, /noerase, zvalue = 0.0, /follow, &
44  ccolor = [30, 80, 110, 160, 230, 250], /overplot
45  device, /close
46  end
```

Lines $2-14$ read the data to be manipulated from the related files. Lines $16-25$ are responsible for generating a three dimensional surface plot of the potential schema over the $x-y$ plane while plotting the contours above the surface. These lines also export the figure produced to an output jpeg file. Lines $27-34$ perform the contour plotting and the real space orientation of the substrate layer simultaneously on the same scaling and on the computer screen. Either of these two plots can be filtered by cancelling the undesired plot by putting a semicolon in front of the line that produces the relevant plot. Lines $36-45$ perform the same job as in lines $27-34$ but this time they print the image on a postscript file. The name of the postscript file as well as the specific properties of the postscript image can be altered by adding some certain commands.

APPENDIX C: FOURIER TRANSFORM

Crystal structures are best understood by their diffraction images. Diffraction is the mechanism of scattering of waves or particles when they hit the structure that we would like to have information about. The scattered waves or particles experience constructive or destructive interference and by detecting the intensities of the scattered media and visualizing that data, we find out the underlying -obvious or misterious- structure of the crystal.

There is a close relation between the physical diffraction phenomena and the mathematical model of Fourier transformation. If we limit our discussion to X-ray diffraction and denote the amplitude of the diffracted wave from atom j in the atomic configuration we are searching the inner structure of by A_j, the diffracted wave could best be represented as

$$A_j = f_j e^{2\pi i \vec{k}\cdot\vec{r}_j} \tag{C.1}$$

where f_j is the scattering factor of the atom j, $\vec{r}_j$ is the real space coordinate of atom j atom and $\vec{k}$ is a unit vector in the diffraction space which is also called the reciprocal space [15]. If there are N atoms in the real space, then the total diffracted wave amplitude is

$$F(\vec{k}) = \sum_{j=1}^{N} A_j = \sum_{j=1}^{N} f_j e^{2\pi i \vec{k}\cdot\vec{r}_j} \tag{C.2}$$

where $F(\vec{k})$ is defined as the Fourier transform of the discreet scattering factor function f. $F(\vec{k})$ is called the structure factor and it describes the amplitude and phase of the diffracted waves. The structure factor F is a complex number and can be described as

$$F = A + iB \tag{C.3}$$

where $A = \sum_{j=1}^{N} f_j \cos\phi_j$, $B = \sum_{j=1}^{N} f_j \sin\phi_j$ and ϕ_j is the phase of the wave diffracted from atom j. The intensity disribution given by

$$I(\vec{k}) \approx |F|^2 = A^2 + B^2 \tag{C.4}$$

is just the diffraction pattern of the structure we are interested.

The software used in this thesis for obtaining diffraction images of the final structures of the adsorbate layers is taken from [10]. It is also modified to calculate the structure factors. The code is written in C programing language for Linux systems and is presented below.

```
1   #include < stdio.h >
2   #include < stdlib.h >
3   #include < math.h >
4
5   struct point
6   {
7     double x, y;
8   };
9
10  int getnumber(double * v)
11  {
12    int c;
13    for(;;)
14      {
15      switch(c = getchar())
16      {
17        case' ' : case' \ t' : case' \ n' : case' \ r' :
18        case',' : case'(': case')' : case'{': case'}' : case' [' : case']' :
19        continue;
20        case'#' :
21        while((c = getchar()) ! =' \n' && c = EOF)
22        ;
23        continue;
24        default :
25        ungetc(c, stdin);
26        return(scanf("%lf", v) == 1);
27      }
28    }
29  }
30
31  main(argc, argv)
```

```
32
33  char * argv[];
34  {
35    double srange = 10.0;
36    int n = 100;
37    double brightness = 1;
38    double scale;
39    int i, j, k;
40    struct point s;
41    struct point * source;
42    double re, im;
43    int nsource, allocated;
44    double dot, mag, maxmag;
45
46    if(argc < 1 || (n = atoi(argv[1])) <= 0 )
47    {
48      fprintf(stderr, "Usage: fourier N srange [brightness] < point.position.file > image.file.pgm \n \n");
49      exit(1);
50    }
51    if(n == 1) n == 2;
52    if(argc > 2) srange = atof(argv[2]);
53    if(argc > 3) brightness = atof(argv[3]);
54    / * Read in the light sources, enlarge the table when necessary * /
55    allocated = 15;
56    source = (struct point *)malloc(allocated * sizeof(struct point));
57    nsource = 0;
58    while(getnumber(&source[nsource].x) && getnumber(&source[nsource].y))
59    {
60      nsource + +;
61      if(nsource >= allocated)
62      {
63        allocated * = 2;
64        source = (struct point *)realloc(source, allocated * sizeof(struct point));
65      }
66    }
67    fprintf(stderr, "%d light sources. \ n", nsource);
68    scale = nsource > 0 ? brightness * 256 / nsource : 1;
69    / * produce portable grey map header * /
70    printf("P5 \ n%d %d \ n255 \ n", n, n);
71    maxmag = 0;
72    for(i = 0; i < n; i + +)
73    {
74      s.y = srange * ((float)i/(n - 1) - .5);
75      for(j = 0; j < n; j + +)
76      {
77        s.x = srange * ((float)j/(n - 1) - .5);
78        re = im = 0;
79        for(k = 0; k < nsource; k + +)
```

```
80          {
81            dot = 2 * M_PI * (s.x * source[k].x + s.y * source[k].y);
82            re += cos(dot);
83            im += sin(dot);
84          }
85          mag = scale * sqrt(re * re + im * im);
86          putchar(mag >= 255 ? 255 : (int)mag);
87          if(maxmag < mag) maxmag = mag;
88        }
89      }
90      if (maxmag > 0)
91  fprintf(stderr,"Max mag %.1f; max brightness which won't saturate: %g\n", maxmag,
    brightness * 256/maxmag);
92      exit(0);
93  }
```

This code takes a text file that contains the real space coordinates of a configuration and then calculates the diffraction image in pgm format. If the sums in lines 82 and 83 are evaluated continously without initialization the structure factor can be calculated. The program also supplies the maximum diffraction amplitude in the variable *maxmag*.

The pgm output from the Fourier transformation program presented above is a greyscale image that contains the brightness data as a matrix element for every single point in the image region. Therefore, the radial intensity distribution and the angular intensities could easily be calculated using the numerical values of brightness values in the pgm images which correspond to intensities. In order to do that the following IDL routine is written.

```
1  pro rdf
2  filename = dialog_pickfile(filter =′ ∗.pgm′)
3  read_ppm, filename, d
4  S = size(d)
5  A = S[1]
6  mid = A/2
7  md = mid − 0.5
```

```
p = fltarr(4)
tt = 0
pi = 4.0 * atan(1.0)
step = 4.0
c = intarr((mid/step) + 1)
fname =' rdff.txt'
openw, 10, fname
for r = 0.0, mid, step do begin
  ac = 0
  fr = floor(r)
  for i = 0, A - 1 do begin
    ia = abs(i - mid)
    if((ia le (fr + 2))) then begin
      for j = 0, A - 1 do begin
        ja = abs(j - mid)
        if((ja le (fr+2)) && (sqrt(ia^ 2+ja^ 2) le (fr+2)) &&  (sqrt(ia^ 2+ja^ 2) ge (fr-2))) then begin
          p[0] = sqrt((i - 0.5 - md)^2 + (j - 0.5 - md)^2)
          p[1] = sqrt((i - 0.5 - md)^2 + (j + 0.5 - md)^2)
          p[2] = sqrt((i + 0.5 - md)^2 + (j - 0.5 - md)^2)
          p[3] = sqrt((i + 0.5 - md)^2 + (j + 0.5 - md)^2)
          pmax = max(p)
          pmin = min(p)
          if ((r le pmax) && (r ge pmin)) then ac = ac + d[i, j]
        endif
      endfor
    endif
  endfor
  c[tt] = ac
  print, tt, ac
  tt = tt + 1
  printf, 10, r, ac
endfor
device, decomposed = 0, retain = 2
plot, c
close, 10
mm = reverse(sort(c))
rmax1 = mm[0] * step
rmax2 = mm[1] * step
rmax3 = mm[2] * step
print,' rmax1 =', rmax1
print,' rmax2 =', rmax2
print,' rmax3 =', rmax3
read,' Please enter which maximum shall be analyzed (1) (2) (3) or enter r  :', nmax
case nmax of
  1 : rmax = rmax1
  2 : rmax = rmax2
  3 : rmax = rmax3
  else : rmax = nmax
```

```
endcase
incra = 0.05
fname2 =′ adff.txt′
openw, 11, fname2
for theta = 0.0, 2.0 * pi, incra do begin
  x = mid + floor(rmax * cos(theta))
  y = mid + floor(rmax * sin(theta))
  angd = d[x, y]
  printf, 11, theta, angd
endfor
close, 11
end
```

The code reads the pgm file as an array, fixes the midpoint of the image and then starting from the midpoint, sweeps the image radially and adds up the intensity values for the related radius. After obtaining and plotting the radial intensity distribution, prompts the user to enter the value of the radius for which the angular intensity calculations would be performed. The user can select one of the first three maxima in the radial intensity distribution or enter a radius value manually. The program then sweeps the selected radius in angular direction from 0 to 2π radians recording the intensity data at the present polar location. Both the radial intensiy distribution data and the angular intensity calculations at desired radius are written in seperate files.

APPENDIX D: DOMAIN STRUCTURE ANALYSIS

The following code written in IDL investigates the domain structure in a given unit cell (a 50Å by 50Å unit cell for our computations) with a user defined number of atoms (this is a facility to add the code the applicability for a system with any desired density).

```
1   pro grains
2   common stred, aaaa, neigh, ang, NNN
3   ; set this to 1 for postscript output
4   iplot = 2
5
6   set_plot, 'x'
7   if (iplot eq 1) then begin
8     set_plot, 'ps'
9     device, /encapsulated, /color, filename =' grains.eps'
10    xoffset = 1.0, yoffset = 1.0
11  endif else begin
12    device, retain = 2
13  endelse
14
15  rrr = 0.2
16  Size = 50
17  filename = dialog_pickfile(filter =' *.txt')
18  openr, 10, filename
19  read,' Enter the number of points to be analyzed :', N
20  aa = fltarr(2, N)
21  readf, 10, aa
22  close, 10
23  plot,aa[0, *], aa[1, *], /isotropic, xtitle='x-axis ( A)', ytitle='y-axis ( A)', xstyle=1, ystyle=1,
    psym=circ(rad=rrr, /fill), symsize=4
24
25  cnt = 0
26  for i = 0, N - 1 do begin
27    if((aa[1, i] le - 20) || (aa[1, i] ge 20 )) then cnt + +
28  endfor
29
30  NN = N + cnt
31  aaa = fltarr(2, NN)
32  for i = 0, N - 1 do begin
33    aaa[0, i] = aa[0, i]
34    aaa[1, i] = aa[1, i]
35  endfor
36  i = N
```

```
37    for j = 0, N - 1 do begin
38      if(aa[1, j] le  - 20) then begin
39        aaa[0, i] = aa[0, j]
40        aaa[1, i] = aa[1, j] + Size
41        i + +
42      endif
43      if(aa[1, j] ge 20) then begin
44        aaa[0, i] = aa[0, j]
45        aaa[1, i] = aa[1, j] - Size
46        i + +
47      endif
48    endfor
49
50    cnt = 0
51    for i = 0, NN - 1 do begin
52      if((aaa[0, i] le  - 20) || (aaa[0, i] ge 20 )) then cnt + +
53    endfor
54    NNN = NN + cnt
55
56    aaaa = fltarr(2, NNN)
57    for i = 0, NN - 1 do begin
58      aaaa[0, i] = aaa[0, i]
59      aaaa[1, i] = aaa[1, i]
60    endfor
61    i = NN
62    for j = 0, NN - 1 do begin
63      if(aaa[0, j] le  - 20) then begin
64        aaaa[0, i] = aaa[0, j] + Size
65        aaaa[1, i] = aaa[1, j]
66        i + +
67      endif
68      if(aaa[0, j] ge 20) then begin
69        aaaa[0, i] = aaa[0, j] - Size
70        aaaa[1, i] = aaa[1, j]
71        i + +
72      endif
73    endfor
74
75    neigh = intarr(6, NNN)
76    ang = fltarr(6, NNN)
77    for i = 0, NNN - 1 do begin
78    find_neighbours, i
79    find_angles, i
80    endfor
81    eps = 0.35
82    carr = intarr(NNN)
83    c1 = 1
84    for i = 0, NNN - 1 do begin
```

```
85      if(carr[i] eq 0) then begin
86        carr[i] = c1
87        for j = 0, NNN - 1 do begin
88          if(carr[j] eq 0) then begin
89            c2 = 0
90            for k = 0, 5 do begin
91              if(abs(ang[k, i] - ang[k, j]) le eps) then c2 + +
92            endfor
93            if(c2 eq 6) then carr[j] = c1
94          endif
95        endfor
96        c1 + +
97      endif
98    endfor
99    for i = 0, NNN - 1 do begin
100     if((aaaa[0, i] lt - 25) || (aaaa[0, i] gt 25)) then carr[i] = 0
101     if((aaaa[1, i] lt - 25) || (aaaa[1, i] gt 25)) then carr[i] = 0
102   endfor
103   maxcar = max(carr)
104   cpp = intarr(maxcar)
105   for i = 0, NNN - 1 do begin
106     if(carr[i] ne 0) then cpp[carr[i] - 1] + +
107   endfor
108   fact = fix(255/maxcar)
109   for i = 0, maxcar - 1 do begin
110     aap = fltarr(2, cpp[i])
111     slct = where(carr eq (i + 1), count)
112     if(count ne 0) then begin
113       for j = 0, count - 1 do begin
114         k = slct[j]
115         aap[0, j] = aaaa[0, k]
116         aap[1, j] = aaaa[1, k]
117       endfor
118     endif
119     c1 = i * 25 + 100
120     c2 = 250 + i * 10
121     c3 = i * 24 + 15
122     oplot,aap[0,*],aap[1,*],psym=circ(rad=rrr,/fill),symsize=4,color=c1*20*10
123   endfor
124   if (iplot eq 1) then device, /close
125   end
126
127   pro find_neighbours, ind
128   common stred
129   dist = fltarr(NNN)
130   Size = 50
131   j = 0
132   for i = 0, NNN - 1 do begin
```

```
133     dx = aaaa[0, i] − aaaa[0, ind]
134     dy = aaaa[1, i] − aaaa[1, ind]
135     if(dx > Size/2) then dx = dx − Size
136     if(dx < (−1.) ∗ Size/2) then dx = dx + Size
137     if(dy > Size/2) then dy = dy − Size
138     if(dy < (−1.) ∗ Size/2) then dy = dy + Size
139     dist[j] = sqrt((dx)^2 + (dy)^2)
140     j + +
141   endfor
142   srt = sort(dist)
143   for i = 0, 5 do begin
144     neigh[i, ind] = srt[i + 1]
145   endfor
146   end
147
148   pro find_angles, ind
149   common stred
150   Size = 50
151   angl = fltarr(6)
152   for i = 0, 5 do begin
153     dd = neigh[i, ind]
154     dx = aaaa[0, dd] − aaaa[0, ind]
155     dy = aaaa[1, dd] − aaaa[1, ind]
156     if(dx > Size/2) then dx = dx − Size
157     if(dx < (−1.) ∗ Size/2) then dx = dx + Size
158     if(dy > Size/2) then dy = dy − Size
159     if(dy < (−1.) ∗ Size/2) then dy = dy + Size
160     if(dx ne 0) then angl[i] = atan(dy/dx) else angl[i] = 2 ∗ atan(1.)
161     if(angl[i] < 0) then angl[i] = angl[i] + 8. ∗ atan(1.)
162   endfor
163   srt = sort(angl)
164   for i = 0, 5 do begin
165     ang[i, ind] = angl[srt[i]]
166   endfor
167   end
```

Lines $1-23$ read the coordinate data in the selected file with the correct number of data in it, and then plot the data on the screen. They contain the information for a postscript output as well. Lines $25-73$ are responsible for enlarging the unit cell with periodic boundary conditions taken into consideration in order to prevent the edge effects in the final picture with the grains visualized by different colors. Lines

75 and 76 allocate the required arrays for each atom in the extended configuration containing the data for their neighbour atom listing and the angles they make relative to the center atom. Lines $77-80$ fills these two arrays with their corresponding value using the procedures $find\_neighbours$ (lines $127-146$) and $find\_angles$ (lines $148-167$). $find\_neighbours$ tests for the distance between the center atom and all the remaining atoms and then takes the indices of the six closest atoms in the neighbour list. $find\_angles$ sets up cartesian coordinate axes at the center atom and calculates the angles the neighbours make with the horizontal in radians. Then it sorts the angles for comparison purposes and then records the values in the angles array. Lines $81-98$ compares the sorted angles within some predefined accuracy (line 81), and if all six angles coincide these lines mark the related atoms indicating that they belong to the same domain. Lines $99-125$ identify every single domain and then plot them one by one assigning different color indices. The resulting display is an interesting plot of the coordinates in which every domain is plotted with a different color. By the aid of this analysis, the number of domains in the final output of the simulation together with the size of the domains are visualized.

REFERENCES

1. Flückiger, T., Y. Weisskopf, M. Erbudak, R. Lüscher and A.R. Kortan, "Nanoepitaxy: Size Selection in Self-Assembled and Oriented Al Nanocrystals Grown on a Quasicrystal Surface", *Nano Letters*, Vol. 3, 1717, 2003.

2. Saitoh, K., K. Tsuda and M. Tanaka, "New Structural Model of an $Al_{72}Ni_{20}Co_8$ Decagonal Quasicrystal", *Journal of the Physical Society of Japan*, Vol.67, No.8, pp.2578-2581, 1998.

3. Abraham, F.F., "Computer Simulations of Surfaces, Interfaces, and Physisorbed Films", *The Journal of Vacuum Science and Technology B*, Vol.2, No.3, pp.534-549, 1984.

4. Reichhardt, C., C. J. Olson and F. Nori, "Commensurate and Incommensurate Vortex States in Superconductors With Periodic Pinning Arrays", *Physical Review B*, Vol.57, No.13, 1998.

5. International Union of Pure And Applied Chemistry, *IUPAC Compendium of Chemical Terminology*, http://www.iupac.org/publications/compendium/index.html

6. Hu, J. and R. M. Westerwelt, "Commensurate-Incommensurate Transitions In Magnetic Bubble Arrays With Periodic Line Pinning",*Physical Review B*, Vol.55, No.2, 1997.

7. Abraham, F.F., S. W. Koch and W. E. Rudge, "Molecular-Dynamics Computer Simulation of The Weakly Incommensurate Phase of Monolayer Krypton on Graphite", *Physical Review Letters*, Vol.49, No.25, 1982.

8. Abraham, F.F., W. E. Rudge, D. J. Auerbach and S. W. Koch, "Molecular-Dynamics Simulations of The Incommensurate Phase of Krypton on Graphite Using More Than 100000 Atoms", *Physical Review Letters*, Vol.52, No.6, 1984.

9. Zeppenfeld, P., J. Goerge, V. Diercks, R. Halmer, R. David and G. Comsa, "Orientational Ordering On A Corrugated Substrate: Novel Pinwheel Structure For N_2 Adsorbed On $Cu(110)$", *Physical Review Letters*, Vol.78, No.8, 1997.

10. Senechal, M., *Quasicrystals and Geometry*, Cambridge University Press, Cambridge, 1996.

11. Suck, J.B., M. Schreiber and P. Häussler, Eds., *Quasicrystals: An Introduction to Structure, Physical Properties, and Applications*, Springer Series in Material Science, Germany, 2002.

12. Kittel, C., *Introduction to Solid State Physics*, John Wiley & Sons Inc., New York, 1996.

13. Steinhardt, P.J., "New Perspectives on Forbidden Symmetries, Quasicrystals, and Penrose Tilings", Proceedings of the National Academy of Sciences USA, Colloquium Paper, Vol.93, pp.14267-14270, December, 1996.

14. Rappaport, D.C., *The Art of Molecular Dynamics Simulation*, Cambridge University Press, Cambridge, 1998.

15. Giacovazzo, C., Ed., *Fundamentals of Crystallography*, Oxford University Press, Oxford, 2002.

16. See e.g. Challis, L.J., "Physics in Less Than Three Dimensions", *Contemporary Physics*, Vol.33, No.2, pp.111-127, 1992.

17. Weisskopf, Y., R. Lüscher and M. Erbudak, "Structural Modifications Upon Deposition of Fe on the Icosahedral Quasicrystal Al-Pd-Mn", *Surface Science*, Vol.578, pp.35-42, 2005.

18. Lüscher, R., M. Erbudak and Y. Weisskopf, "Al Nanostructures on Quasicrystalline Al-Pd-Mn", *Surface Science*, Vol.569, pp.163-175, 2004.

19. Kirkpatrick, S., C. D. Gelatt Jr. and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol.220, pp.671-680, 1983.