STUDIES ON THE CHANNELING OF FAST PARTICLES THROUGH CRYSTALS

by

JEENA K

Thesis submitted to
University of Calicut
in partial fulfillment of the requirements
for the award of the degree of
DOCTOR OF PHILOSOPHY

Department of Physics
University of Calicut
December 2007

Dedicated to my parents

CERTIFICATE

Certified that the work presented in this thesis is a bonafide work done by Ms. Jeena. K, under my guidance in the Department of Physics, University of Calicut and that this work has not been included in any other thesis submitted previously for the award of any degree either in this university or any other university/institution.

Dr. K. Neelakandan

Supervising Guide

University of Calicut

December 2007

DECLARATION

I hereby declare that the work presented in this thesis is based on the original work done by me under the guidance of Dr. K. Neelakandan, Professor, Department of Physics, University of Calicut and has not been included in any other thesis submitted previously for the award of any degree.

Jeena. K

University of Calicut

December 2007

ACKNOWLEDGEMENTS

First of all I express my deep sense of gratitude to Prof. K. Neelakandan, Department of Physics, Calicut University, for his constant guidance and supervision throughout the course of this work.

My sincere thanks to Prof. K. M. Varier, Head, Department of Physics, University of Calicut for providing me the facilities needed to bring out this thesis.

I thank whole heartedly Dr. K. Rajasekharan, Malabar Christian College, Calicut and Dr. B. R. S. Babu, Department of physics, Calicut University, without whose help this thesis would not have been possible.

I express my sincere thanks to Mr. K. K. Abdullah, Mr. Bipin and Mr. K. M. Udayanandan, for their kind cooperation and immense help rendered to me. I am thankful to all the teaching and non teaching staff of department of physics and my friends for offering a good support throughout my course work. I also acknowledge the financial support rendered by University of Calicut to carry out this research work..

No word can communicate my heartfelt gratitude to my parents and family members for their support and encouragement.

Jeena. K

Contents

List of Figures

List of Tables

Chapter 1

An Introduction to Channeling

Any particle traversing amorphous matter or a misaligned crystal experiences a number of uncorrelated collisions with atoms. These encounters occur with a distribution of small or large impact parameter. A variety of processes take place in the collision events. The most common are, angular scattering in multiple collisions with the atomic nuclei and energy loss in collisions with atomic electrons. For a homogeneous and isotropic target material, the impact parameters involved in collisions with the individual target atoms are independent of the relative orientations of the beam direction and target. Hence the yields of these interactions are random.

When the target material is mono crystalline, the situations become quite different. The distribution of impact parameters and the yields of physical processes are found to be strongly dependent on the relative orientations of beam and target. If a beam of positive charged ions is incident on a crystal, the angle between the direction of the incident beam and a particular axis or plane in the crystal is within a small but predictable range. The gradually changing electrostatic repulsion between the incident particle and each successive atomic nucleus of the crystal steers the beam through the crystal lattice. This anisotropic effect is commonly refered as channeling.

1.1 Basic Ideas

From a classical standpoint, one may qualitatively understand the channeling effect as follows. If the direction of a charged particle incident upon the surface of a mono crystal lies close to a major crystal direction, the particle, with high probability, will suffer a small angle scattering as it passes through the several layers of atoms in the crystal. If the direction of the particle's momentum is close to atomic planes, ions follow trajectories which oscillate to and fro in the open channels as they pass through the crystal. Such a motion as shown in figure(1.1) is refered as planar channeling.

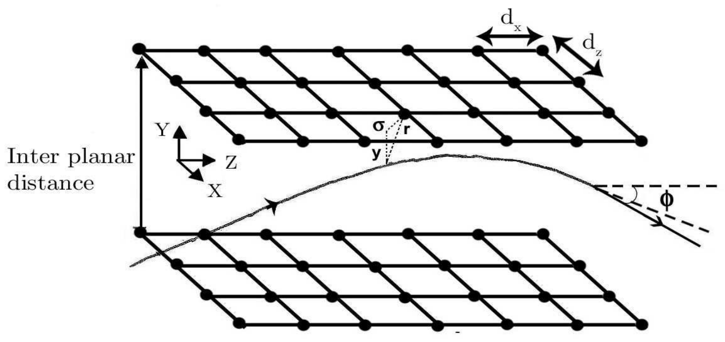

Figure 1.1: Schematic representation of highly exaggerated planar channeling

The same phenomena occur when the direction of the momentum of the charged particle incident upon the surface of a crystal lies close to a major crystalline high symmetry axis. The particle then again undergoes a correlated series of gentle small angle collisions. This is known as axial channeling. See figure(1.2). Similarly for very low angle of incidence with respect to a channeling direction an ion might even get trapped between a set of nearest atomic rows as it traverses the crystal. This type of motion is called hyper-channeling. See figure(1.3)

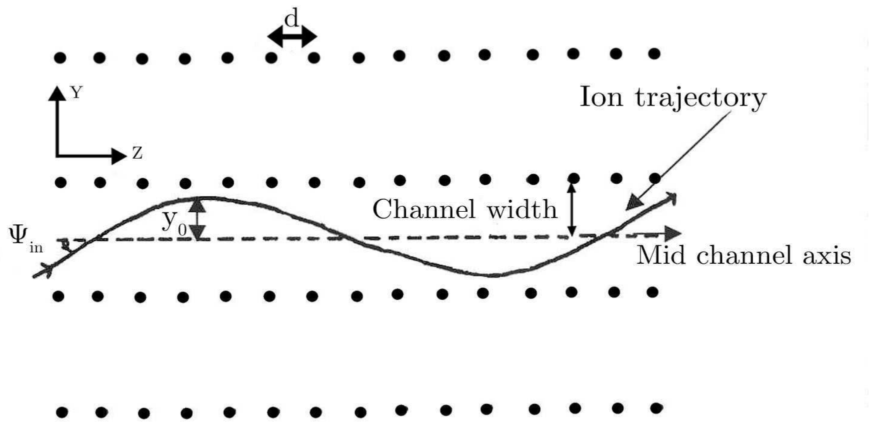

Figure 1.2: Schematic representation of highly exaggerated axial channeling. The shaded circles represent lattice sites of crystal

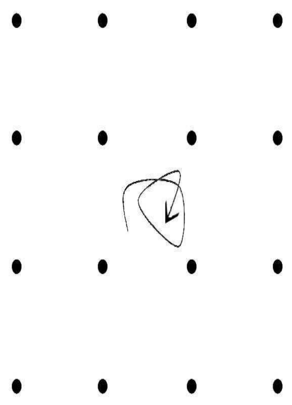

Figure 1.3: Schematic representation of hyper-channeling of ions. The black circles represent the mean position of atomic rows when viewed in the direction of a simple axis and the curves are the projected motion of the ion on to the plane normal to the direction

1.2 Experimental Arrangement

A schematic diagram of a typical experimental arrangement for measurements on the channeling effect is shown in figure(1.4). A well collimated beam of particles is incident upon a mono crystalline target mounted on a goniometer. The target is oriented so that there is a small angle ψ between some axial or planar direction in the crystal and the beam direction. Channeling measurements usually consist of determining either the ψ dependence of the yield of close encounter processes or the distributions in space or momentum of the transmitted beam. For thin targets the transmitted beam can be examined with a photographic plate, a fluorescent screen or detector 1. The yield of close encounter processes is measured by keeping detector 2 at a wide angle to the incident beam. We now consider some of the notable experiments in channeling bringing out the salient features of channeling effects.

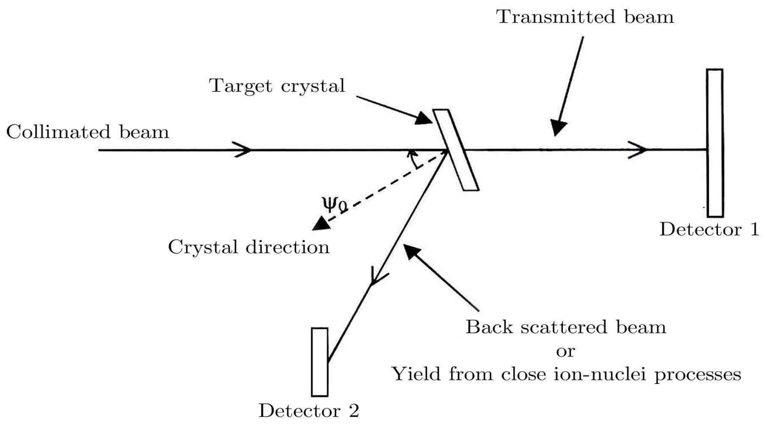

Figure 1.4: Schematic arrangement of a typical channeling experiment

1.3 Penetration Experiments

Experiments fall into two main categories. Either ions are transmitted through thin single crystals into a detector or radioactive ions are allowed to come to rest in a thick crystal and their range is measured.

1.3.1 Transmission experiment

Figure(1.4) with detector 1 alone gives the schematic arrangement of transmission experiment. Results of Nelson and Thomson[1], using a 50keV proton beam and gold target showed increased penetration of ions close to a channel direction. Using a current measuring electrode as detector 1, the intensity of the emergent beam is measured. Figure(1.5) shows the typical variation

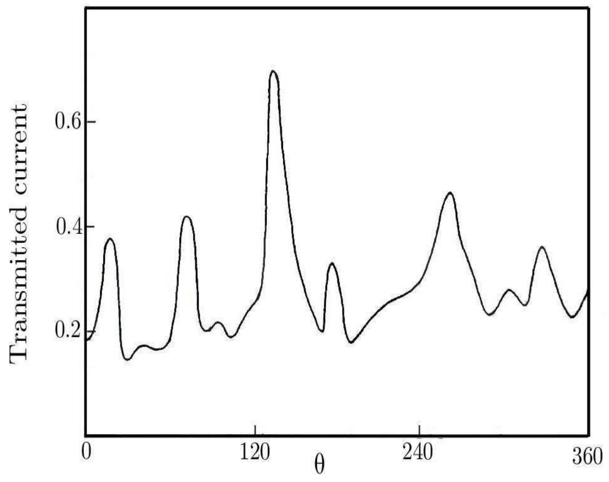

Figure 1.5: Variation of the transmitted ion current with rotation about ⟨111⟩ axes in Au crystal.

in measured transmitted current when the crystal is rotated about the ⟨111⟩ axes. The peaks in the transmitted current occur when the incident beam was within about 4^0 of the ⟨110⟩ axes. For the fcc structure of gold these are the most closely packed directions and hence the directions of most open spaces. The effects were found to persist for proton energies up to at least two orders of magnitude greater, although at such high energies the angular range over which the effect could be seen falls to a fraction of a degree.

1.3.2 Range measurements

Range measurements of the Chalk River and Munich groups [2],[3] showed enhancement of penetration for incidence close to channeled direction for heavy ions of kilovolt energies. Range of an ion is defined as the longitudinal

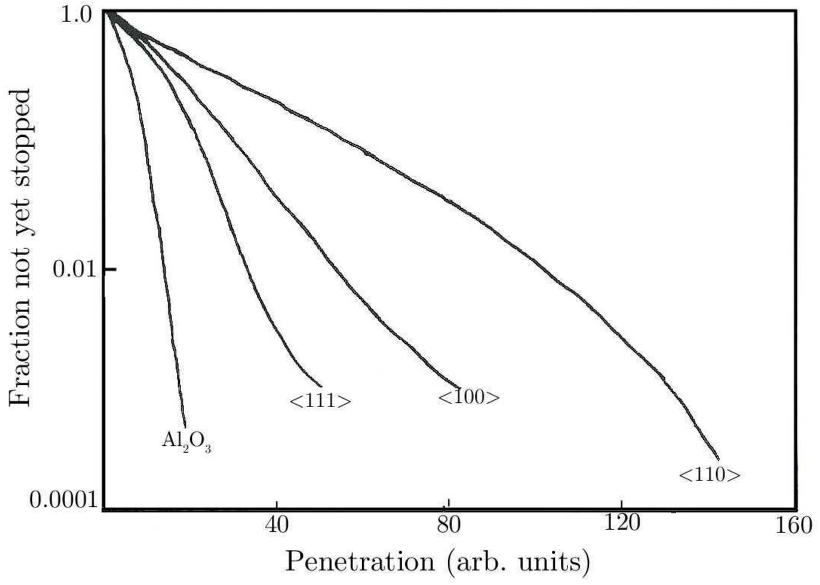

Figure 1.6: Penetration of 40keV kr^{86} ions along the principal crystallographic directions of Al and in amorphous Al_2O_3

distance covered by the ion in the crystal before it loses all its energy and is finally brought to rest. Higher range for a channeled ion comes from a number of factors. First the channeled trajectory loses less energy in travelling a given distance, since fewer violent encounters occur with individual atoms. Secondly, the ion travels more nearly in a straight line than a random trajectory which wanders through the crystal in a zig zag path. Figure(1.6) shows how the distribution of krypton ions which came to rest in an aluminium crystal varies with the orientation of the target. The results clearly illustrate, that the most open channels are those between the most closely packed rows in the crystal structure. For fcc crystals, $\langle 110 \rangle$ direction is the most closely packed and the range is found to be maximum along this direction.

1.4 Close Encounter Process

A channeled trajectory always avoids the nuclei, which are the centres of repulsion for the positive ions and are producing the required steering action for channeling. In order to initiate a nuclear reaction such as

$$Si^{28} + p \rightarrow P^{29} + \gamma \tag{1.1}$$

$$Cu^{65} + p \rightarrow Zn^{65} + n \tag{1.2}$$

the protons have to approach the nucleus to a distance roughly 10^{-4} times the channel width. Therefore if a proton beam is incident on a crystal close to channel directions, we can expect a minimum in the measurement of nuclear reaction yields. The typical orientation dependence of the yield of a close encounter process measured in a channeling experiment[4] is illustrated in figure(1.7). In measurements on the yield of close encounter processes, the detector is kept in a random direction, in order to eliminate the possible anisotropic effects. The yield is measured by varying the angle ψ over a small range centered on $\psi = 0$. The yield χ is normalized to random. The yield χ=1 is the yield that would be measured from a hypothetical amorphous target having the same characteristics as the target chosen.

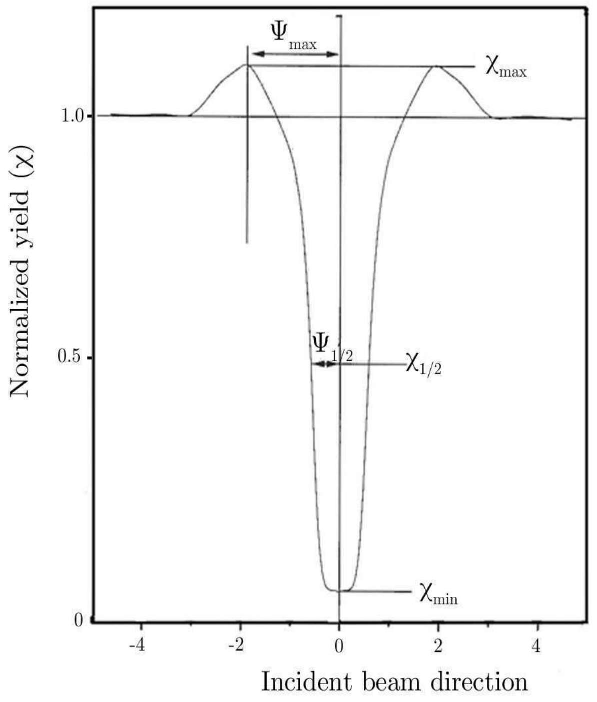

Figure 1.7: The normalized depth dependent yield of a close-encounter process

When $\psi = 0$ the channeled fraction of the incident beam is a maximum, and thus the yield of close encounter process is at its minimum value χ_{min}. As the crystal is tilted away from the beam, the channeled fraction decreases, and χ rises to a maximum value χ_{max}, which can be somewhat greater than unity. This results in a shoulder like pattern on either side of the minimum. At still larger angular deviations, the yield approaches the random value. ψ_{max} is the angle at which χ=χ_{max}. Channeling is characterized by the minimum yield χ_{min} and half-width $\psi_{1/2}$ at the point where the yield $\chi_{1/2} = \frac{1}{2}[1 + \chi_{min}]$. The measured yield χ is dependent upon the depth

x at which the close encounter process occurs inside the target crystal. The depth dependence of χ is primarily a consequence of multiple scattering of the beam as it progresses through the crystal.

1.5 Blocking

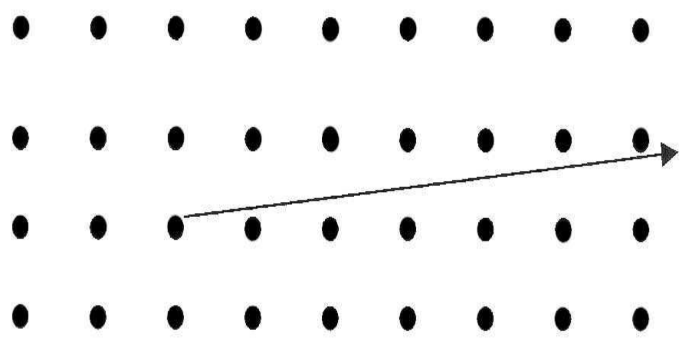

Figure 1.8: Schematic representation of blocking effect

In an experiment closely related to the nuclear reaction, one may take a crystal containing alpha radioactive atoms as members of the lattice. The emitted alpha particles will not be able to emerge from the crystal in the exact channel directions, since they would then have to pass through a row of intervening nuclei, and they are steered away as shown in figure(1.8). This is often referred to as blocking effect. If we set a detector to collect alpha particles emitted from a crystal in a specific direction, the counting rate should fall to a minimum when the detector is placed in the channel direction. Figure(1.9) shows the results of such an experiment done by Domeij et al[5].

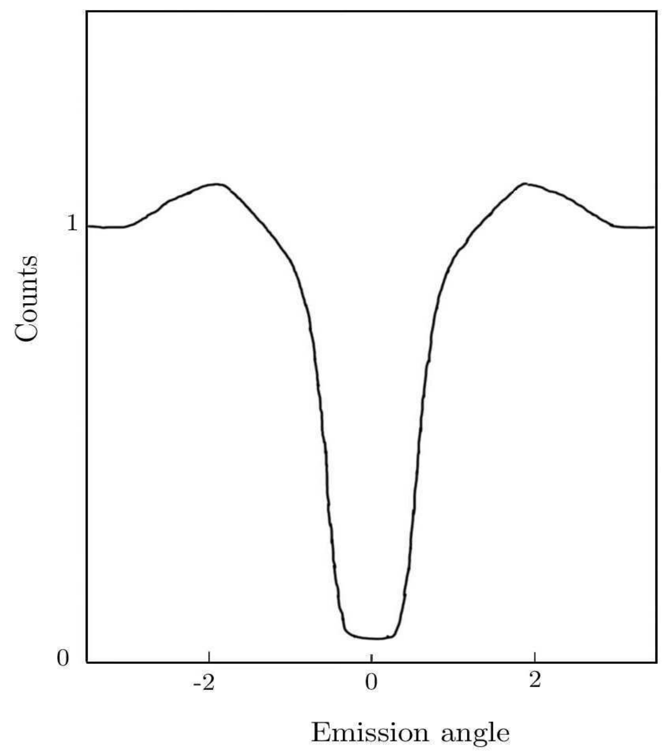

Figure 1.9: Typical intensity variation as a function of emission angle for 5.5MeV alpha particles from Ra^{222} atoms embeded in Tungsten crystal. The dip is centered on a simple crystal axis

1.6 Back Scattering Experiments

When channeling of the incident ion occurs, the chances of an ion being scattered through a large angle and returning to the surface will be greatly reduced. In any experiment which measures the back scattered current of particles from a crystal, we expect a large attenuation whenever the ion beam is channeled along a simple crystal direction. Nelson and Thomson[1] found the first evidence for the strong influence of channeling on backscattering yields. They observed a sharp minima in the scattering of 50keV H^{+}, He^{+},

Ne^+ and Xe^+ ions from a Cu crystal, whenever the incident beam is along a channeling direction.

1.7 Theory of Channeling

The passage of energetic positively charged ions through a crystal lattice is a complicated process. To understand how the crystal symmetry steers a particle beam, let us briefly consider the particle interaction with a single atom.

1.7.1 Ion crystal interaction

The interaction between an ion and a crystal is divided into(i) elastic interactions with the atomic nuclei and (ii) inelastic interactions involving the electrons. The elastic interaction between the ion and the atom at a distance r is described by an interaction potential $V(r)$. At low separations, $V(r)$ is described by the coulombic repulsion of the bare nuclei modified by an appropriate function which describes electronic screening. The following potentials are commonly used.

1. The Lindhard standard potential[6]

$$V(r) = \frac{Z_1 Z_2 e^2}{r} \left\{ 1 - \frac{r}{(r^2 + C^2 a_T^2)^{\frac{1}{2}}} \right\} \tag{1.3}$$

where Z_1 and Z_2 represents the atomic number of ion and atom respectively, e is the electronic charge, C is an adjustable parameter approximately equal to $\sqrt{3}$ and a_T is the Thomas Fermi screening radius given by

$$a_T = 0.8853 a_0 \left(Z_1^{\frac{2}{3}} + Z_2^{\frac{2}{3}} \right)^{-\frac{1}{2}} \tag{1.4}$$

where $a_0 = 0.53A^0$, is the Bohr radius.

2. The Moliere potential[7]

$$V(r) = \frac{Z_1 Z_2 e^2}{r}\left\{0.1e^{-\frac{6r}{a_T}} + 0.35e^{-\frac{0.3r}{a_T}} + 0.55e^{-\frac{1.2r}{a_T}}\right\} \tag{1.5}$$

and

3. Bohr potential[8]

$$V(r) = \frac{Z_1 Z_2 e^2}{r} e^{-\frac{r}{a_T}} \tag{1.6}$$

At relatively large separations, the interatomic potential is often described by a Born-Mayer potential

$$V(r) = A_{BM} e^{-Br} \tag{1.7}$$

Where A_{BM} and B are empirical parameters.

When the ions are channeled, to a first approximation, ions effectively *see* a continuously spread out rows of positively charged nuclear strings or nuclear planes. The ion-nuclei interaction can then be replaced by ion-atomic row or ion-atomic plane interaction using the so called continuum string potential or continuum planar potential respectively. This continuum model is briefly described in the next section.

1.8 Continuum Model

A comprehensive treatment of continuum model was given by Lindhard [9]. The essential simplification in this model is to average the ion-atom interaction potential along the major axes or planes so that the scattering is reduced from three to two or one dimension respectively. The great advantage of this model is that it enables one to describe the average properties of a given trajectory or alternatively to describe the average properties of an entire beam of ions. The continuum potential at a distance $r_\perp$ from a string of atomic nuclei is written as the average of inter-atomic potential $V(r)$ given by

$$U_1(r_\perp) = \frac{1}{d}\int_{-\infty}^{+\infty} dz V\left(\sqrt{z^2 + r_\perp^2}\right) \tag{1.8}$$

where d is the inter atomic spacing along the atomic row, z is the distance of the ion to the nucleus measured along the row in the channeling direction so that $r = \sqrt{z^2 + r_\perp^2}$ is the distance between the ion and the nucleus(figure(1.10)). Substituting eqn(1.3) in eqn(1.8) gives the standard

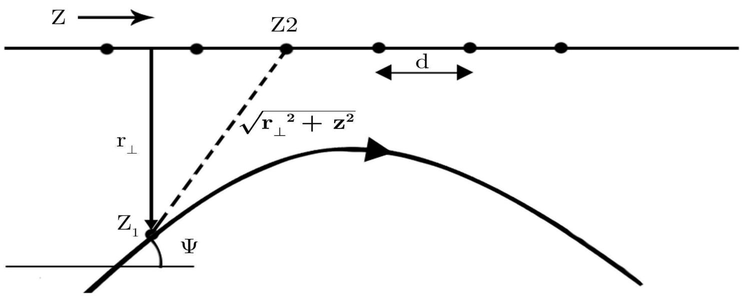

Figure 1.10: Collisions of channeled particles with idealized static linear array of atoms.

Lindhard continuum potential for a string at $\vec{r}_{\perp i}$

$$U_1(|\vec{r}_\perp - \vec{r}_{\perp i}|) = \left(\frac{2Z_1Z_2e^2}{d}\right) 0.5ln\left\{1 + \left(\frac{Ca_T}{|\vec{r}_\perp - \vec{r}_{\perp i}|}\right)^2\right\} \tag{1.9}$$

the potential at $\vec{r}_\perp$ due to all strings in the plane is then

$$U(\vec{r}_\perp) = \sum_{i=1}^{N} U_1(|\vec{r}_\perp - \vec{r}_{\perp i}|) - U_{min} \tag{1.10}$$

where U_{min} is so chosen that $U(\vec{r}_\perp)$ becomes zero at points where potential is a minimum[10].

1.8.1 Axial motion in the continuum model

For an ion of energy E moving at a relatively low angle ψ_{in} to a low index direction the total transverse energy at the point $\vec{r}_\perp$ is given by

$$E_\perp = U(\vec{r}_\perp) + E{\psi_{in}}^2 \tag{1.11}$$

Since the potential energy is independent of velocity, transverse energy is conserved throughout the motion. Hence the ions are always confined to regions where $U(\vec{r}_\perp) \leq E_\perp$. Lindhard had shown that once statistical equilibrium is attained there is an equal probability of finding the ion anywhere in the accessible area $A(E_\perp)$, so that the probability may be written as

$$P(E_\perp, \vec{r}_\perp) = \frac{1}{A(E_\perp)} \qquad U(\vec{r}_\perp) \leq E_\perp \tag{1.12}$$

$$= 0 \qquad U(\vec{r}_\perp) > E_\perp. \tag{1.13}$$

The above equation implies a minimum approach distance $r_{\perp c}$ to a string determined from the equation

$$E_\perp = U(r'_\perp) \equiv U_1(r_{\perp c}) \tag{1.14}$$

where $|r'_\perp| = |\vec{r}_\perp - \vec{r}_{\perp i}| = r_{\perp c}$, $\vec{r}_{\perp i}$ being the position vector of the i^{th} string in the transverse plane. Hence all close encounters with the string for which the impact parameter $q < r_{\perp c}$ is ruled out. This explains the reduction in the yield of close encounter processes for channeling ions. The continuum approximation is valid only for ions with large impact parameters and small transverse angles. The approximation breaks down for a critical distance of approach $r_{\perp crit}$, which defines a critical value E_{crit} for transverse energy and critical angle ψ_{crit} for angle of incidence ψ_{in}. The three values are related by the equation,

$$U(r_{\perp crit}) = E_{crit} = E\psi_{crit}^2 \tag{1.15}$$

where ψ_{crit} is of the order of a characteristic angle ψ_1, written as

$$\psi_1 = \left(\frac{2Z_1Z_2e^2}{Ed}\right)^{\frac{1}{2}} \tag{1.16}$$

When the ion energy is less than a characteristic energy E' defined as

$$E' = \frac{2Z_1Z_2e^2d}{a_T^2} \tag{1.17}$$

ψ_{crit} is of the order of another characteristic angle ψ_2 expressed as

$$\psi_2 = \left\{\frac{Ca_T}{d}\left(\frac{Z_1Z_2e^2}{Ed}\right)^{\frac{1}{2}}\right\}^{\frac{1}{2}} \tag{1.18}$$

1.8.2 Continuum approximation for planes

If the ion is moving in an inter planar channel of width d_p, with an angle of incidence ϕ with respect to an atomic plane, and is trapped between two adjacent X-Z planes, the continuum potential at a distance y from an X-Z plane is given by

$$Y_1(y) = (Nd_p)\int_0^{+\infty} 2\pi r dr V\left(\sqrt{r^2+y^2}\right) \tag{1.19}$$

where (Nd_p) is the number of atoms per unit area in the plane. The Lindhard standard potential yields the following expression for $Y_1(y)$ by substituting eqn(1.3) into eqn(1.19)

$$Y_1(y) = 2\pi Z_1 Z_2 e^2 (Nd_p) a_T \left[\left\{ \left(\frac{y}{a_T} \right)^2 + C^2 \right\}^{\frac{1}{2}} - \left(\frac{y}{a_T} \right) \right] \tag{1.20}$$

The transverse energy associated with the motion is given by

$$E_\perp = Y(y) + E\phi^2 \tag{1.21}$$

There is a finite potential barrier $E_B = Y(0)$ which an ion must overcome in order to penetrate through one atomic plane into the next channel. The transverse angle necessary to overcome this barrier is

$$\phi_r = \left(\frac{E_B}{E} \right)^{\frac{1}{2}} = \left(\frac{2\pi Z_1 Z_2 e^2 (Nd_p) a_T}{E} \right)^{\frac{1}{2}} \tag{1.22}$$

All ions with transverse energy $E_\perp < E_B$ will be trapped within a single inter-planar channel. ϕ_r is thus a measure of the critical angle ϕ_{crit} for planar channeling.

1.9 Classification of Trajectories

Consider an ion beam incident on a crystal at an angle ψ_{in} with respect to a channel axis and transverse energy $E_\perp$ given by eqn(1.11). Ions with $E_\perp < E_{crit}$ will become channeled, while those with $E_\perp >> E_{crit}$ freely penetrate the continuum strings and they are said to describe random trajectories(figure(1-11)).

Ions with $E_\perp$ less than a certain value E_0 can be trapped within the central region of a single channel. These particles execute a helical orbit as they penetrate through the crystal. Such ions are said to be proper or hyper channeled(figure(1.3)). Ions with $E_\perp$ of the order of E_0 are quasi-hyper channeled(figure(1-12)). They are trapped within a set of atomic rows for sometime, then escapes to get trapped in another set of atomic rows.

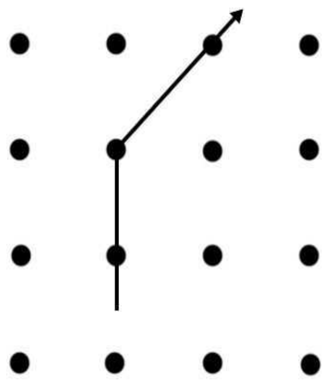

Figure 1.11: Random trajectories

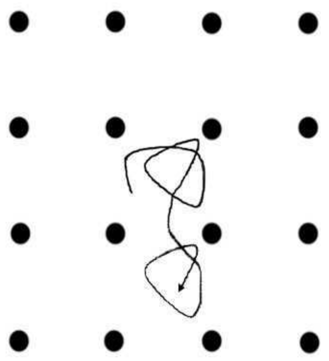

Figure 1.12: quasi-hyper channeled trajectories

Ions with $E_\perp > E_0$ and less than E_{crit} are called axially channeled(figure(1.2)). Another class of ions with $E_\perp$ greater than $E{\psi_{crit}}^2$ but less than $4E{\psi_1}^2$ are called quasi-axial channeled(figure(1-13)). Even though they sometimes exhibit channeling properties, they are capable of penetrating continuum strings.

If $E_\perp$ is less than $E{\phi_r}^2$, ions become planar channeled(figure(1.1)). For $E_\perp$ between $0.25{\phi_r}^2$ and $2.25{\phi_r}^2$ ions become quasi-planar channeled. They are trapped between a pair of atomic planes but soon escape cutting through several atomic planes(figure(1-14)) to be trapped again between another set of planes.

By eqn(1.15), which defines E_{crit}, channeled ions cannot approach

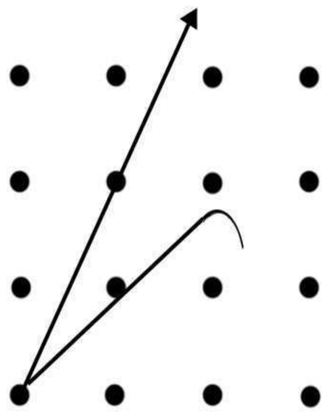
Figure 1.13: Quasi-axial trajectories

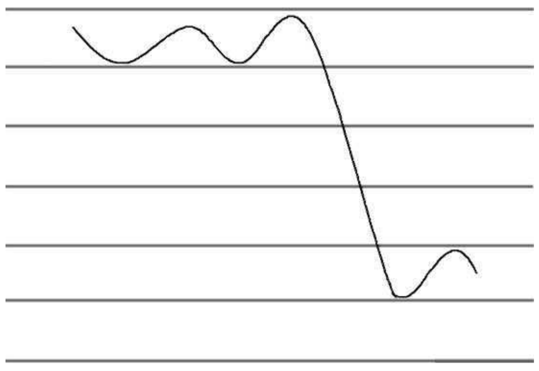
Figure 1.14: Quasi-planar trajectories

the continuum strings to a distance less than $r_{\perp crit}$. This would mean that channeled ions with $\psi < \psi_{crit}$ would not take part in close encounter processes. Hence in the measurement of the yield of these processes, there should be contributions only in the region of $\psi > \psi_{crit}$ thus causing a dip in the channeling direction[11]. The chances of close encounters are more for quasi-channeled ions than for random ions, resulting in a characteristic shoulder on either side of channeling dip. Higher than random value of χ_{max} in the figure(1-7) is attributed to the refractive effect of quasi-channeled ions as shown in the figure(1-15). As the ions cut across transverse planes or atomic rows the transverse energy approaches a minimum value resulting in an increased *time spend* on the planes or the rows. This gives the ions a greater probability for close encounter interactions with the atomic nuclei, thus resulting in a higher than random yield.

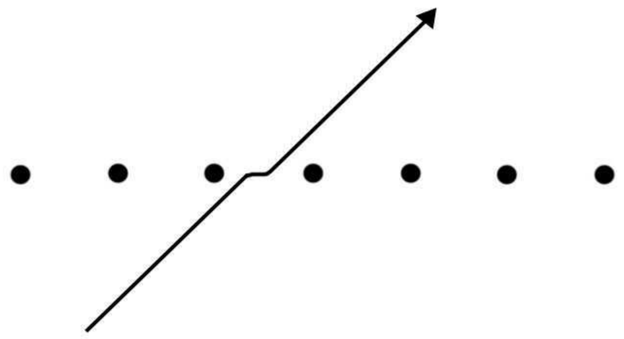

Figure 1.15: refractive effect

1.10 Computer Simulations of Channeling

Computer simulation is a major tool for studying the interactions of ions with solids. There are two main objectives of computer simulations. Firstly, we have the direct simulation of experiments, where a theoretical analogue is produced for comparison with experiments. In the second category, computer simulation is used as an ideal experiment on which to test and develop analytical models. To simulate the passage of an ion through a crystal lattice on a computer, the three dimensional structure of the crystal lattice is stored in the computer memory, together with the interaction forces which bind the atoms together. Various computer simulation models are discussed briefly in the following sessions.

1.10.1 Binary collision model

Earlier analytical methods were based on the binary collision model[12],[13]. In this model the ion is assumed to interact with only the nearest atom. Moliere potential given by eqn(1.5) is taken as the interaction potential. Thermal vibrations are simulated by giving each lattice atom a displacement with coordinates chosen at random from a Gaussian distribution. This treatment is valid provided the energy is sufficiently high and scattering angles are small. One of the shortcomings of this theory is that it does not consider the interaction of the ion with a large number of nuclei in the neighbouring rows and planes, although it is this aspect which leads to the characteristic features of channeling.

1.10.2 N-body model

The basic idea of this model is to set up a numerical crystallite whose atoms interact through a potential energy function. A disturbance is introduced involving the sudden displacement of an atom of the crystallite with a substantial kinetic energy. The equations of motion are integrated until the added energy is sufficiently dissipated. In addition to the forces from the inter atomic potential, external forces may be applied to atoms in the crystallite boundary. In general the field at any point is obtained by considering the ion atom interaction using an appropriate binary potential with each of the neighbouring atoms in the crystal. This molecular dynamics model was first applied to radiation damage studies by Vineyard et al[14]. In the seventies and early eighties due to low computer speeds N-body model consumed very large computation time. Therefore this model was not preferred for most calculations.

1.10.3 Hybrid model

A Monte Carlo program for the calculation of channeling phenomena is described in this model. Smulders and Boerma[15] tried to combine the advantages of binary and continuum model, by considering the interaction with the nearest atom using binary collision model for small distances and with the surrounding atomic rows using continuum model for large distances thus reducing the computer time.

1.11 History and Present Work

The channeling effect was discovered theoretically by a computer simulation in 1962. This discovery by Robinson and Oen[16], of ion channeling and its effects in crystals greatly improved the understanding of the interaction of ion beams with solids and has since been used for characterizing, analysing and processing materials. The actual existence was later demonstrated by further experimental work with mono crystalline targets[2],[3]. These discoveries marked the beginning of a period in which, interest in the channeling effect grew rapidly. Lehmann and Leibfried[17] made an analytic calculation of the trajectories for low energy ion channeling. Further experimental evidence on channeling effect was given by Nelson and Thompson[1], Erginsoy[18] and others. A topic of much interest is the channeling of particles at relativistic velocities[19],[20]. The channeling phenomenon, theory, wide variety of experiments and its various applications have been described in detail in several review articles of Morgan[10], Gemmel[21], Beloshitsky[22] and several others.

With the availability of high speed digital computers, it has become possible to analyze experimental results using Particle Trajectory Approximation(PTA) developed by Ellison et al[23]. Here an N-particle beam is generated in a computer and the trajectory of each particle within the crystal is computed by solving the equation of motion

$$M\frac{d^2r}{dt^2} = -\frac{\partial U(\vec{r_\perp})}{\partial r} \tag{1.23}$$

Where M is the mass of the ion particle and $U(\vec{r_\perp})$ is the ion-crystal potential at any point $\vec{r}$ within the crystal, given by Lindhard potential(eqn (1.9)). Trajectories were computed upto a desired depth, which is usually the thickness of the crystal on which a transmission experiment is performed. The measured physical quantities like transverse momentum and energy, of the emerging particles is compared with the value determined from the exit position and momentum coordinates of the computed trajectories. The computed quantity is comparable with the measured value in the limit of large N. This is the principle of PTA. Ellison et al[23] used it successfully to simulate the *doughnut spectrum* to give the transverse momentum distribution of emerging particles. Their experiment showed an inherent property of the differential equation(1.23) namely that the equation is highly unstable; that is the two particles which start from two close points do not remain close together, for long, instead they quickly diverge away from each other. The non-linearity of the equation of motion and the instability of its solution suggest the possibility of chaos in channeling.

Henry Poincare the famous mathematician was the first who observed rapidly diverging trajectories in many dynamical systems. Though the discovery was made in the early 19th century, extensive studies of the behaviour as a distinct phenomenon called *chaos* was done only after the advent of digital computers in the seventies. The phenomenon of chaos and its characteristics are discussed in chapter 2. Chapter 3 considers the investigation of chaos in channeling. Chapter 4 deals with the simulation of protons channeled in silicon, the existence of chaos in the system and it concludes with the salient findings of our present study in channeling.

References

[1] Nelson R. S. and M. W. Thompson, Phil. Mag, **8**, 167(1963)

[2] Lutz H.O. and R. Sizmann, Phys. Lett, **5**, 113(1963)

[3] Piercy G. R., F. Brown, J. A. Davies, and M.Mc Cargo, Phys. Rev. Lett, **10**, 399(1963)

[4] Andersen J. U., Davies J. A., Nielsen K. O., and Andersen S. L., Nuc.Instrum. and Meth, **38**, 210(1964)

[5] Domeij B., and Bjorqvist K., Phys..Lett, **14**,.127(1965)

[6] Lindhard J., Nielsen V. and Scharff, M. Dansk. Vid. Selsk, Mat. Fys. Medd, **36**, 10(1968)

[7] Moliere G., Zeits. fur. Nat, **2a**, 133(1947)

[8] Bohr N., Dansk, Vid. Selsk, Mat. Fys. Medd **18**, 8(1948)

[9] lindard J., Dansk, Vid. Selsk, Mat. Fys. Medd **34**, 14(1965)

[10] D. V. Morgan, Channelling: Theory, observation and applications(John Wiley and Sons, 1973)

[11] Deepak N. K., Rajasekharan K. and Neelakandan K., Pramana J. Phys, **54**, 845(2000)

[12] Muller M., Nucl. Instr. Meth, **213**, 453(1983)

[13] Muller M., Nucl. Instr. Meth, **B17**, 141(1986)

[14] J. B. Gibson, A. N. Goland, M. Milgram and G. H. Vineyard, Phys. Rev, **120**, 1229(1960)

[15] P. J. M. Smulders and D. O. Boerma, Nucl. Instrum. Methods. B, **29**, 471(1987)

[16] Robinson M. T. and O. S. Oen, Phys. Rev, **132**, 2385(1965)

[17] Lehmann C. and G. Leibfried, J. Appl. Phys, **34**, 2821(1963)

[18] Erginsoy C., H. E. Wegner and W. M. Gibson Phys. Rev. Lett, **13**, 530(1964)

[19] Andersen J. U., Bonderup E. and Pontel R. H., Ann. Rev. Nuc. sci, **33**, 453(1983)

[20] Beloshitsky V. V., Komarov F. F., Phys. Rep, **93**, 117(1982)

[21] D. S. Gemmell., Rev. Mod. Phys, **46**, 129(1974)

[22] Beloshitsky V. V., Komarov F. F. and Kumakhov M. A., Phys. Rep, **139**, 293(1986)

[23] Ellison J. A., Chui S. T. and Gibson W. M., Phys. Rev. **B18**, 5963(1978)

Chapter 2

Chaos

2.1 Introduction

Chaos is the term attributed to the irregular and unpredictable evolution of many nonlinear dynamical systems. A central characteristic feature of such a system is that, a small initial difference in the state of the system grows exponentially with time and after a very short time its history as well as future becomes indeterminable. For a non chaotic system on the other hand, a small initial difference grows only linearly. In spite of its unpredictability, a chaotic system is governed by deterministic equations. But unlike a stochastic system subject to random external forces, the irregularity of a chaotic system is part of the intrinsic dynamics of the system.

In depth study of chaos has shown that two necessary conditions for a system described by a set of first order differential equations to be chaotic are that (i) the system has at least three dynamical variables and (ii) the equation of motion has one nonlinear term coupling several variables. The nonlinearity condition makes the equation almost unsolvable analytically and hence necessitates the use of computers, to obtain numerical solutions.

Therefore extensive study of nonlinear dynamics and chaos were done only in the last three decades. This has lead to identify a number of dynamical systems exhibiting chaos. The chaotic behaviour has been observed in mechanical oscillators, electrical circuits, nonlinear optical systems, chemical reactions, heated fluids and many other systems[1-5]. A large number of experimental work were also started during this period in various fields of science[6-8]. Here we emphasize on two types of dynamical systems. A discrete time dynamical system called maps, takes the current state of the system as input and updates the situation by producing a new state as output. The other important type of dynamical system is essentially the limit of discrete systems with smaller and smaller updating times. These continuous time dynamical systems are described by a set of differential equations.

2.2 Feature of Chaos

The time evolution of any system is represented as a *phase trajectory* in the *phase space* of the system(figure(2.1)). The phase space is a mathematical

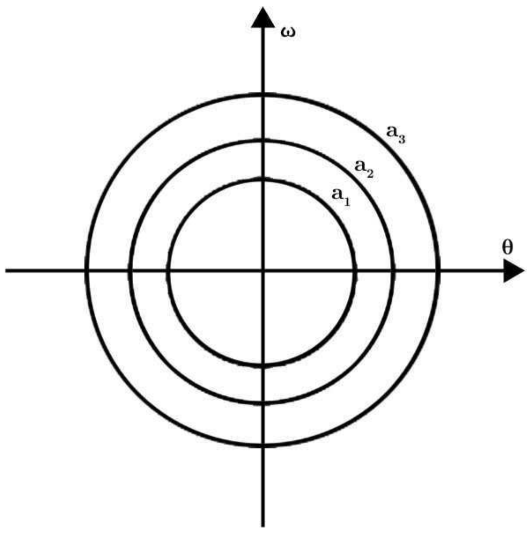

Figure 2.1: Phase diagram of the linear pendulum with angular velocity ω and angular displacement θ as axes. Each value of a_i yield a closed orbit of fixed energy called phase trajectory.

space with orthogonal coordinate directions representing each of the variables, required to specify the instantaneous state of the system. Thus for a particle moving in one dimension the phase space is a two dimensional plane made up of position and velocity directions or their equivalents like position and momentum directions. If one dynamic variable is unbounded but periodic, the phase diagram can also be made periodic, by defining appropriate boundary conditions. The values of the phase coordinates at any instant defines a point in its phase space. Using the equations of the system the trajectory passing through the point can be obtained by evaluating its past and its future, provided the system is deterministic. If the trajectory is crossed the past and the future become ambiguous and the system is said to be indeterministic(figure(2.2)). This is one central feature of a chaotic system.

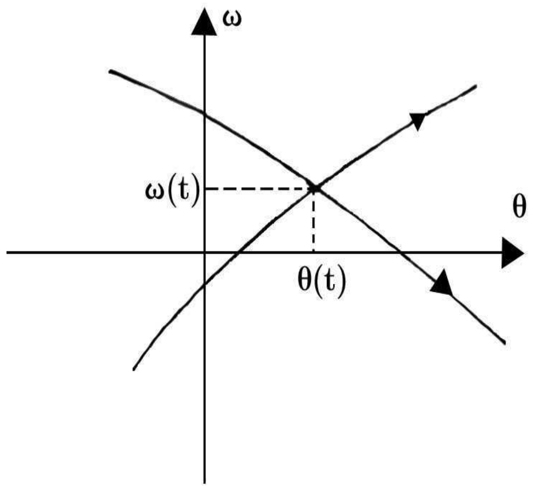

Figure 2.2: The non crossing property of phase trajectories. Crossing of trajectories violates uniqueness of trajectories in a deterministic dynamical system.

Physical systems can be conservative or dissipative. A set of nearby points can be connected together to create an area in two dimensional phase space or a volume in a multidimensional phase space. If the system is conservative neighbouring points maintain the same distance in phase space, resulting in the preservation of area or volume generated by the points(figure(2.3)).

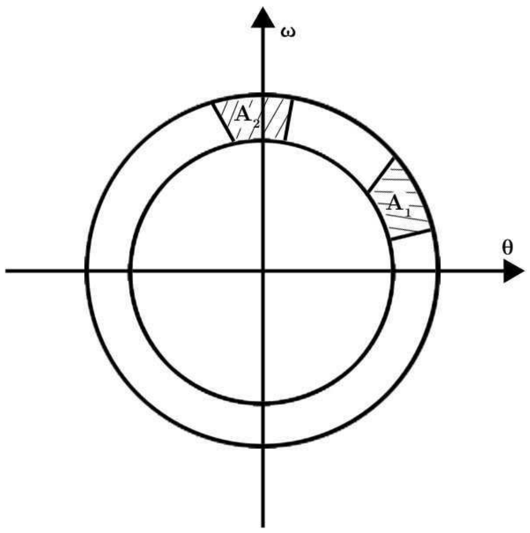

Figure 2.3: Preservation of phase space area.

Some dynamical systems follow the Hamilton's equations of motions. These systems preserve area or volume in phase space and hence are conservative systems. Such systems are called Hamiltonian systems. For dissipative systems the phase space area or volume can shrink and ultimately go to a point called an *attractor*. If the point attractor is an equilibrium state of the system, it is called a $focus$(figure(2.4a)). A focal point is sum of the two critical points in a phase space, the other being a saddle point. For a saddle point there are some stable trajectories which go towards the point and some unstable trajectories which go away from the point(figure(2.4b)).

Some trajectories are closed and stable attracting neighbouring

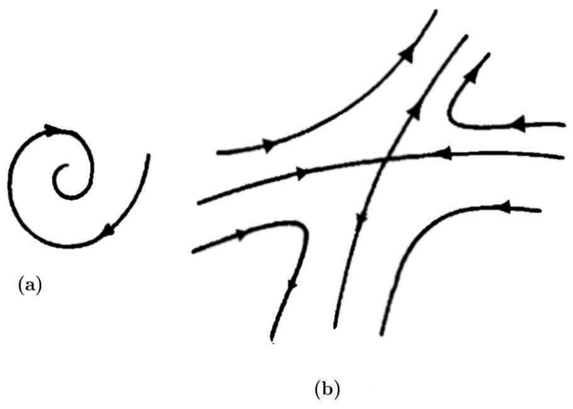

Figure 2.4: Critical points in phase space (a)focal point (b)saddle point.

trajectories. Such trajectories are called *limit cycle*. A phase space can have regions made up of points from where trajectories will eventually converge on an attractor:-a focal point or limit cycle. Such regions are called basins of attraction. One region can sometimes be separated from another by a curve called *separatrix*(figure(2.5)).

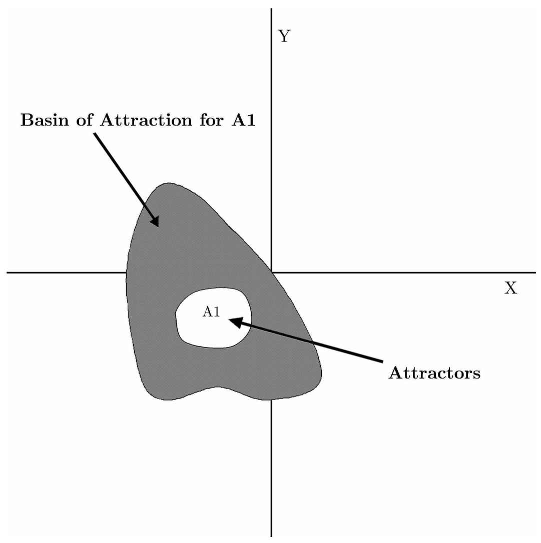

Figure 2.5: A sketch of attractor A_1 and basins of attraction in state space. The line bounding the basin of attraction forms a separatrix.

A phase space trajectory can be viewed stroboscopically in the sense that the system is viewed at regular intervals and the state of the system is marked as points in the phase space diagram. This strobe diagram is called a Poincare section. We give below some examples of well studied chaotic systems.

2.3 Chaos in Dynamical Systems:

2.3.1 Lorenz model

This is a highly simplified model of a convecting fluid introduced by Edward Lorenz[9] to describe convection in the atmosphere. Lorenz demonstrated that even a very simple set of equations may have solutions whose behaviour is essentially unpredictable. In simple physical terms this model treats the fluid system as a layer that is heated at the bottom and cooled at the top. The fluid motion and the resulting temperature differences can be expressed in terms of three variables X(t), Y(t) and Z(t). X is related to the time dependence of fluid stream function, and its derivative with respect to the spatial variables gives the components of the fluid flow velocity. The variables Y and Z are related to the time dependence of the temperature deviations. Using these variables Lorenz model is written as three coupled differential equations,

$$\frac{dX}{dt} = \sigma(Y - X) \tag{2.1}$$

$$\frac{dY}{dt} = X(\rho - Z) - Y \tag{2.2}$$

$$\frac{dZ}{dt} = XY - \beta Z \tag{2.3}$$

where σ is called the Prandtl number, which is defined to be the ratio of the kinetic viscosity of the fluid to its thermal diffusion coefficient. The Rayleigh number ρ is a dimensionless measure of the temperature difference between the top and bottom of the fluid layer and β is related to the vertical height of the fluid layer to the horizontal size of convection rolls. For a choice of parameters $\sigma = 10$, $\beta = \frac{8}{3}$, Lorenz found numerically that the system behaves chaotically whenever ρ exceeds a critical value=24.74. The phase plot for $\sigma = 10$, $\beta = \frac{8}{3}$ and $\rho = 28$ gives an attractor in the three dimensional XYZ space coordinates. A trajectory can be started from an initial position (X_0, Y_0, Z_0). The trajectory then rotates about two unstable fixed points. On starting another trajectory from an initial position $X_0 + dX, Y_0 + dY, Z_0 + dZ$ close to the original one, it can be seen that the second trajectory evolves separately with the difference between the two trajectories increasing exponentially. This sensitivity to initial conditions is sometimes called $butterfly\ effect$ and is a characteristic feature of chaos. Figure(2.6) shows the trajectory in the XYZ space projected on to a two dimensional XZ plane.

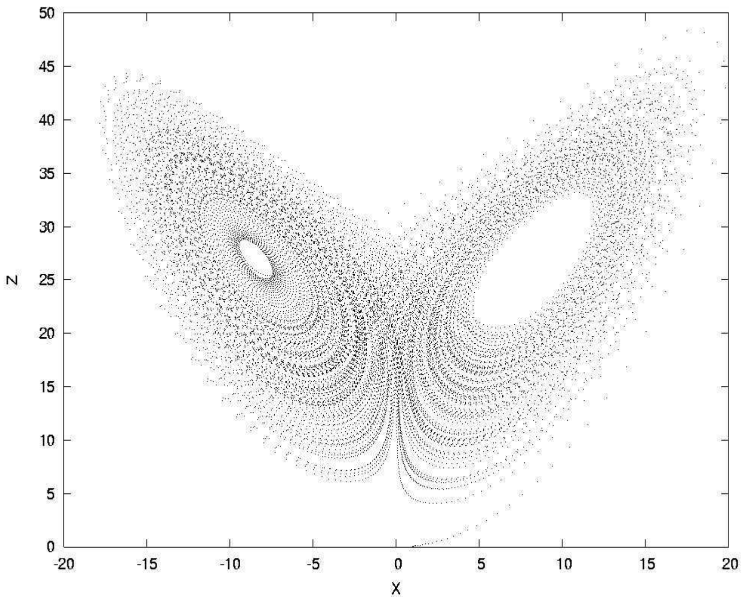

Figure 2.6: Projection of Lorenz attractor in 2-dimensional phase space

The trajectories even though they look simple, they are infact complicated. Even after millions of rotations it has been seen that the system never repeats. If one magnifies the trajectories one can see that, the structure even at the minute scale retains the characteristic features of the gross structure. The resultant path is sometimes compared with a sheet of *pastry dough*, being repeatedly stretched and folded making the structure *self similar*. Geometrical objects possesing this property have non integer dimensions and hence are called fractals. Attractors that are fractals are called *strange attractors*. Chaos always result in strange attractor, but not all strange attractors are chaotic.

2.3.2 Logistic maps

In many cases maps are used as models for physical systems even if the underlying equations are unknown. Maps arise in many disciplines such as biology, engineering, economics and physics. One such map showing chaos is the logistic map[10] given by,

$$x_{n+1} = rx_n(1 - x_n) \tag{2.4}$$

with x in the range [0,1] and $0 < r < 4$. For a given value of r, eqn(2.4) is iterated from a starting x_0, and after a few hundred iterations x_{n+1} reaches a stable fixed point. The process is repeated for other values of r. Figure(2.7) gives a plot of stable fixed points against the values of r. It can be seen that up to r=3, there is just one stable fixed point. At r=3 there is a pitchfork bifurcation resulting in a two cycle. This continues up to 3.4495. With r slightly larger than 3.54, the second bifurcation takes place and so on. At r=3.57 chaos sets in. Most values beyond 3.57 exhibit chaotic behavior , but around 3.82 period doubling cascade occur again.

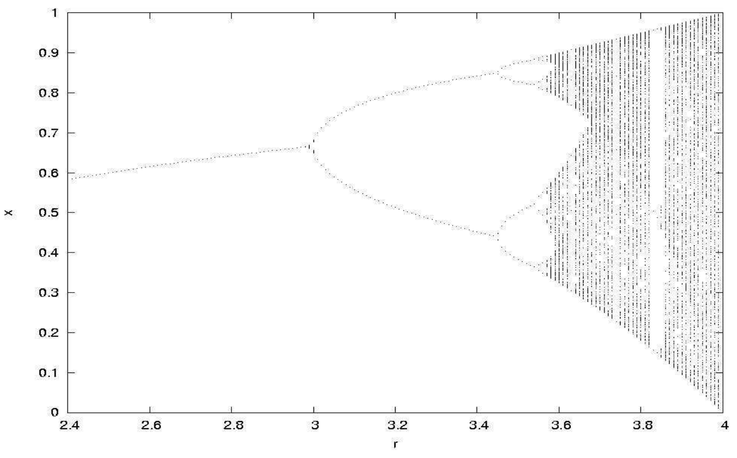

Figure 2.7: Bifurcation diagram for logistic map

The self similarity is evident from the figure. The discrete dynamical system described by the eqn(2.4) is characteristic of many one dimensional systems which exist in nature.

2.3.3 Henon map

To explore the behaviour of two-dimensional maps, let us consider a map function called the Henon map, which has become a classic in the literature of non linear dynamics. The Henon map function is essentially a two-dimensional extension of the one-dimensional quadratic map. This map function was first discussed by Henon[11] and is given by

$$x_{n+1} = y_n + 1 - ax_n^2 \tag{2.5}$$

$$y_{n+1} = bx_n \tag{2.6}$$

where a and b are (positive) bifurcation parameters. The parameter b is a measure of the rate of area contraction (dissipation). The Henon map is the most general 2-D quadratic map with the property that the contraction is independent of x and y.

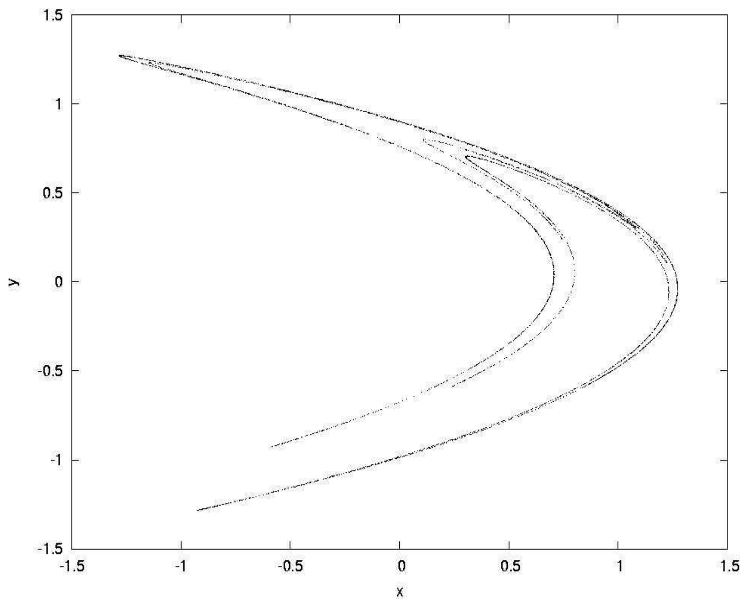

Figure 2.8: Henon attractor for a=1.4 and b=0.3

When b = 0, the Henon map reduces to the quadratic map. Bounded solutions exist for the Henon map over a range of a and b values, some yielding chaotic solutions. The map has been well studied for $a = 1.4$ and $b = 0.3$, where the attractor is strange and chaotic. Figure(2.8) shows the attractor of the Henon map in x-y space for $a = 1.4$ and $b = 0.3$.

2.4 Hamiltonian Systems

Hamiltonian systems are a class of dynamical systems without any dissipation. For such a system we expect that volume of initial conditions would remain constant for all time and there would exist no attractors for the trajectories. Hamiltonian systems exist in various fields such as astronomy[12], superconductivity[13], optics[14] and particle accelerator dynamics[15]. In the Hamiltonian system with N degrees of freedom, the dynamics is completely specified by a simple function, the Hamiltonian $H(q, p)$ where q and p are the canonical coordinates and momenta respectively. Then the trajectories $p(t)$, $q(t)$ in $2N$ dimensional phase space can be generated by solving the Hamilton's equation of motion,

$$\frac{dq_i}{dt} = \frac{\partial H(q,p)}{\partial p_i} \tag{2.7}$$

$$\frac{dp_i}{dt} = -\frac{\partial H(q,p)}{\partial q_i} \tag{2.8}$$

In many cases the Hamiltonian does not explicitly depend on time and the energy $E = H(q,p)$ is conserved along the trajectory. One of the basic properties of Hamilton's equations is that they preserve $2N$ dimensional volume in phase space. As a consequence of this result Hamiltonian systems do not have attractors in the usual sense.

2.5 Integrable and Non Integrable Systems

For a special class of Hamiltonian systems, there are as many constants of the motion as there are degrees of freedom. Such systems are called integrable systems[16]. In most cases, the constants of motion are not the ps in terms of which we initially wrote the Hamiltonian. The constants of the motion, however, can always be expressed as functions of the original qs and ps. The constants of motion are usually called action variables and are commonly written as $J_i(q,p), i = 1, 2, ..., N$. For an integrable Hamiltonian system, the phase space trajectories are confined to an N-dimensional surface in phase space. Associated with each $J_i(q,p)$ there is another variable labeled $\theta_i(q,p)$, called the angle variable. The J_i and θ_i are chosen so that Hamilton's equations, expressed in terms of J_i and θ_i have the same mathematical form as the original Hamilton's equations expressed in terms of the qs and ps.

$$\dot{\theta}_i = \frac{\partial H(\theta,J)}{\partial J_i} \tag{2.9}$$

$$\dot{J}_i = -\frac{\partial H(\theta, J)}{\partial \theta_i} \tag{2.10}$$

If the eqns(2.9) and (2.10) are satisfied, the variables (θ, J) are related to the variables (q, p) by a canonical transformation. If the Hamiltonian depends only on J_i and not on θ_i for all $i = 1, 2, 3, ...N$, we have

$$\dot{J}_i = 0 \tag{2.11}$$

and the J_i are the N constants of the motion. A Hamiltonian system that satisfies the eqns(2.9),(2.10),(2.11) is called an integrable system. If the number of constants of motion is less than the degrees of freedom, the system is non integrable. The given list will tell us what kinds of Hamiltonian systems are integrable[17].

1. All Hamiltonian systems with one degree of freedom and for which H is an infinitely differentiable function of q and p, are integrable and the corresponding action J satisfies $H = \omega J$, where $\omega = \frac{\partial H}{\partial J}$.

2. All Hamiltonian systems for which Hamilton's equations are linear in q and p are integrable.

3. All Hamiltonian systems with nonlinear Hamilton's equations that can be separated into uncoupled systems of one degree of freedom are integrable.

For an integrable system, the trajectory in phase space will remain on the constant energy surface. The idea of transition from integrability to non integrability is given by KAM theorem[18]. Since the behaviour of an integrable Hamiltonian system is always periodic or quasi periodic, an integrable system cannot display chaotic behavior. Much of the literature on the chaotic behaviour of Hamiltonian systems has focused on systems that are, in some sense, just slightly non integrable. A number of non integrable Hamiltonian systems have been identified and studied for their chaotic behavior. The standard map and kicked oscillator are some of the well studied chaotic systems.

2.6 The Standard Map

Many of the theoretical results for universal behaviour of non integrable Hamiltonian systems have come from the study of area preserving iterated map functions. In this section we will discuss the standard map, first introduced by B. V. Chirikov[19]. The Chirikov standard map function is usually written as a function two variables, r and θ, which can be interpreted as the polar coordinates of a trajectory intersection point on a two-dimensional Poincare plane:

$$r_{n+1} = r_n - \frac{K}{2\pi} sin 2\pi\theta_n mod 1 \tag{2.12}$$

$$\Theta_{n+1} = \theta_n + r_{n+1} mod 1 \tag{2.13}$$

For the standard map, the angle θ and the variable r are defined to be in the range[0,1]. K is a positive nonlinearity parameter. It is usual to replace the variable r_n with the variable J_n to use the action angle notation. When K=0, the system will be integrable and the trajectory can be put on a torus and the corresponding phase portrait will be a closed curve. But when $K > 0$, the system becomes non integrable and the phase portrait will be complicated. To illustrate the behaviour of the trajectories of the standard map, we have plotted the trajectory points generated with K=0.5(fig(2.9)) by taking several initial conditions. For each initial point, the standard map has been iterated 200 times and each of the resulting trajectory points has been plotted in the θ-J plane. If the initial condition yields a chaotic orbit,they will wander throughout an area. For small perturbation(K=0.5) there are many KAM tori. When K is increased more and more, the non linearity

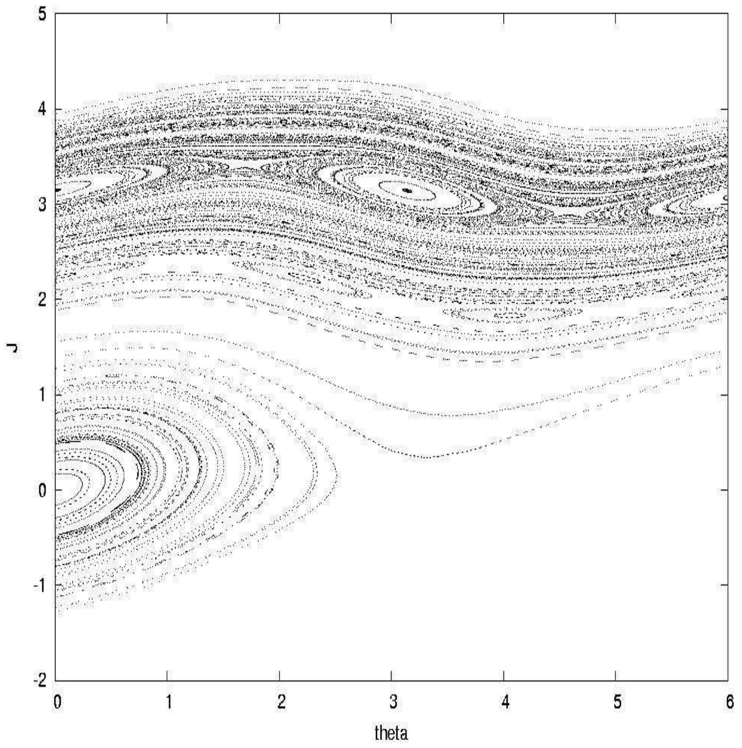

Figure 2.9: The phase portrait of the standard map with K=0.5

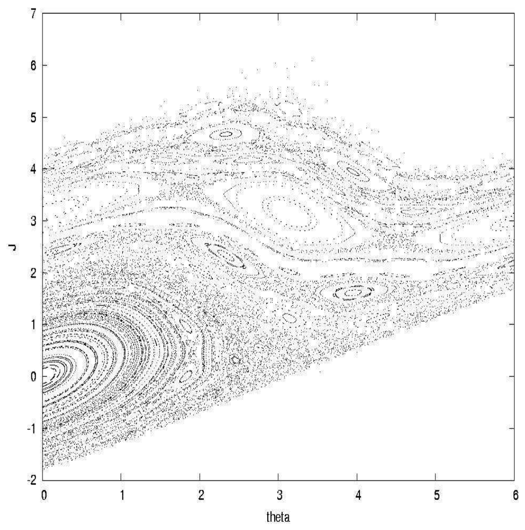

Figure 2.10: Phase portrait with K=1

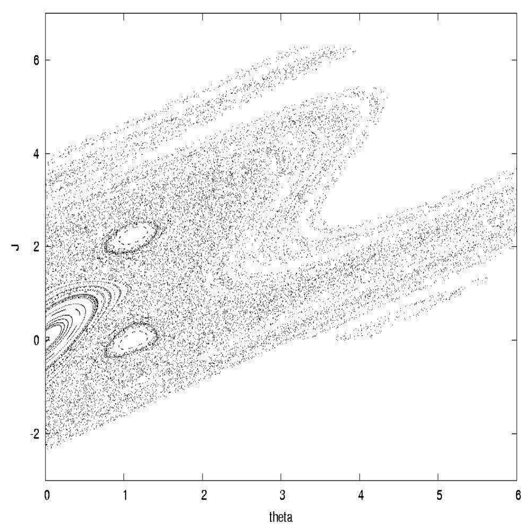

Figure 2.11: Phase portrait with K=2.5

becomes stronger and KAM tori starts disappearing. As the non-linearity increases further, all the tori are eventually destroyed and a single trajectory will wander over nearly the entire allowed region of phase space. Phase plot for K=1 and K=2.5 are given in fig(2.10) and (2.11).

2.7 Quantifying Chaos

Quantifying chaos will help us to separate chaos from noise or randomness in the system. Quantifiers can also sometimes give an estimate of the number of degrees of freedom for the system. Another reason for quantifying chaos is that they may help us sort systems into universality classes. Changes in the quantifiers may be linked to important changes in the dynamical behaviour of the system. In characterizing chaos quantitatively we make use of two different, but related type of description. The first type emphasizes the dynamics(time dependence) of chaotic behavior. The familiar Lyapunov exponent is an example of this type of description. Various kinds of entropy tell us how the system evolves in time and what happens to the nearby trajectories as time goes on. The second type of quantifier emphasizes the geometric nature of trajectories in state space. When the system evolves for a long time, we examine the geometry of the resulting trajectories in phase space. Fractal dimensions and correlation dimensions are quantities which come under this category and they characterise the chaotic attractors.

2.7.1 Lyapunov exponent

Consider a dynamical system described by a set of ordinary differential equations. For a one dimensional state space let x_0 be one initial point and x a nearby initial point. The trajectories that arises from x_0 and x are represented by $x_0(t)$ and $x(t)$ respectively. Let $s = x(t) - x_0(t)$ be the distance between two trajectories. The time dependent equation is assumed to be

$$\dot{x}(t) = f(x) \tag{2.14}$$

since we assume that x is close to x_0, we can use a Taylor series expansion to write

$$f(x) = f(x_0) + \frac{df(x)}{dx} |_{x_0} (x - x_0) + ... \tag{2.15}$$

The rate of change of distance between the two trajectories is given by

$$\begin{aligned} \dot{s} &= \dot{x} - \dot{x_0} \\ &= f(x) - f(x_0) \\ &= \frac{df(x)}{dx} |_{x_0} (x - x_0) = s\frac{df(x)}{dx} |_{x_0} \end{aligned} \tag{2.16}$$

where higher derivative terms in Taylor series expansion of $f(x)$ are neglected. Since we expect the distance to change exponentially in time, we introduce the Lyapunov exponent λ as the quantity that satisfies

$$s(t) = s_0 e^{\lambda t} \tag{2.17}$$

Where $s_0 = s(t = 0)$. If we differentiate equation(2.17) with respect to time,

$$\dot{s} = \lambda s_0 e^{\lambda t} = \lambda s \tag{2.18}$$

which gives

$$\lambda = \frac{df(x)}{dx} |_{x_0} \tag{2.19}$$

Thus we see that if λ is positive, the trajectories diverge. In state space with more dimensions we can associate a Lyapunov exponent with the rate of expansion or contraction of trajectories for each of the directions in the state space.

For an iterated map, the divergence or convergence of trajectories is exponential as a function of iteration number. Let x_0 be a point on the attractor and $x_0 + \epsilon$ its neighbour. We then apply the iterated map function n times to each value and consider the difference between these results.

$$d_n = \mid f^n(x_0 + \epsilon) - f^n(x_0) \mid \tag{2.20}$$

If the behavior of the system is chaotic, we expect this distance to grow exponentially with n, so we write

$$\frac{d_n}{\epsilon} = \frac{\mid f^n(x_0 + \epsilon) - f^n(x_0) \mid}{\epsilon} = e^{\lambda}n \tag{2.21}$$

Then

$$\lambda = \frac{1}{n} \ln \frac{\mid f^n(x_0 + \epsilon) - f^n(x_0) \mid}{\epsilon} \tag{2.22}$$

If we now let $\epsilon \longrightarrow 0$, we recognize from elementary calculations that the ratio on the R.H.S of the equation(2.22) is just the definition of f^n with respect to x. We have also seen that, the derivative of f^n can also be written as a product of n derivatives of f^n evaluated at the successive trajectory points $x_0, x_1, x_2, ...$ and so on. Thus we can put the definition of Lyapunov exponent in a more intuitive form

$$\lambda = \frac{1}{n}(\ln \mid f'(x_0) \mid + \ln \mid f'(x_1) \mid +..... + \ln \mid f'(x_n) \mid) \tag{2.23}$$

equation(2.23) tells us that the Lyapunov exponent is just the average of the natural logarithm of the absolute value of the derivatives of the map function, evaluated at the trajectory points. In general we can say that a one dimensional iterated map function has chaotic trajectories for a particular parameter value if the average Lyapunov exponent is positive for that parameter value.

Lyapunov exponents can also be calculated from a one-dimensional time series of data. Consider two points in a space, x_0 and $x_0 + \Delta x_{0,}$ each of which will generate an orbit in that space using some equation or system of equations. These orbits can be thought of as parametric functions of a variable that is something like time. If we use one of the orbits a reference orbit, then the separation between the two orbits will also be a function of time. This separation is also a function of the location of the initial value and has the form $\Delta x(x_0, t)$. In a system with attracting fixed points or attracting periodic points, $\Delta x(x_0, t)$ diminishes asymptotically with time. If a system is unstable, then the orbits diverge exponentially for a while, but eventually settle down. For chaotic points, the function $\Delta x(x_0, t)$ will behave erratically. Then the mean exponential rate of divergence of two initially close orbits defined by the formula

$$\lambda = \lim_{t \to \infty} \frac{1}{t} \ln \frac{|\Delta x(x_0, t)|}{\Delta x_0} \tag{2.24}$$

is chosen as the Lyapunov exponent. It works for discrete as well as for continuous systems.

2.7.2 Significance of Lyapunov exponents

For $\lambda < 0$, the orbit attracts to a stable fixed point(figure 2.12) or stable periodic orbit(figure 2.13). Negative Lyapunov exponents are characteristic of dissipative or non-conservative systems. Such systems exhibit asymptotic stability; the more negative the exponent, the greater the stability.

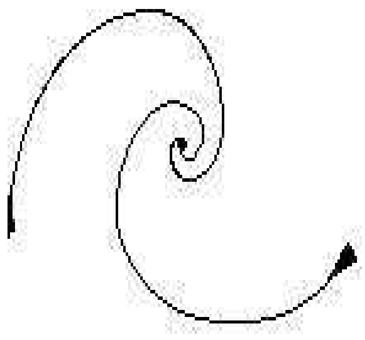

Figure 2.12: dissipative(attracting fixed point), $\lambda < 0$

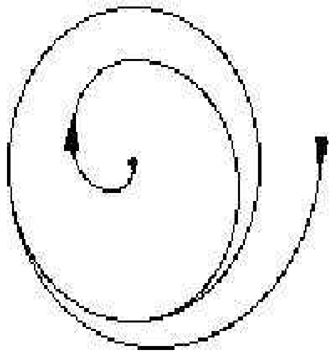

Figure 2.13: dissipative(attracting orbit), $\lambda < 0$

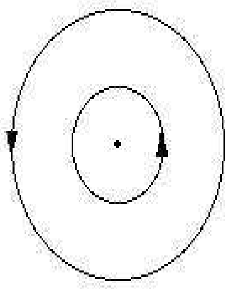

Figure 2.14: conservative(neutral fixed point and orbits), $\lambda = 0$

For $\lambda = 0$, the orbit is a neutral fixed point (figure 2.14). A Lyapunov exponent of zero indicates that the system is in some sort of steady state mode. A physical system with this exponent is conservative. Such systems are said to exhibit Lyapunov stability. For $\lambda > 0$,the orbit is unstable and chaotic. Nearby points, no matter how close, will diverge to any arbitrary separation. These points are said to be unstable.

In the last thirty years several mathematical techniques have been developed for determining the Lyapunov exponent and other quantifying parameters in chaos. This lead to the study of chaos in a wide variety of physical systems. In the next chapter we consider the study of chaos in channeling.

References

[1] Hao Bai-Lin, Chaos, World scientific, Singapore(1985)

[2] Thomas Kapitanaih, Controlling chaos, Academic press(1996)

[3] S. Ushiki, Chaotic dynamical system, World Scientific, Singapore(1992)

[4] M. J. Feigenbaum, Los Almos Science **1**, 4(1980)

[5] J. P. Eckmann., Rev. Mod. Phys, **53**, 643(1981)

[6] M. Giglio, S. Musazzi and U. Perini, Phys. Rev. Lett. **47**, 243(1981)

[7] J. L. Hudson and J. C. Mankin, J.Chem. **74**, 617(1981)

[8] J. Testa, J. Perez and C. Jeffries, Phys. Rev. Lett. **48**, 714(1982)

[9] E. N. Lorenz, J. Atmos. Sci, **20**, 130(1963)

[10] R. May, Nature **261**, 459(1976)

[11] M. Henon, Comm. Math. Phys, **50**, 69(1976)

[12] R. A. Kerr, Science **240**, 986(1988)

[13] C. Reichardt and F. Nori, Phys. Rev. Lett. **82**, 414(1999)

[14] A. Mekis, J. U. Nockel, G. Chen, A. D Stone and R. K. Chang, Phys. Rev. Lett, **75**, 2682(1995)

[15] A. Chao, D. Johnson, S. Peggs et al, Phys. Rev. Lett, **61**, 2752(1988)

[16] R. C. Hilborn, Chaos and Nonlinear Dynamics, Oxford, NewYork(1994)

[17] R. H. G. Helleman, Fundamental problems in statistical mechanics, **5**, 165(1980)

[18] V. I. Arnold, Mathematical methods in classical mechanics, Springer Verlag, New York(1978)

[19] B. V. Chirikov, Physics Reports, **52**, 263(1979)

Chapter 3

Chaos in Channeling

3.1 Previous Work

The existence of chaos in channeling was first demonstrated by Shulga et al[1]. They considered the channeling of relativistic electrons and positrons through silicon crystal in the $\langle 110 \rangle$ direction. The particle interaction with crystal is described by the continuum potential. This conserves the momentum component of particle in channeling direction chosen as the Z axis. This reduces the motion to the two-dimensional X-Y plane transverse to the channel axis. The conservation of the transverse energy($E_\perp$) of the channeled particle gives the equation of motion, *one integral of motion*. The existence of the second integral is investigated by the Poincare cross section technique. For this, the trajectory of a particle is drawn in the four-dimensional phase space($x, \dot{x}, y, \dot{y}$); x and y being two orthogonal position directions in the transverse plane and $\dot{x}$ and $\dot{y}$ their time derivatives. Since transverse energy($E_\perp$) is a constant the trajectory will lie on a three-dimensional surface defined by its value. A Poincare section is generated by plotting the points of intersection of this trajectory with a two-dimensional plane such as y, $\dot{y}$. If the second integral exists, the intersection should give a curve determined by the integral of motion. Motion of the particle will then be regular and quasi-periodic. On the other hand if the second integral does not exist the intersection points are randomly distributed in a region bounded by a separatrix. Motion will be then irregular and chaotic. Shulga et al[1] showed that for both channeled and random particles motion can be chaotic.

Kimball et al[2] assumed the velocity of the particle in the direction of channeling to be a constant. In the transverse plane the particle is assumed to be acted upon by a time dependent force generated by the nearest neighbour atom. Then the equation of motion in the transverse plane can be written as

$$M\frac{d^2r}{dt^2} = F(r,t) \tag{3.1}$$

with

$$F(r,t+\tau) = F(r,t)$$

The transverse force F is periodic with periodicity τ, τ being the time taken by the particle to move one lattice distance in the channel direction. Because of the time dependence of the transverse force, the transverse energy of the particle is no longer conserved. After a series of assumptions the equation of motion is reduced to a set of recursion relations

$$p_{n+1} = p_n - \frac{K}{\pi} sin(\pi q_n) \tag{3.2}$$

$$q_{n+1} = q_n + p_{n+1} \tag{3.3}$$

Where q_n and p_n are generalized transverse position and momentum coordinates. K is a parameter depending on factors like lattice constant and the instantaneous kicks generated by neighbour atoms. The above equations are the standard map equations discussed in section(2.6). The trajectories generated by the recursion relations(eqns(3.2),(3.3)) are plotted as points in the s-p phase space. It is shown that for small values of K, corresponding to channeled trajectories, closed orbits are obtained. For higher values of K, the single orbits breaks up into islands and for still higher values of K the motion becomes chaotic.

Later in the 1990's Ling chen et al[3] demonstrated chaos for the case planar channeling. As described in section(1.8.2), in the case of planar channeling, the motion reduces to a one-dimensional problem, since the velocity components in the channeling direction(z axis) as well as in the direction parallel to the atomic planes and orthogonal to the channel axis(y axis) are conserved. Then the motion in the x direction perpendicular to the atomic planes are described by the Hamiltonian equations

$$\frac{\partial(mv)}{\partial t} = -\frac{\partial H}{\partial x} \tag{3.4}$$

$$\frac{\partial x}{\partial t} = \frac{\partial H}{\partial(mv)} \tag{3.5}$$

where m is the mass of the particle and v its velocity in the x direction. The Hamiltonian H for planar channeled particle is

$$H = \frac{mv^2}{2} + U(x,t) \tag{3.6}$$

The interaction potential $U(x,t)$ is described by the Moliere potential(eqn(1.5)) modulated by a time dependent delta function corresponding to the impulses, from the nearest neighbour atomic plane. The generalized coordinates p and q for the transverse momentum and position respectively can be defined by the equations

$$p = \frac{2Tv}{d_p} \tag{3.7}$$

$$q = \frac{2x}{d_p} \tag{3.8}$$

where d_p is the inter planar distance and T is the time period required for a channeled particle to travel a distance of one periodic unit in the channel direction. The generalized coordinate p is then related to the instantaneous transverse angle ψ by the relation

$$p = \psi \left(\frac{2b}{d_p} \right) \tag{3.9}$$

where b is the lattice constant. Then using suitable approximations the following recursion relations are obtained

$$p_{n+1} = p_n + \frac{2F(q_n)T^2}{(md_p)}$$

where

$$F(q_n) = \frac{-E_0{\psi_p}^2}{a} \Sigma_i \Sigma_j \alpha_i exp \left(\frac{-\beta_i d_p |q_n - q_j|}{2a} \right) \tag{3.10}$$

$$p_{n+1} = p_n + K \Sigma_i \Sigma_j \alpha_i exp \left(\frac{-\beta_i d_p |q_n - q_j|}{2a} \right) \tag{3.11}$$

$$q_{n+1} = q_n + p_{n+1} \tag{3.12}$$

$$K = \frac{2\pi N b^2 Z_1 Z_2 e^2}{E_0 d_p} \tag{3.13}$$

The summation j is for the number of planes considered, N is the atomic density in the channeling plane and K is the parameter depending on the interaction between the channeling particles and the lattice planes. Chen et al[3] considered the channeling of 10keV protons in tungsten crystal between the (100) planes. This results in the K value of 0.427. Choosing the initial value of q to be zero, various values of p in the range $0 < p < 0.4$ are selected to define the initial sets of (p_0, q_0) in the (p, q) phase space. The trajectories for each sets of (p_0, q_0) are then determined by the equations(3.11) and (3.12). It was found that the interior orbits in the phase space corresponding to small values of the incident angle ψ_0 are elliptical in shape. They represent the regular channeled motion. As p increases corresponding to increase in ψ_0 the orbits begin to get deformed and eventually split to form islands. The motion thus become chaotic. On increasing p_0 and hence ψ_0 islands disappear and the near elliptic orbits return. These are indicative of random motion. On varying K Chen et al[3] found that K has to be in the range 0.35 to 1, for chaotic motion.

Chen at al[3] could successfully relate these findings to experimental observations. For this they assumed that the distribution in the transverse momentum of the ions are directly related to the yield for close-encounter process. On plotting the distribution of the transverse momentum of channeled particles along tungsten(100) and silicon(100) planes, they obtained the results comparable to the experimental observation. Their findings show that the characteristic dip in the close-encounter processes is due to well channeled particles. On the other hand chaotic motion produces a yield close to random motion superimposed by a fine structure. The result of channeled and chaotic motion then produces the characteristic dip modulated by a fine structure as observed by Logan et al[4] and Anderson et al[5].

The works of Shulga, Kimball and Chen were of qualitative nature. All of them involved plotting the phase-space trajectory and searching for the existence of open curves which are indicative of chaos. Their results conclusively proves chaos in channeling. The result of Chen et al[3] also shows possible experimental observations of chaotic motion in crystals. Hence it underlies the necessity of further investigation of chaos in channeling. A quantitative estimate of chaos by evaluating the Lyapunov exponents is therefore appropriate.

3.2 Calculation of Lyapunov Spectra

An algorithm for computing the entire Lyapunov spectrum was put forward by Wolf et al[6], which was based on the technique developed independently by Bennetin et al[7], Shimada and Nagashima[8] for determining a complete spectrum from a set of differential equations. The method involves the measurement of the average rate of divergence of neighbouring trajectories on an attractor. Consider a small sphere centered at a point on the attractor and whose surface is made up of points of neighbouring trajectories. As the system evolves in time the sphere is deformed as a result of the different rates of expansion in different directions. For example in the case of a two-dimensional system the sphere becomes an ellipsoid with the principal axis along the directions of expansion and contraction(figure(3.1)).

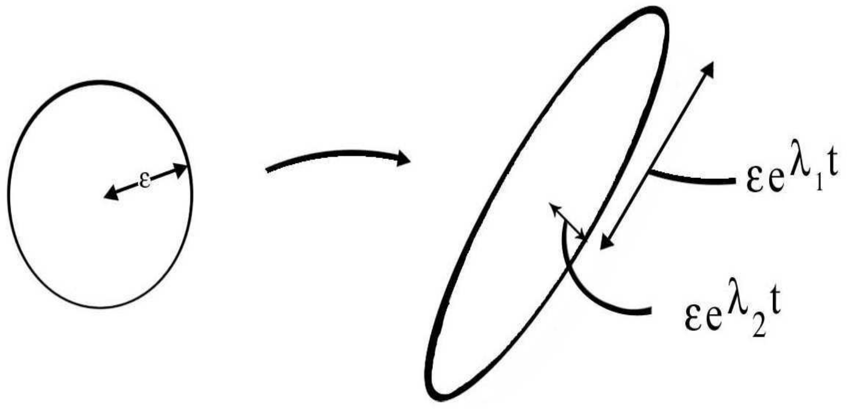

Figure 3.1: Evolution of ellipsoid

Lyapunov exponents are then defined as a rate of expansion along the different principal axis. For the i^{th} principal axis, the corresponding exponent is defined as

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \ln \frac{L_i(t)}{L_i(0)} \tag{3.14}$$

where $L_i(t)$ is the radius of the ellipsoid along the i^{th} principal axis at time t. Thus we have n-Lyapunov exponents defined for an n-dimensional phase space. It is impractical to perform the actual computation in the way suggested by the definition, because the initially close phase points would soon diverge from each other by distances approaching the size of the attractor, and the computation would then fail to capture the local rates of divergence and contraction. Therefore vectors connecting the surface of the ellipsoid to the center must be shrunk periodically or renormalized to ensure that the size of the ellipsoid remains small and that its surface points correspond to trajectories near that of the center point. This method is called Gram-Schmidt reorthonormalization(GSR) and the procedure is illustrated in figure(3.2).

Let the linearized equations of motion act on the initial frame of orthonormal vectors to give a set of vectors $v_1, ...v_n$. Then GSR provides the following orthonormal set $v_1', ...v_n'$.

$$v_1' = \frac{v_1}{\|v_1\|} \tag{3.15}$$

$$v_2' = \frac{v_2 - < v_2 \cdot v_1' > v_1'}{\|v_2 - < v_2 \cdot v_1' > v_1'\|} \tag{3.16}$$

$$\vdots$$

$$v_n' = \frac{v_n - < v_n \cdot v_{n-1}' > v_{n-1}' - .... - < v_n \cdot v_1' > v_1'}{\|v_n - < v_n \cdot v_{n-1}' > v_{n-1}' - ...... - < v_n \cdot v_1' > v_1'\|} \tag{3.17}$$

GSR never affects the direction of the first vector in a system, as this vector

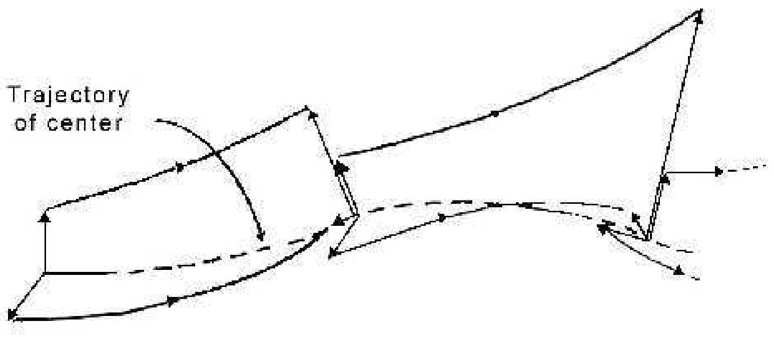

Figure 3.2: Readjustment of the size of vectors along the principal axis

tends to seek out the direction in the tangent space which is most rapidly growing. The second vector has its component along the direction of the first vector removed, and is then normalized. Due to re-normalization the vectors v_1' and v_2' are orthogonal and span the same two dimensional subspace that is most rapidly growing.The area defined by these vectors is proportional to $2^{(\lambda_1+\lambda_2)t}$. The length of the vector v_1 is proportional to $2^{\lambda_1 t}$. Therefore the growth rate of length and area leads to the determination of both the lyapunov exponents for a two-dimensional system. In general for an n-dimensional phase space all the n lyapunov exponents can be determined by monitoring the long term evolution of the n volume.

For a discrete dynamical system defined by map equations

$$x_{n+1} = f(x_n, y_n) \tag{3.18}$$

$$y_{n+1} = g(x_n, y_n) \tag{3.19}$$

where x_n and y_n are the $(n-1)^{th}$th iterate of an arbitrary initial condition x_1,y_1. Let an initial vector be defined as $\begin{pmatrix} \delta x_n \\ \delta y_n \end{pmatrix}$. The linearisation of this map then gives

$$\begin{pmatrix} \delta x_{n+1} \\ \delta y_{n+1} \end{pmatrix} = J_n \begin{pmatrix} \delta x_n \\ \delta y_n \end{pmatrix} \tag{3.20}$$

where the jacobian J_n is

$$J_n = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} \tag{3.21}$$

An orthonormal frame of principal axis vectors such as ((0,1),(1,0)) is evolved by applying the product jacobian to each vector. Thus the current axis vector $\begin{pmatrix} \delta x_n \\ \delta y_n \end{pmatrix}$ may be written as ,

$$\begin{pmatrix} \delta x_n \\ \delta y_n \end{pmatrix} = J_{n-1} \left(J_{n-2} ......J_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \tag{3.22}$$

The magnitude of each current axis vector diverges and the angular separation between the two vectors goes to zero. GSR corresponds to the replacement of each current axis vector. The growth rate of the length of the first vector and the growth rate of the area defined by both vectors is used to compute both Lyapunov exponents.

3.3 Lyapunov Exponent in Planar Channeling

We consider the planar channeling of 10keV protons in Gold between (100) planes. Using the value of $4.078 A^0$ as the lattice constant of gold we get

$2.039A^0$ as the value of inter planar distance d_p. Then equation (3.13) yields the value of $K = 0.7007$. Then according to Chen et al[3] this value comes within the range 0.35 to 1 for chaotic motion. The phase space trajectories are then generated by using the programme *planar.for* given in appendix I . For all trajectories the initial value q_0 is taken to be zero. Then p_0 is varied between 0.1 to 0.4 to produce different trajectories. Starting from initial values (p_0, q_0) subsequent values of p and q are obtained using the map equations((3.11),(3.12)). Subroutine *fcn* accepts input from the main calling programme the current values of p_n and q_n through the variable x in the programme and returns the new values through the variable *xnew*. The summation in equation is performed by considering three pairs of neighbour planes. The map equations are linearized by using the Jacobian

$$J = \begin{pmatrix} 1 & -K\Sigma_i\Sigma_j\alpha_i\beta_i exp\left(\frac{-\beta_i d_p|q_n-q_j|}{2a}\right) \\ 1 & 1-K\Sigma_i\Sigma_j\alpha_i\beta_i exp\left(\frac{-\beta_i d_p|q_n-q_j|}{2a}\right) \end{pmatrix} \tag{3.23}$$

This Jacobian is evaluated by the subroutine *fcn* and the new elements are returned to the calling program. These elements are used by the main programme to obtain the current axis vectors. Renormalizations are done after every three iterations. Lyapunov exponents are evaluated after about thousand renormalization. The main programme outputs the Lyapunov exponents on to a data file and the evolution in phase space by writing the values of *p*,*q* on to another data file *pq.d*. The programme *pqgen.c* reads from the data file to generate the phase space trajectory. There is also an option for viewing the planar motion in the y-z real space. Figure (3.3) shows the trajectories in *pq* phase space.

From figure(3.3), it can be seen that for low values of p, ions are

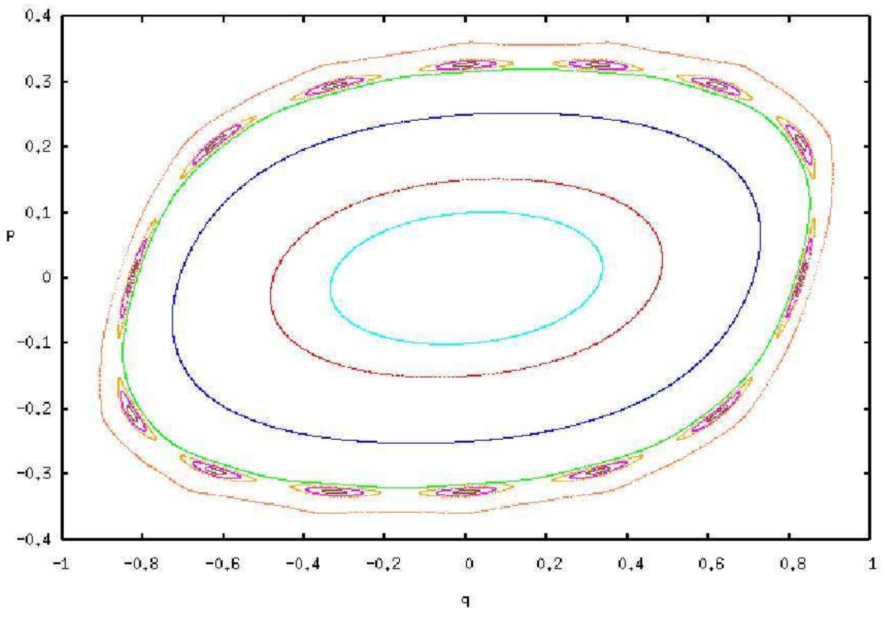

Figure 3.3: phase diagram of 10keV proton channeling along gold(100) plane

well channeled giving near elliptical orbits. From about p=0.315 the closed curve begins to develop a ripple showing the onset of chaos. At p=0.321, the orbits splits to form separate islands. The islands then shrink to form single points at p=0.327. The motion is then no longer chaotic. Increasing p further once again sets in chaos. The points grow to become islands again. They continue to grow until they merge to form double orbits at p=0.335. Throughout this chaotic phase the ions stay well channeled and periodic. Beyond p=0.38, the motion becomes random with no stable orbits in (p, q) plane.

Since planar channeled motion is represented by a two-dimensional phase space there will be two lyapunov exponents. The map equations(3.11) and (3.12) being conservative the sum of Lyapunov exponents should give zero. As expected $planar.for$ gives two lyapunov exponents of the same magnitude with opposite sign. Table(3.1) shows the positive values of lyapunov exponent(λ) for various values of p. The values of λ are reported in the unit of bits/iterations. Even though all trajectories yields one positive value of λ it can be seen that the values are relatively low beyond p=0.375, when the motion becomes random. The obtained values are hence consistent with the interpretations based on the phase space diagram except for the values of p less than 0.315.

Table 3.1: Lyapunov exponents calculated in units of bits per iterations for 10keV protons in Au(100), for various p_0 values

p_0	λ
0	0.004
0.1	0.1895
0.15	0.2585
0.2	0.3206
0.25	0.3894
0.3	0.4703
0.31	0.4887
0.32	0.5119
0.34	0.5358
0.35	0.5541
0.36	0.5739
0.37	0.5982
0.38	0.2276
0.39	0.0388
0.4	0.0117

For the case of silicon with lattice constant $b = 5.431A^0$ the inter planar spacing $d_p = 1.92A^0$ for (100) plane. Then the channeling of 50keV protons gives a K value of 0.023. Calculations then show that all phase trajectories are elliptical without any splitting. Closed curves in phase space are assumed to represent regular, periodic trajectories. But our calculations show that the Lyapunov exponent can be positive, even in such cases, even though the magnitude of the exponent is very small. Table(3.2) shows the positive Lyapunov exponents for the trajectories studied. The very low values obtained shows that the motion is regular, periodic and non-chaotic, consistent with the observation of Chen et al[3] .

Table 3.2: Lyapunov exponents calculated in units of bits per iterations for 50keV protons in Si(100), for various p_0 values

p_0	λ
0.01	0.001
0.02	0.001
0.024	0.0012
0.026	0.002
0.028	0.018
0.03	0.0263
0.032	0.0322
0.034	0.037
0.036	0.0414
0.038	0.0455
0.04	0.049

3.4 Conclusion

Lyapunov exponents have been determined for planar channeled trajectories using the approximate map equations of Chen et al[3]. It is found that the maximum lyapunov exponent is always positive upto a p_0 value of 0.4. But it is significant only when the parametric value K in the equation(3.11) is between 0.35 and 1. Even when the phase trajectories are closed indicative of regular periodic motion the maximum lyapunov exponent is positive though very small in magnitude. The results are recently being presented by Jeena et al[9].

References

[1] N. F. Shulga, Y. L. Bolotin, V. Y. Gonchar and V. I. Truten, Phys. Lett. A, **123**, 357(1987)

[2] J. C. Kimball, G. Petschel and N. Cue, Nucl. Instum. Methods. Phys. Res. B, **33**, 53(1988)

[3] L. Chen, A. E. Kaloyeros, and G. wang, Chaos, **4**, 85(1994)

[4] L. R. Logan, P. J. Schultz, J. A. Davies and T. E. Jackman, Nucl. Instum. Methods. B, **33**, 58(1988)

[5] J. U. Anderson, W. M. Augustyniak and E. Uggerhoj, Phys. Rev. B **3**, 705(1971)

[6] Alan Wolf, J. B. Swift, Harry L. Swinney and John A. Vastano, Physica, **16D**, 285(1985)

[7] G. Bennetin, L. Galgani, A. Giorgilli and J. M. Strelcyn, Meccanica, **15**, 9(1980)

[8] I. Shimada and T. Nagashima, Prog. Theor. Phys, **61**, 1605(1979)

[9] K. Jeena, K. Rajasekharan and K. Neelakandan, *chaos in channeling* Accepted for presentation: Int. conf. on recent developments in non linear dynamics; Feb 2008, Trichy, India.

Chapter 4

Lyapunov Exponents From Time Series Data

The study of chaos in channeling using the map equations is a crude approximation to the real system. This is obvious from the fact that the equations are heavily dependent on the parameter K which can take the same value in a wide variety of crystal systems, by varying the energy of the channeled particle suitably. The map equations are also conservative whereas the real system is not, hence the need arises for a more realistic study of chaos in channeling. The use of many body model described in section (1.10.2) is the most realistic approximation to the ideal system.

4.1 N-body Model

Let an ion be incident on the crystal at a small angle ψ_{in} with respect to the normal to the surface, the crystal itself cut with the normal along a crystal axis called the channel axis. Choosing the Z-axis to be along the channeling direction an X-Y plane transverse to this direction is considered as shown in figure(4.1). If E is the energy of the ion and M its mass, then the velocity of the ion $v = \sqrt{\frac{2E}{M}}$. The component of the velocity in the channeling direction is

$$v_z = vcos\psi_{in} \tag{4.1}$$

While the transverse component of the velocity

$$v_\perp = vsin\psi_{in} \tag{4.2}$$

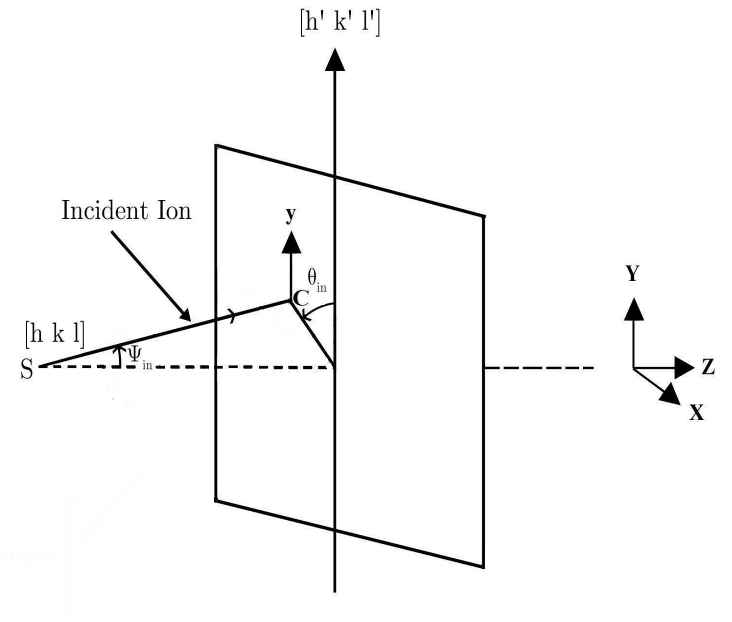

Figure 4.1: Schematic diagram of an experimental arrangement for channeling to study close interaction processes.

The transverse velocity vector may subtend an angle θ_{in} with respect to a reference axis say, [h' k' l']. This angle is called azimuthal angle. The velocity components along the two orthogonal directions X and Y in the transverse plane are

$$v_x = -v_\perp sin\theta_{in} \tag{4.3}$$

and

$$v_y = v_\perp cos\theta_{in} \tag{4.4}$$

The instantaneous position of the ion within the crystal is given by the position vector

$$\vec{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{4.5}$$

where x, y, z are the position co-ordinates of the ion and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are unit vectors along X, Y, Z axes respectively. Then the instantaneous force on the ion is given by the equation

$$M\frac{d^2\vec{r}}{dt^2} = -\left[\frac{\partial V(\vec{r})}{\partial x}\mathbf{i} + \frac{\partial V(\vec{r})}{\partial y}\mathbf{j} + \frac{\partial V(\vec{r})}{\partial z}\mathbf{k}\right] \tag{4.6}$$

Where $V(\vec{r})$ is the potential due to all the nuclei in the crystal. In the N-body model this potential is written as

$$V(\vec{r}) = \sum_{i=1}^{N} V_s(r, r_i) \tag{4.7}$$

where N is the total number of atoms and V_s is the potential due to a single atom at r_i. We used the coulombic potential modified with Lindhard standard potential(eqn(1.3)) to describe the ion atom potential V_s.

Due to thermal energy the nuclei in the crystal oscillates about the equilibrium positions r_i's . To take into account the effect of these oscillations we use an averaging procedure given by Andersen et al[1]. Accordingly the average field at $\vec{r}$ due to a nucleus centered at $\vec{r_i}$ is given by

$$\langle\vec{F}(\vec{r}, \vec{r}_i, T_0)\rangle = \int_{-\infty}^{+\infty} \vec{F}_i(\vec{r}, \vec{r}_i, \vec{r}_i', T_0)dr_i' \tag{4.8}$$

where

$$\vec{F}_i(\vec{r}, \vec{r}_i, \vec{r}'_i, T_0) = -f(\vec{r}_i, \vec{r}'_i)\left[\frac{\partial V_s(r, r'_i)}{\partial r}\right] \tag{4.9}$$

Here T_0 is the temperature of the crystal and $f(r_i, r'_i)$ is the gaussian probability function

$$f(\vec{r}_i, \vec{r}'_i) = \frac{1}{\sqrt{2\pi u_1^2}}\left[exp\left(\frac{-|\vec{r}_i - \vec{r}'_i|^2}{2u_1^2}\right)\right] \tag{4.10}$$

which gives the probability of the nucleus centered at $\vec{r}_i$ to be displaced to $\vec{r}'_i$ at any instant. u_1^2 is the mean-square displacement of the nucleus in any direction.

4.2 Programme Implementation

To simulate the trajectory, the equation of motion (eqn(4.6)) is integrated numerically using the programme gax(Appendix I). The integration is performed by the Runge-Kutta method. The field at any instant is computed by the subroutines $field$ and $closf$. The summation in equation(4.7) is for only those atoms which come within a distance of $3A^0$. Beyond this distance it is found that the atoms do not contribute to field significantly. Rajasekharan and Neelakandan[2] have shown that the thermal vibrations influence the motion of an ion noticeably only when the ion comes close to within a distance of $1A^0$ from the nucleus. The study conducted so far suggests that chaotic motion arises when channeled ions get too close to crystal nuclei. Therefore consideration of thermal vibrations are necessary for a realistic simulation of channeling in crystals. At a distance of $1A^0$ the nearest nucleus outweighs the other nuclei in its contribution to field. The correction for thermal vibrations is applied only when an ion comes within this critical distance. Even after these assumptions the programme consumes a lot of computation time because of the integral in equation(4.8). To overcome this problem the integral is evaluated at 1000 points within a cube of size $1A^0$ with the nucleus at one corner. The computed values are then stored in a data file. When the programme *gax* is run this data is read into memory. The field at any desired point is determined by interpolation of this thermal data and by applying symmetry.

The programme *gax* also uses a sub-routine *eloss* to consider energy loss suffered by the electrons. But calculations of Rajasekharan and Neelakandan[2] have shown that the energy loss suffered by ions of Mev energies moving through crystal is of the order of a few eV/A^0 only. Therefore in transmission experiments through crystals involving thickness of the order of a few thousand angstrom, the energy loss suffered is less than 1 percent of the incident energy. Hence the energy loss is neglected in our present calculation. But the sub-routine *eloss* help us to incorporate the energy loss also easily when need arises.

Figure(4.2) shows the plane transverse to $\langle 110 \rangle$ axis. The dots represents rows of atoms in the $\langle 110 \rangle$ direction. A unit cell in the shape of a rhombus enclosing two atomic rows is considered. The point midway between the rows is chosen as the origin of a coordinate system, defined with X axis along $\langle 100 \rangle$ axis and Y the $\langle 110 \rangle$ axis. All protons studied are assumed to be incident within the cell. The number of steps upto which a trajectory is to be computed, the position coordinates (x, y) of the points of incidence of the proton and its angles ψ_{in} and θ_{in} are the inputs. Programme *gax* then computes the trajectory and outputs its position and velocity coordinates to a datafile *plot.d* at desired intervals of IO. This data file is used to view the trajectories by using the program *gnplot*(Appendix I). This programme

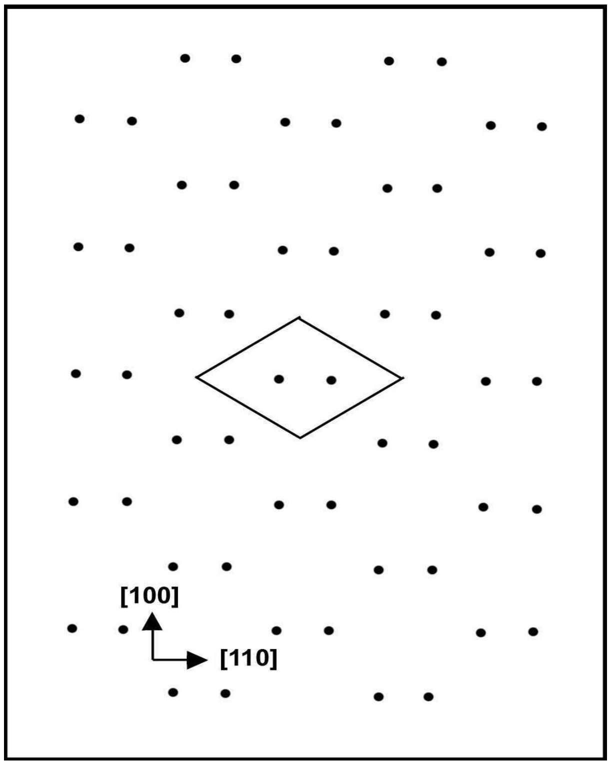

Figure 4.2: The plane transverse to the $\langle 110 \rangle$ direction of silicon.

written in C can generate trajectories in phase space as well as in a plane transverse to the channeling direction.

Figure(4.3) shows the trajectory of a 1MeV proton incident at an angle $\psi_{in} = 0.42^0$ and $\theta_{in} = -15^0$ generated with *gax* and *gnplot*. From the figure it can be seen that the proton avoids the rows of atoms indicating the characteristic feature of axial channeling, even though we haven't made any approximations to incorporate any channeling continuum potentials.

Figure(4.4) shows a plot of an ion incident exactly midway between two atomic rows with $\theta_{in} = 0^0$ and $\psi_{in} = 2^0$. This plot should see a symmetrical distribution of atoms about the (100) direction. Therefore as expected, the trajectory in the transverse plane is a straight line. The

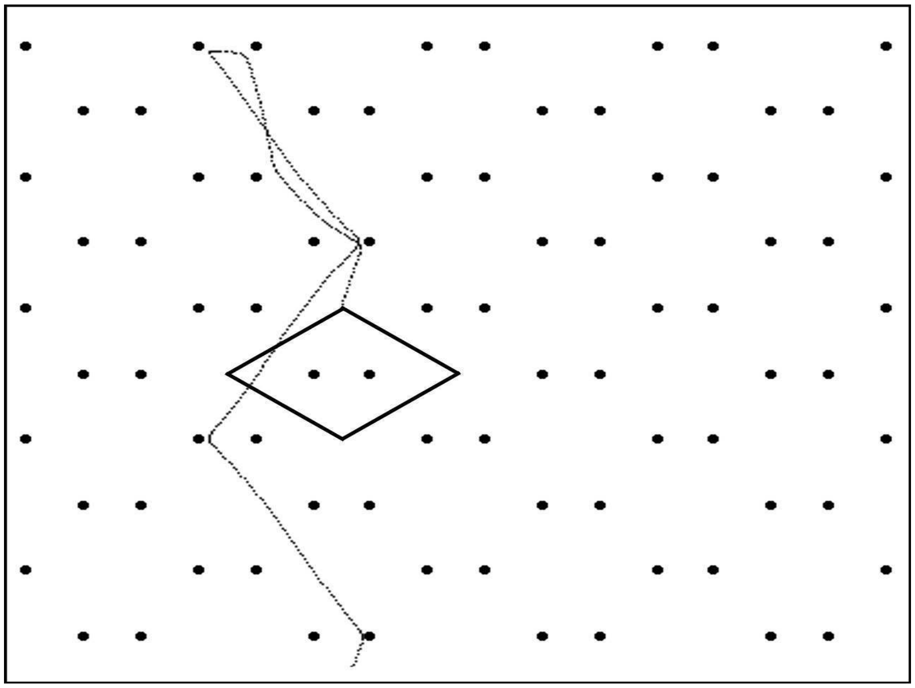
Figure 4.3: Trajectory of a 1MeV proton for $\psi = 0.42^0$ and $\theta = -15^0$

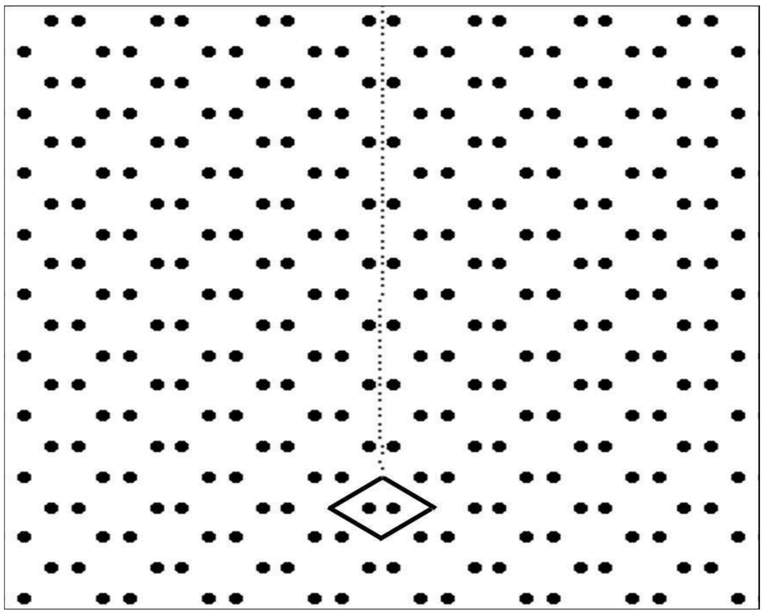
Figure 4.4: Trajectory of a 1MeV proton for $\psi = 2^0$ and $\theta = 0^0$

two figures can be taken to be an indication of a near realistic simulation of particles in a crystal by our programme. The main programme *gax* also determines the Lyapunov spectrum using Wolf's algorithm[3].

When the ion is axially channeled, neglecting the energy loss results in the conservation of momentum in the direction of channeling. Therefore the motion can be analysed in the four-dimensional phase space made up of the generalized position directions (q_x, q_y) and generalized momentum direction (p_x, p_y). These are defined by the equations

$$q_x = \frac{x}{l_x}, q_y = \frac{y}{l_y} \tag{4.11}$$

and

$$p_x = \frac{v_x}{v_\perp}, p_y = \frac{v_y}{v_\perp} \tag{4.12}$$

where l_x and l_y are the periodicities in the crystal structure in the x and y directions respectively and $v_\perp = \sqrt{\frac{2E}{M}}$. The Lyapunov spectra made up of four exponents is evaluated using Wolf's algorithm for four-dimensional phase space. The equation of motion(4.6) is re-written as four one-dimensional equations

$$\dot{q}_x = \frac{v_\perp}{l_x} p_x \tag{4.13}$$

$$\dot{q}_y = \frac{v_\perp}{l_y} p_y \tag{4.14}$$

$$\dot{p}_x = \frac{F_x}{m v_\perp} \tag{4.15}$$

$$\dot{p}_y = \frac{F_y}{m v_\perp} \tag{4.16}$$

where F_x and F_y are the forces in the respective directions. The above non linear equations are then linearized by using the jacobian

$$J = \begin{pmatrix} 0 & 0 & \frac{v}{l_x} & 0 \\ 0 & 0 & 0 & \frac{v}{l_y} \\ \frac{1}{mv_\perp}\frac{\partial F_x}{\partial q_x} & 0 & \frac{1}{mv_\perp}\frac{\partial F_x}{\partial p_x} & 0 \\ 0 & \frac{1}{mv_\perp}\frac{\partial F_y}{\partial q_y} & 0 & \frac{1}{mv_\perp}\frac{\partial F_y}{\partial p_y} \end{pmatrix} \tag{4.17}$$

The sub programme *jacobs* evaluates the jacobian and returns the values to *gax* to compute all the four Lyapunov exponents using the equations (4.13) to (4.16).

4.3 Results

We considered the channeling of 1MeV protons in silicon in the (110) direction. Table(4.1) shows the critical angles for various types of channeling of 1MeV protons in silicon. An ion incident at (0,1.96) with $\psi_{in} = 0.1^0$ and $\theta_{in} = 30^0$ is considered for computation of the Lyapunov spectrum. By table(4.1) the trajectory should be hyperchanneled.

Table 4.1: Critical angles for various types of channeling. For planar channeling the angle considered is the angle made by the incident momentum vector of the ion with the planes, between which the ions are channeled.

Type of channeling	Critical angle in degree
hyper channeling	$\psi = 0.128^0$
axial channeling	$\psi = 0.587^0$
planar channeling	$\phi_r = 1^0$

Table(4.2) shows the values of the four Lyapunov exponents computed in steps of 0.13 upto a depth of $27\mu m$. The values are in the units of

$second^{-1}$.

Table 4.2: Lyapunov spectra calculated using Wolf's algorithm

λ_1	λ_2	λ_3	λ_4
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.985050E+00	0.896956E+00	0.696742E+00	0.472503E+00
0.563397E+01	0.547247E+01	0.333420E+01	0.284969E+01
0.175260E+02	0.171556E+02	0.964848E+01	0.874750E+01
0.400145E+02	0.392545E+02	0.213147E+02	0.198459E+02
0.749423E+02	0.736235E+02	0.392304E+02	0.370774E+02
0.124097E+03	0.122102E+03	0.643196E+02	0.613496E+02
0.189865E+03	0.187056E+03	0.977839E+02	0.938522E+02
0.276533E+03	0.272771E+03	0.141811E+03	0.136738E+03
0.388190E+03	0.383338E+03	0.198397E+03	0.192056E+03
0.527820E+03	0.521794E+03	0.269010E+03	0.261291E+03
0.697896E+03	0.690551E+03	0.354891E+03	0.345675E+03
0.899706E+03	0.890949E+03	0.456671E+03	0.445867E+03
0.113433E+04	0.125338E+04	0.640452E+03	0.627072E+03
0.140352E+04	0.139159E+04	0.710465E+03	0.696170E+03
0.170913E+04	0.169543E+04	0.864281E+03	0.848085E+03
0.205293E+04	0.203735E+04	0.103724E+04	0.101904E+04
0.243714E+04	0.241956E+04	0.123046E+04	0.121013E+04
0.286552E+04	0.284580E+04	0.144586E+04	0.142324E+04
0.334574E+04	0.332372E+04	0.168730E+04	0.166220E+04
0.389129E+04	0.386680E+04	0.196158E+04	0.193374E+04
0.451224E+04	0.448534E+04	0.227357E+04	0.224303E+04
0.522039E+04	0.519101E+04	0.262931E+04	0.259590E+04
0.603976E+04	0.600760E+04	0.304096E+04	0.300426E+04
0.699791E+04	0.696269E+04	0.352231E+04	0.348188E+04

Figure(4.5) shows that the ion stays hyper channeled even upto this depth. But the motion is non-periodic. Calculated values of the four Lyapunov exponents show that there is no convergence even after this depth. The positive value for the sum of exponents means that the transverse energy of the proton is actually increasing with depth. This is consistent with the findings of Desalvo et al[4]. The lack of convergence may be due to the fact that the algorithm is not suitable for this case. Hence we decided to use the time series analysis for the output data.

4.4 Time Series Analysis

In the last decade great progress has been achieved in determining the Maximum Lyapunov Characteristic Exponent(MLCE) from experimental data[3], [5-7]. An experimental result is made up of a series of univariate measurements equidistant in time or some other variable. The data represented as $x_t, t = 1.....n$ is called a time series. Apart from the possible noise in the measurements there is a further problem of determining the dimension of the phase space which could correctly represent the dynamics of the system. Therefore to reveal the dynamical properties of the system a new phase space is constructed. This is often done by the method of delays. Here we generate a new series $(x_n - (m-1)\tau, x_n - (m-2)\tau, ..., x_n)$. The number m is called *embedding dimension* and τ is referred to as the *time delay* or *time lag*. In the new phase space we take an arbitrary point x_t. Then all neighbours x_i in the neighbourhood U_t are considered and the average of the distance between all neighbouring trajectories and the reference trajectory x_t is computed as a function of a relative time say τ . To get rid of the possible fluctuations we take the logarithm of these average distances . Thus one computes

$$S(\tau) = \frac{1}{n} \sum_{t=1}^{n} \ln(\frac{1}{\mid U_t) \mid} \sum_{i \varepsilon u_t} \mid x_{t+\tau} - x_{i+\tau} \mid) \tag{4.18}$$

If $S(\tau)$ increases linearly with identical slopes for all embedding dimension m greater than a value m_0 and for a reasonable range in size of the neighbourhood, the slope is taken as an estimate of the MLCE. This algorithm is implemented using the $TISEAN$ software package which is publicly available from http://www.mpipks-dresden.mpg.de/ tisean.

4.5 Calculation

$TISEAN$ package is not one of the usual kind of programmes where one feeds in the data to get the output. Instead the package is a collection of several short programmes with which the data can be studied in detail before drawing the final conclusions. Here we have analysed the data contained in *plot.d* using the programmes to determine the *embedding dimension* and *time delay* before evaluating the MLCE using the programme *lyap_k*. We have considered the trajectories of 1MeV protons in silicon (110) plane. The incident angles have been so chosen to yield each trajectory representative of one channeling characteristic feature.

4.5.1 Hyper channeling

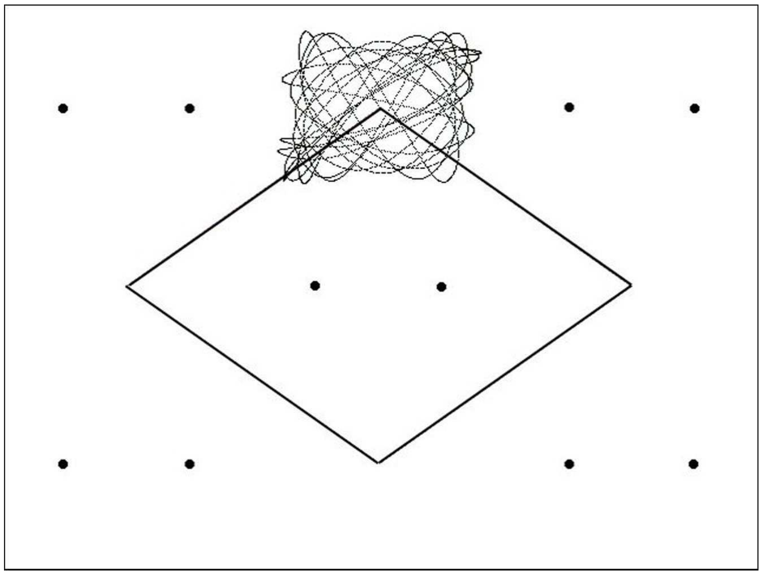

Figure 4.5: Hyper channeled trajectory for $\psi = 0.1^0$, $\theta = 30^0$

Figure(4.5) shows a simulated hyper channeled trajectory for $\psi_{in} = 0.1^0$ and $\theta_{in} = 30^0$ using *gax*. Trajectory is simulated upto 10000 steps and the programme *lyap_k* generates the datafile to plot $S(\tau)$ against τ. Figure (4.6) to (4.13) shows, $S(\tau)$ plotted against τ for delay between 1 and 8 and dimension from 2 to 5. For computation of $S(tau)$, the program considers only the values of x-coordinate in the data file *plot.d*. This is because the information regarding the Lyapunov exponents is contained identically in all the position and velocity data. Hence instead of the x coordinates if y or any of the velocity coordinates v_x and v_y is used, *lyap_k* gives the same output.

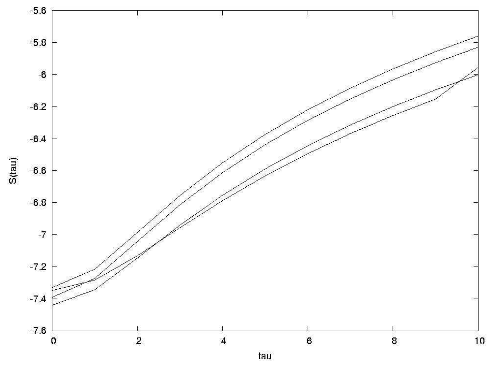

Figure 4.6: delay=1

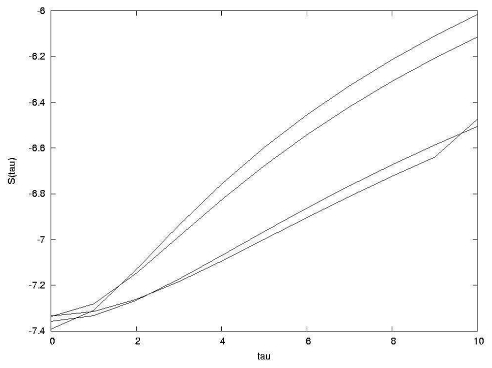

Figure 4.7: delay=2

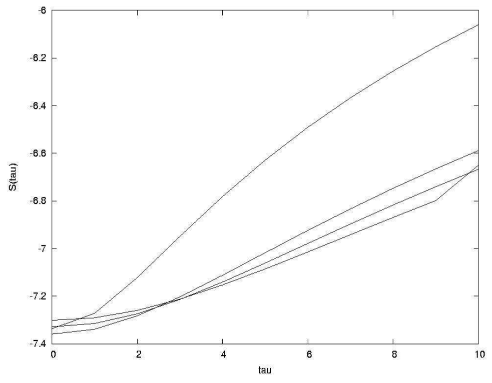

Figure 4.8: delay=3

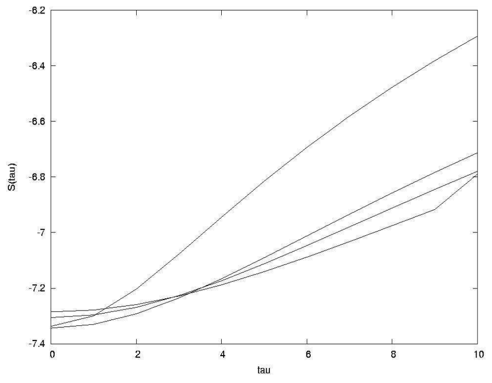

Figure 4.9: delay=4

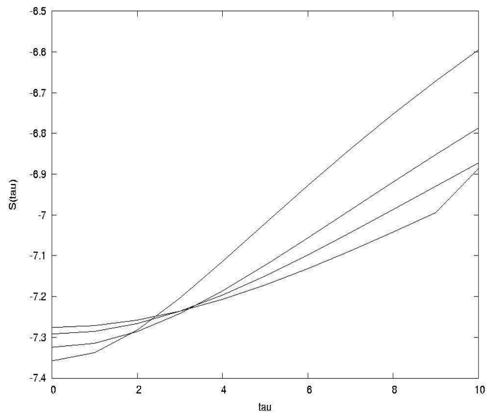

Figure 4.10: delay=5

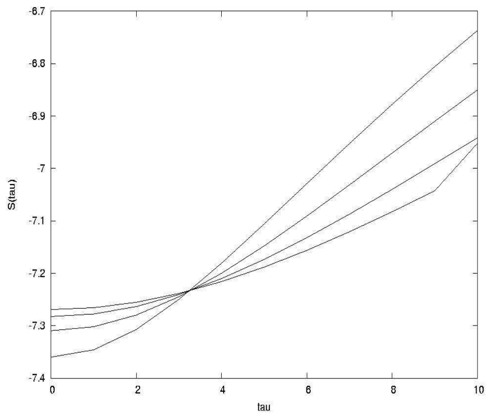

Figure 4.11: delay=6

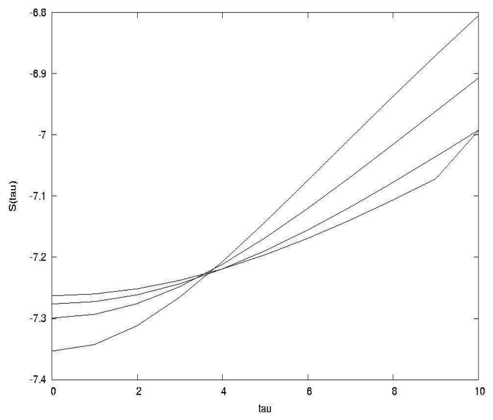

Figure 4.12: delay=7

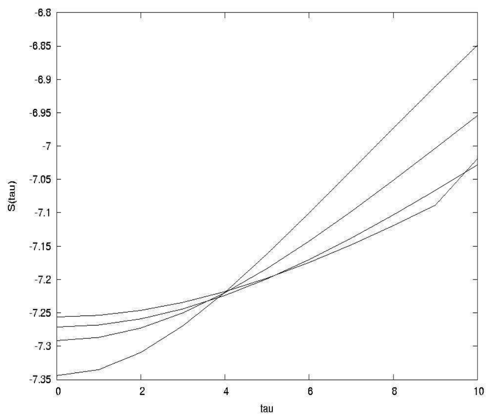

Figure 4.13: delay=8

From the above figures it can be seen that for delay=2, the dimensions 4 and 5 gives identical slopes. Therefore choosing the dimension to be 4, the slope of the straight region of the curve fig(4.14) is determined to yield the Lyapunov exponent as 0.1024/iterations. Since the time step of integration in *gax* was 10^{-16}seconds, dividing the above value by the time step gives the value of MLCE as $0.1024\text{x}10^{16}/s$.

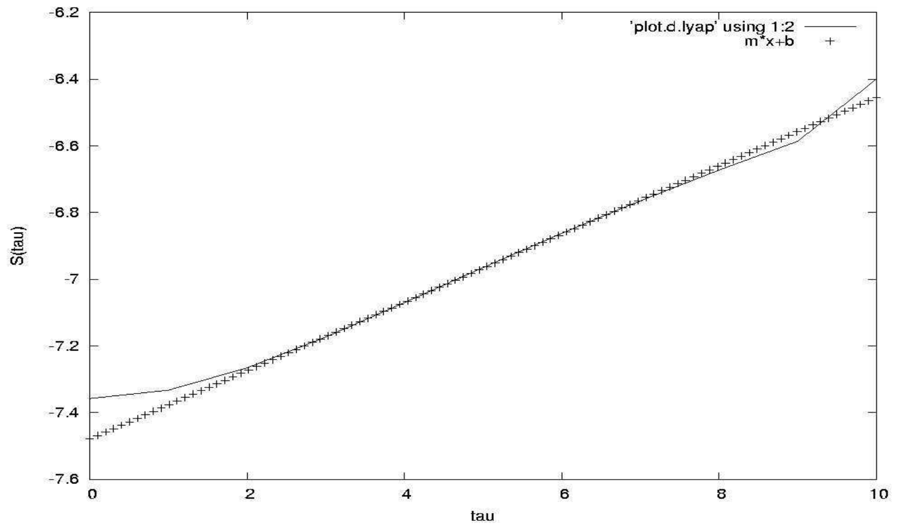

Figure 4.14: Plot of $S(\tau)$ versus τ for hyper channeled trajectory in fig(4.5) with delay=2 and m=4

4.5.2 Quasi-hyper channeling

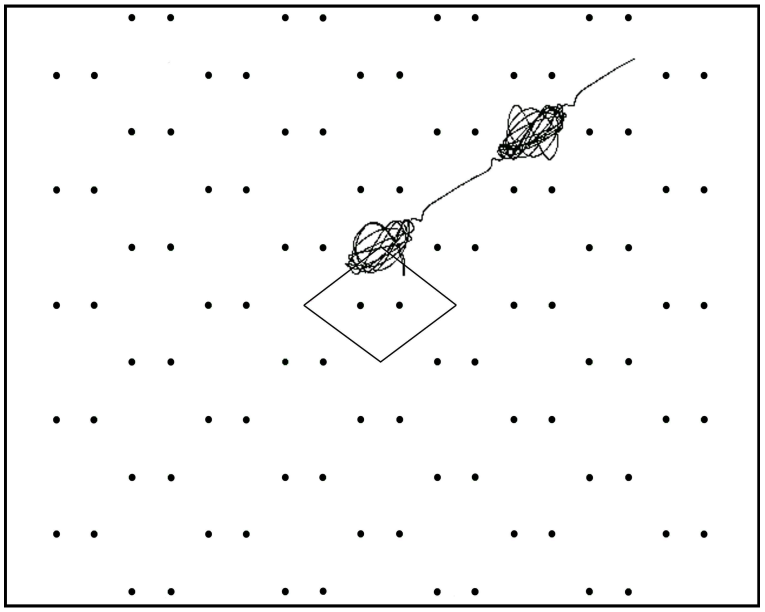

Figure 4.15: Quasi-hyper channeled trajectory for $\psi = 0.114^0$ and $\theta = -30^0$

For $\psi_{in} = 0.114^0$ and $\theta_{in} = -30^0$, a quasi hyper channeled trajectory is simulated using gax as shown in figure(4.15). $S(\tau)$ plotted against τ for delay between 1 and 8 and dimension from 2 to 5 are shown in figures(4.16) to (4.23).

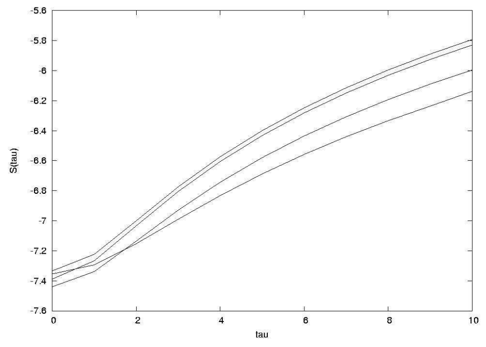

Figure 4.16: delay=1

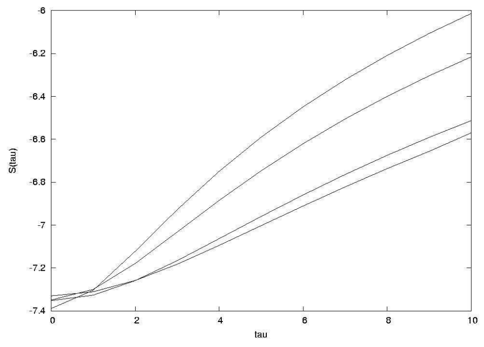

Figure 4.17: delay=2

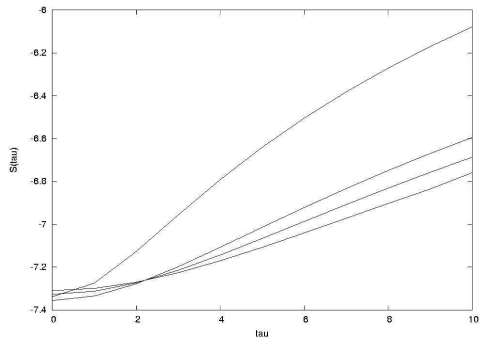

Figure 4.18: delay=3

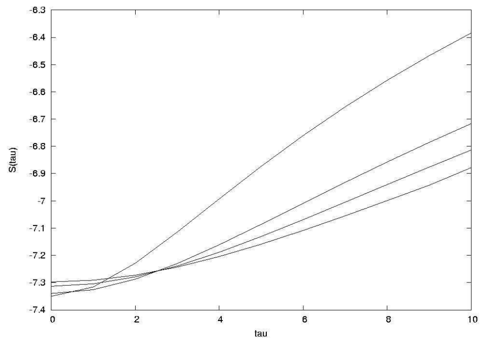

Figure 4.19: delay=4

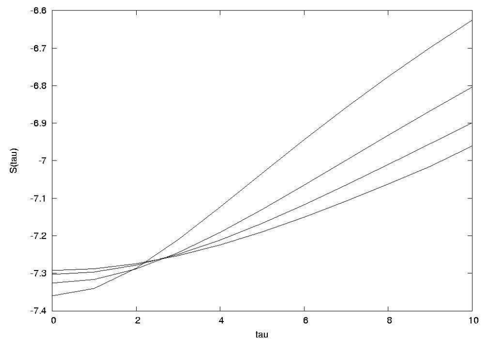

Figure 4.20: delay=5

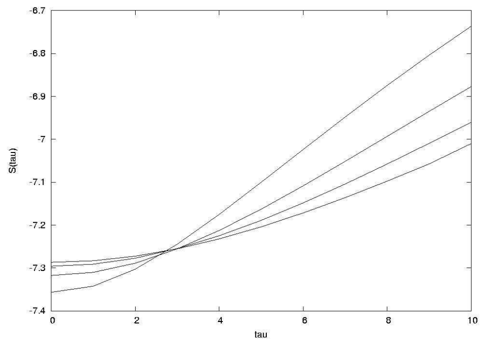

Figure 4.21: delay=6

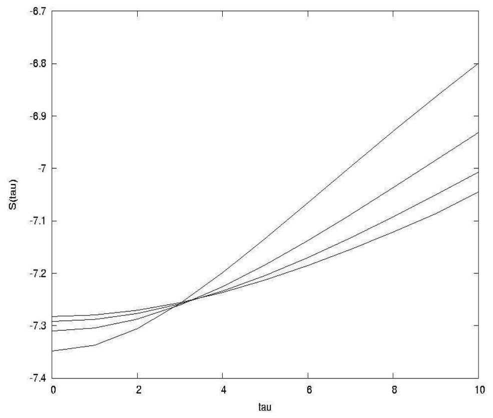

Figure 4.22: delay=7

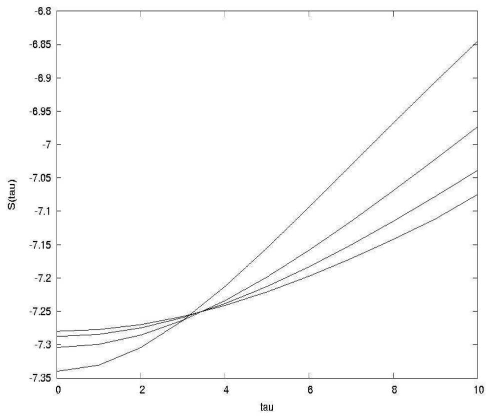

Figure 4.23: delay=8

From the above figures it can be seen that a delay of 2 and dimension of 4 is suitable for finding the Lyapunov exponent. The slope of the curve (fig(4.24)) then yield a MLCE value of 0.0999/iterations.

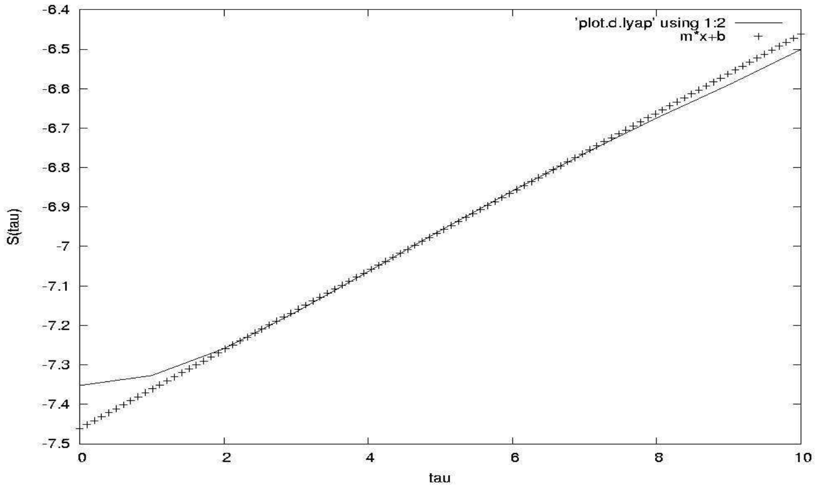

Figure 4.24: Plot of $S(\tau)$ versus τ for quasi hyper channeled trajectory in fig(4.15) with delay=2 and m=4

4.5.3 Axial channeling

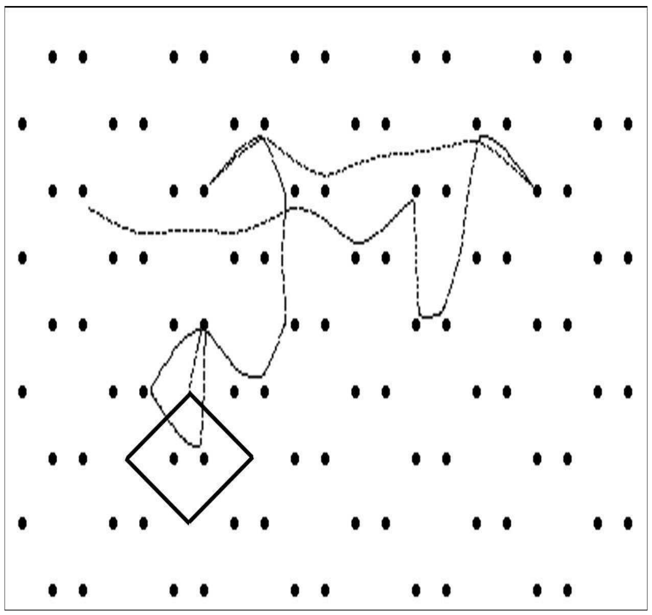

Figure 4.25: Axial channeled trajectory for $\psi = 0.4^0$ and $\theta = -17^0$

An axially channeled trajectory is simulated with $\psi_{in} = 0.4^0$ and $\theta_{in} = -17^0$, using *gax*. Trajectory is simulated upto 10000 steps using *lyap_k*, and $S(tau)$ is plotted against *tau* for delay between 1 and 8 and dimension from 2 to 5 in figures(4.26) to (4.33).

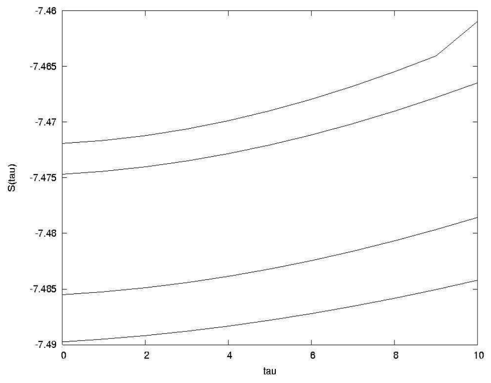

Figure 4.26: delay=1

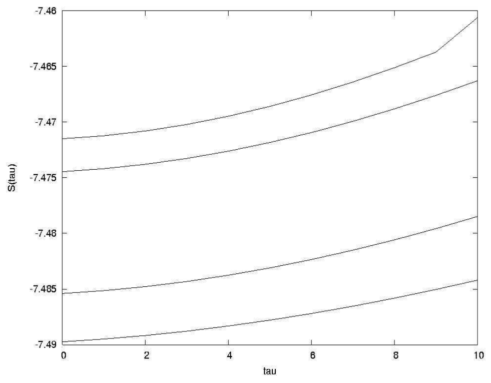

Figure 4.27: delay=2

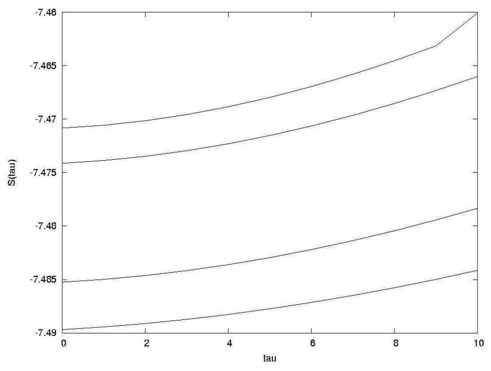

Figure 4.28: delay=3

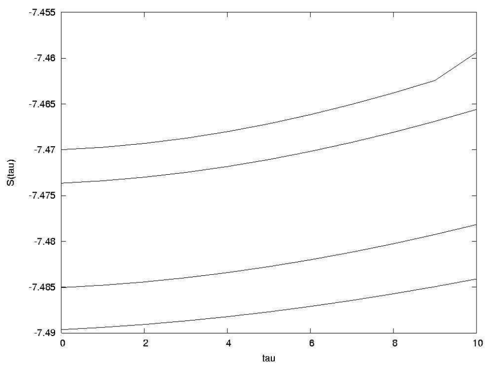

Figure 4.29: delay=4

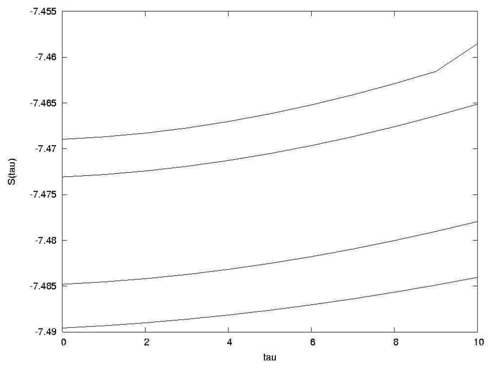

Figure 4.30: delay=5

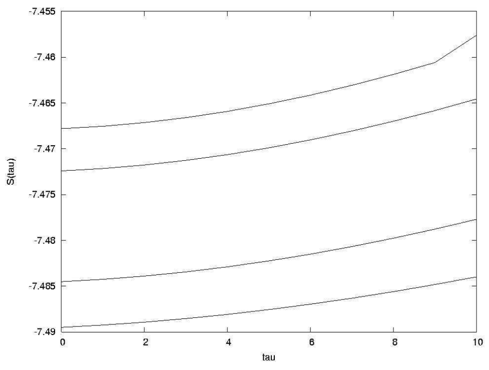

Figure 4.31: delay=6

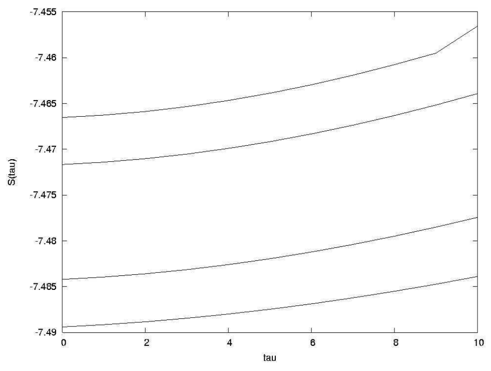

Figure 4.32: delay=7

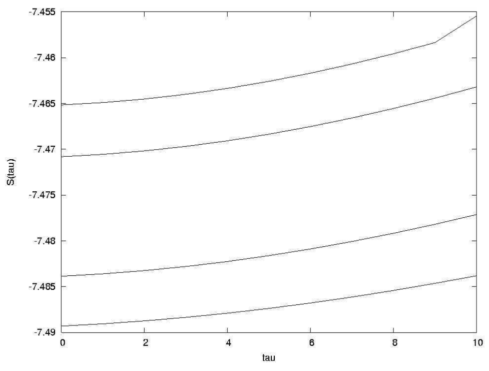

Figure 4.33: delay=8

From the above figures it can be seen that a delay of 2 and dimension of 4 is suitable for finding the Lyapunov exponent. Figure(4.34) shows

$S(\tau)$ vs τ for the chosen delay and dimension. The slope of the selected region of the curve in the figure(4.34) yields MLCE value of 0.00125/iterations.

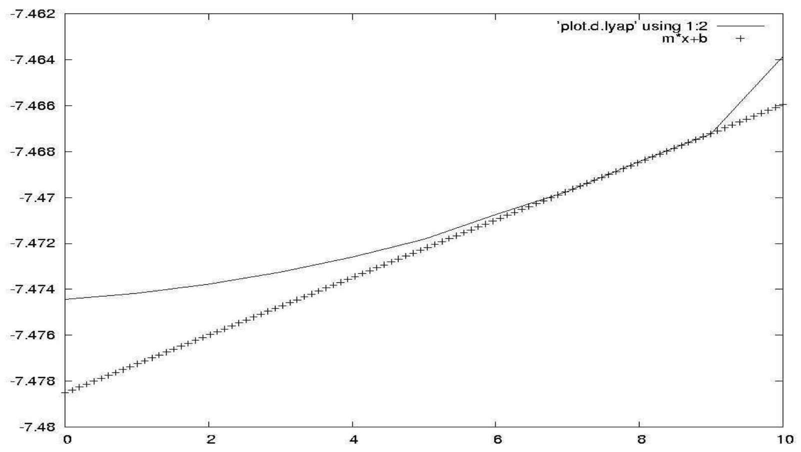

Figure 4.34: Plot of $S(\tau)$ versus τ for axial channeled trajectory in fig(4.25) with delay=2 and m=4

4.5.4 Quasi-axial channeling

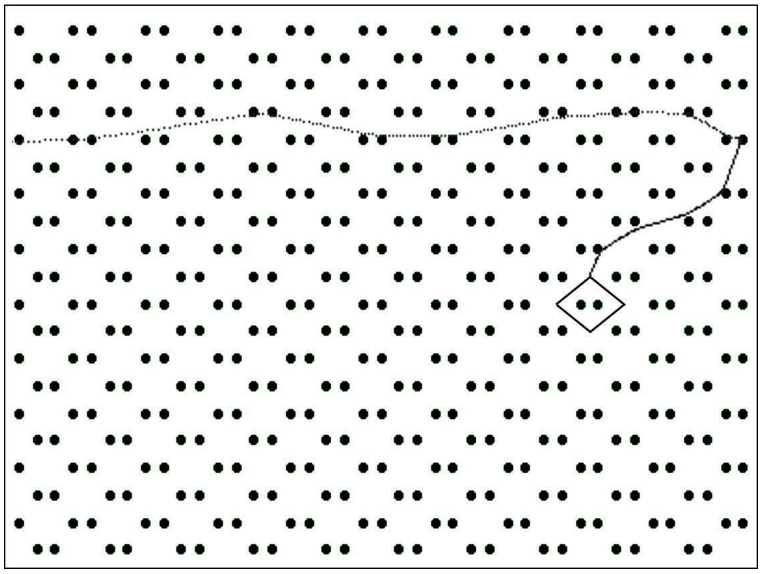

Figure 4.35: Quasi-axial channeled trajectory for $\psi = 0.538^0$ and $\theta = -25^0$

A quasi axially channeled trajectory(figure(4.35)) is simulated for

$\psi_{in} = 0.538^0$ and $\theta_{in} = -25^0$. Figures (4.36) to (4.43) shows $S(tau)$ variation for delay between 1 and 8 and dimension from 2 to 5.

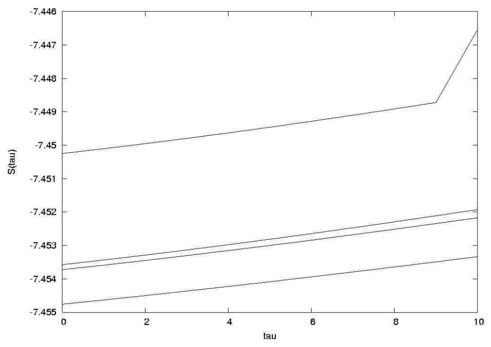

Figure 4.36: delay=1

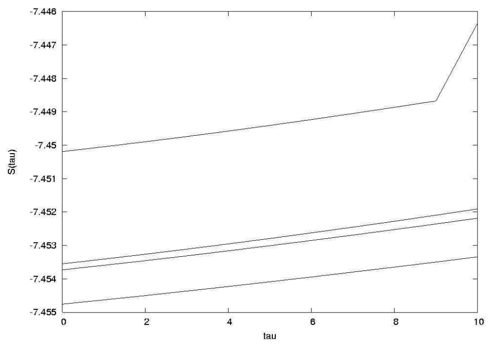

Figure 4.37: delay=2

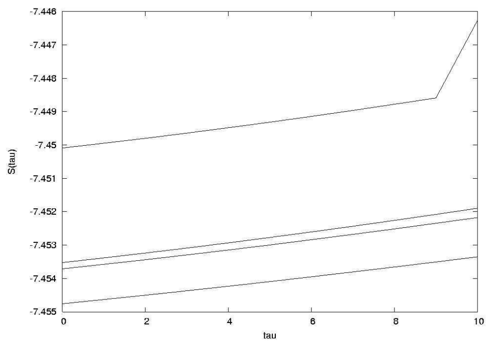

Figure 4.38: delay=3

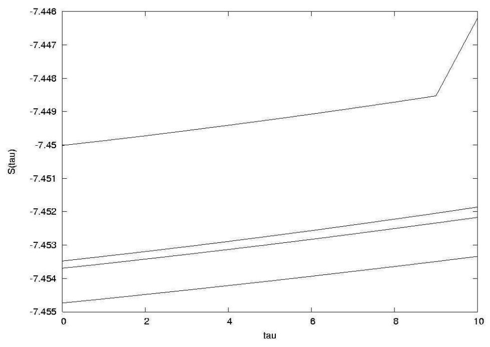

Figure 4.39: delay=4

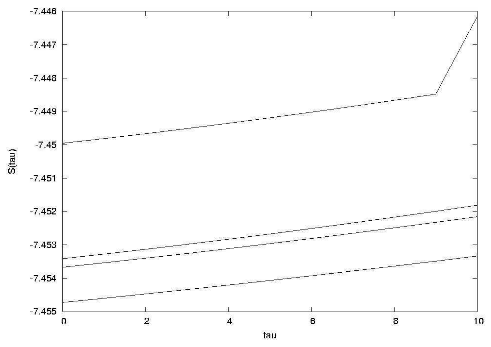

Figure 4.40: delay=5

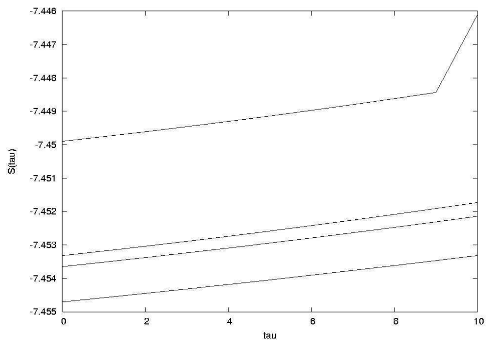

Figure 4.41: delay=6

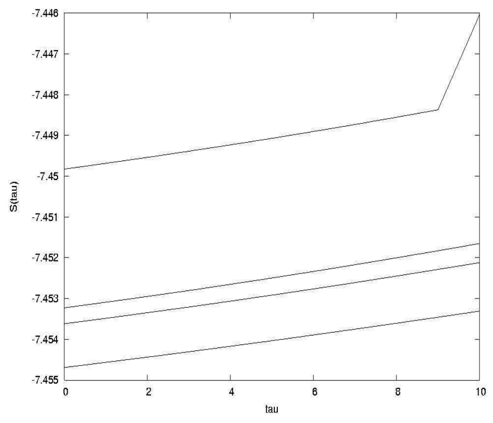

Figure 4.42: delay=7

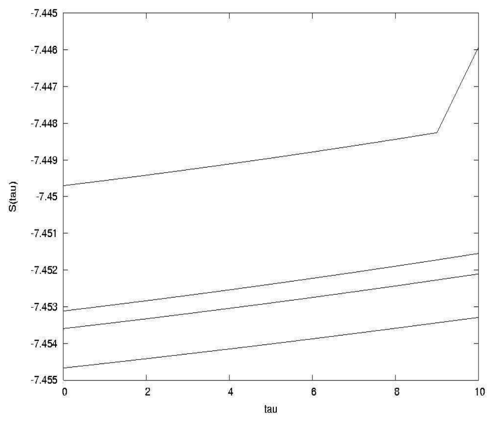

Figure 4.43: delay=8

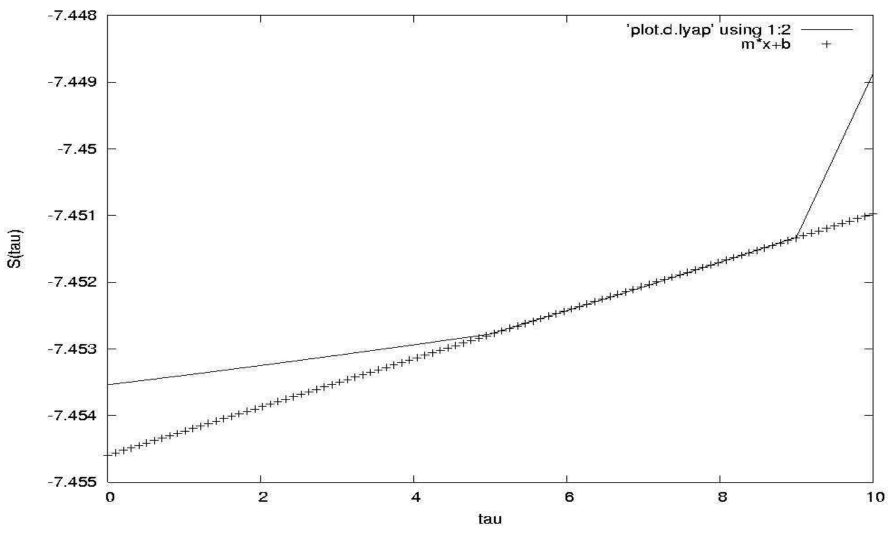

Figure 4.44: Plot of $S(\tau)$ versus τ for quasi axial channeled trajectory in fig(4.35) with delay=2 and m=4

Choosing a dimension of 4 and delay of 2, the slope of the curve in figure(4.44) gives the gives the value of MLCE as 0.00036/iterations.

4.5.5 Planar channeling

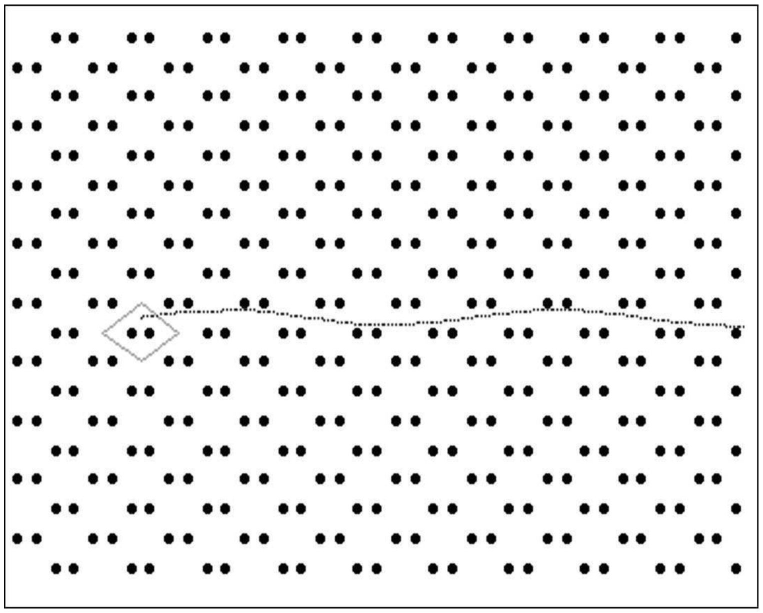
Figure 4.45: Planar channeled trajectory $\psi = 1^0$, $\theta = -83.7^0$

For $\psi_{in} = 1^0$ and $\theta_{in} = -83.7^0$, a planar channeled trajectory is simulated(fig(4.45))using *gax*. Figure(4.46) to (4.54) shows $S(tau)$ plotted against tau for delay between 1 and 8 and dimension from 2 to 5.

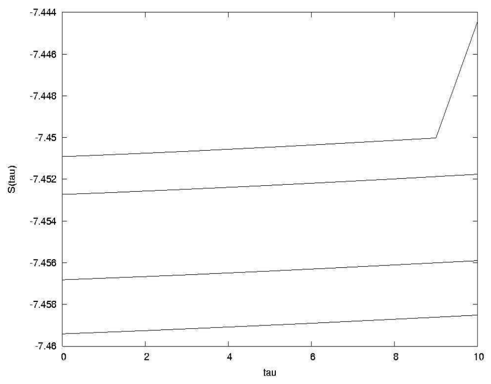

Figure 4.46: delay=1

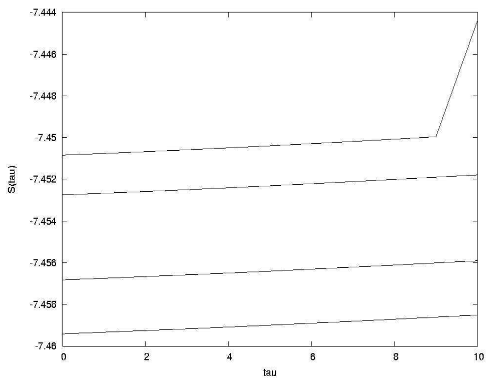

Figure 4.47: delay=2

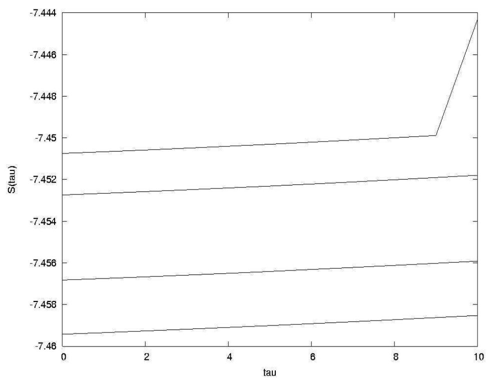

Figure 4.48: delay=3

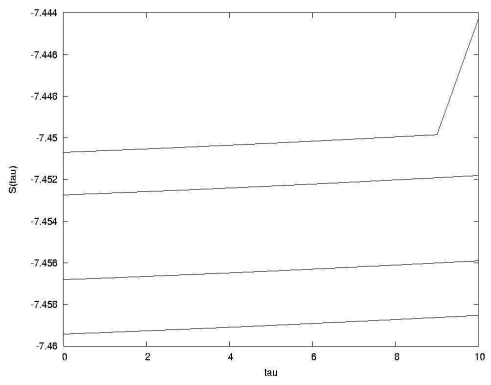

Figure 4.49: delay=4

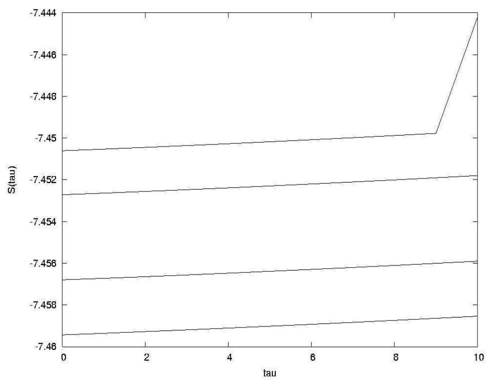

Figure 4.50: delay=5

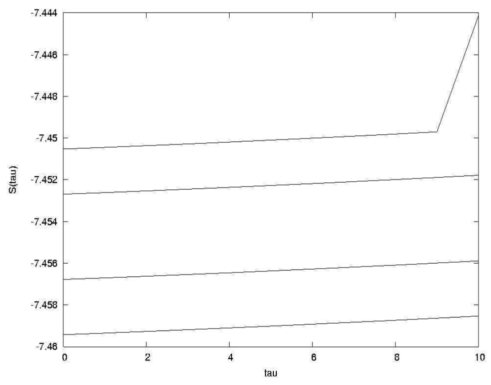

Figure 4.51: delay=6

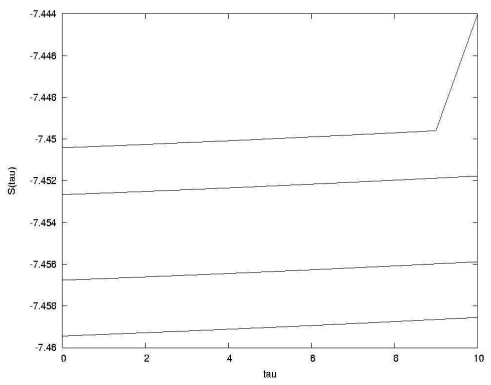

Figure 4.52: delay=7

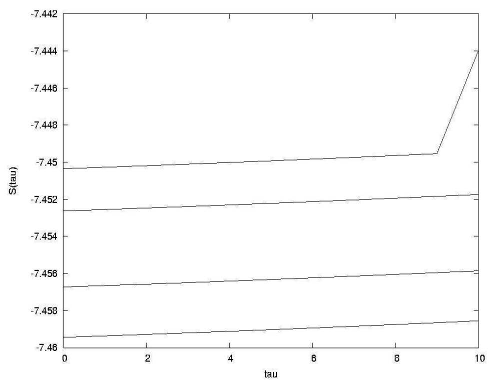

Figure 4.53: delay=8

Choosing a delay of 1 and dimension of 5, the slope of the selected region in the curve(figure(4.54)) gives the value of MLCE as 0.000298/iterations.

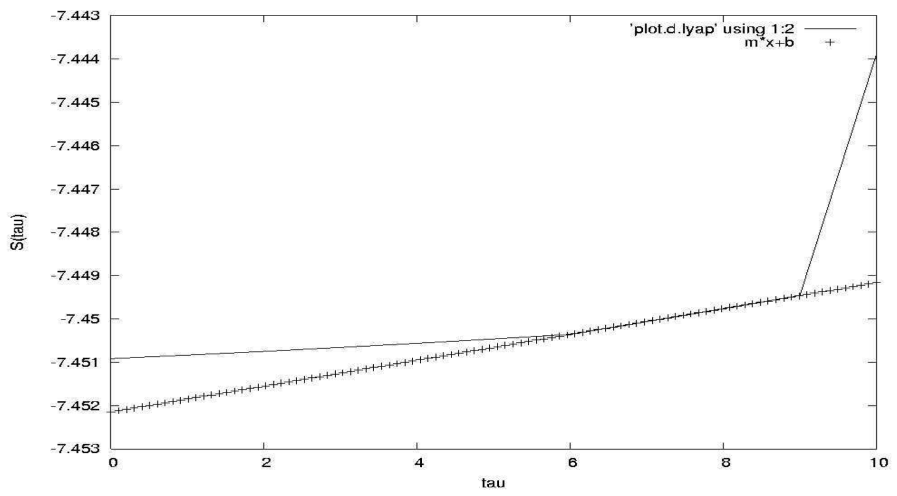

Figure 4.54: Plot of $S(\tau)$ versus τ for planar channeled trajectory in fig(4.45) with delay=1 and m=5

4.5.6 Quasi-planar channeling

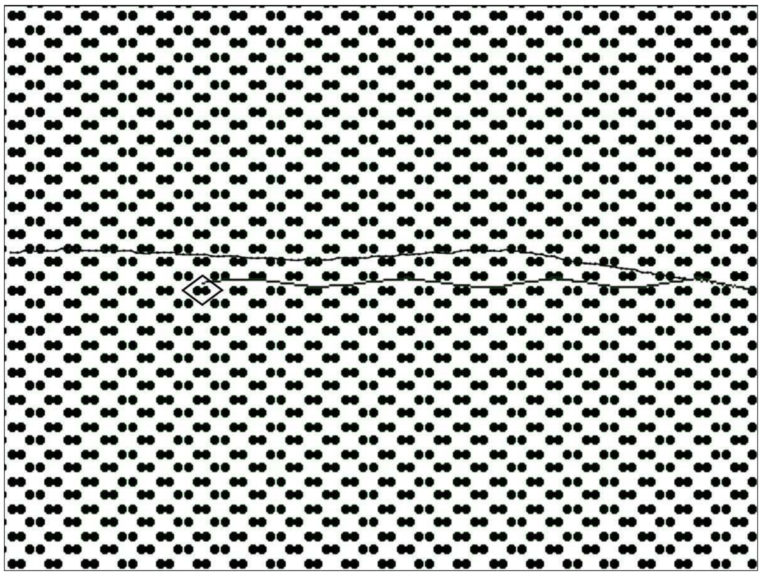

Figure 4.55: Quasi-planar channeled trajectory for $\psi = 1^0$ and $\theta = -83.582^0$

$S(tau)$ is plotted against tau for a quasi planar channeled trajectory for delay between 1 to 8 and dimension 2 to 5 as shown in figures(4.56) to figure(4.63).

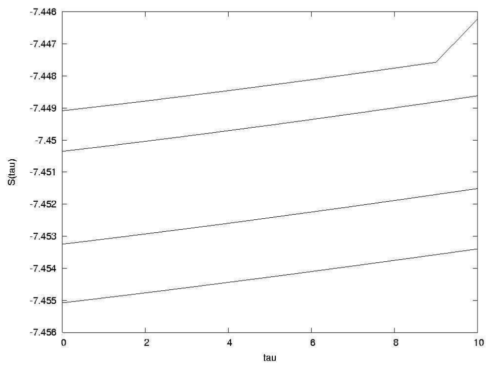

Figure 4.56: delay=1

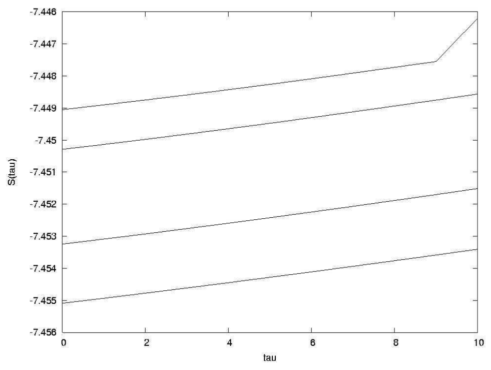

Figure 4.57: delay=2

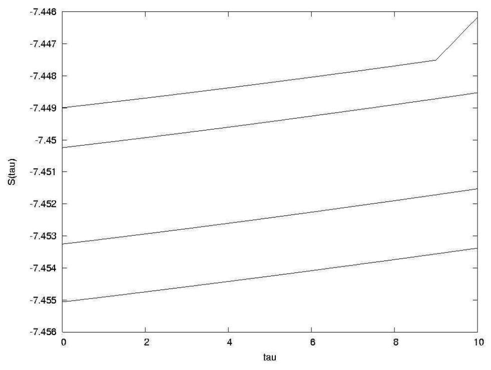

Figure 4.58: delay=3

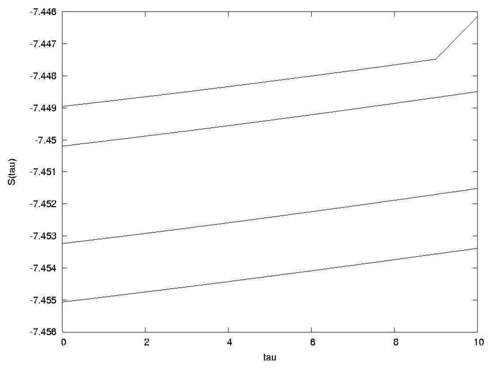

Figure 4.59: delay=4

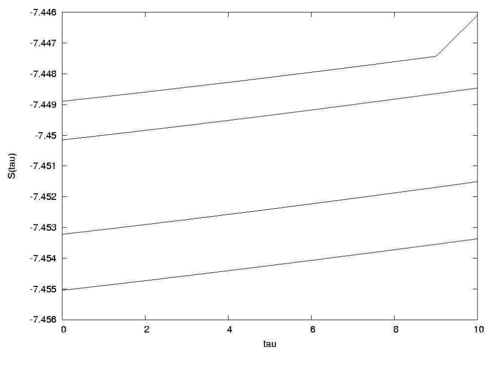

Figure 4.60: delay=5

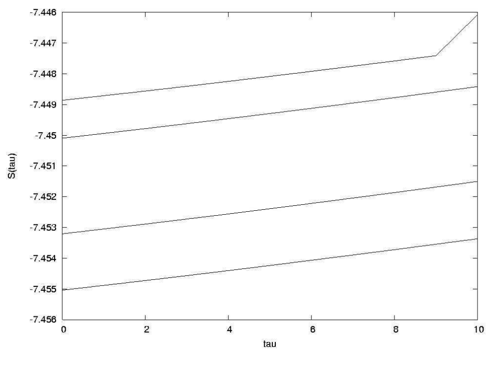

Figure 4.61: delay=6

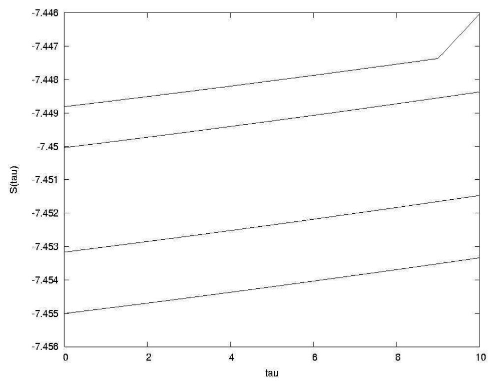
Figure 4.62: delay=7

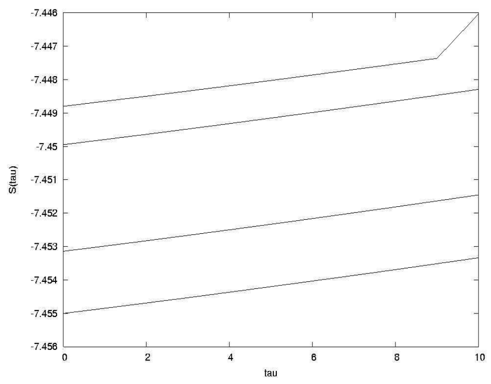
Figure 4.63: delay=8

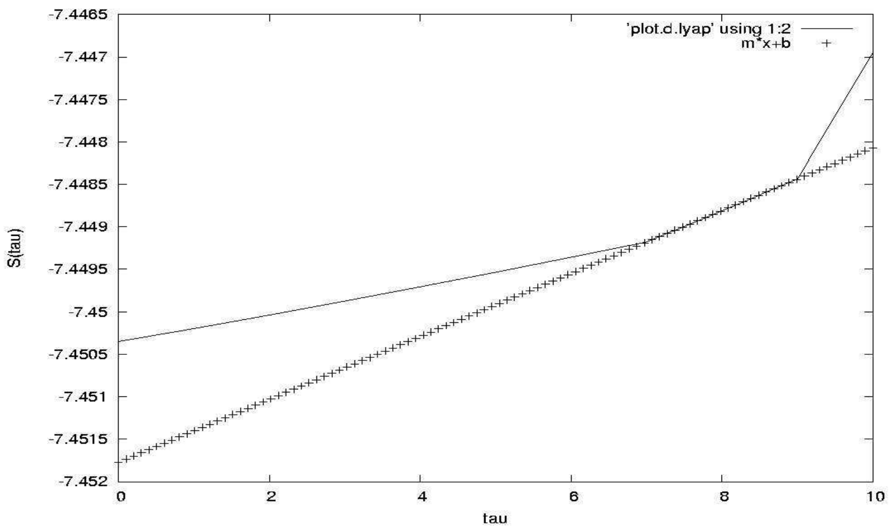

Figure 4.64: Plot of $S(\tau)$ versus τ for quasi planar channeled trajectory in fig(4.55) with delay=1 and m=5

Figure(4.64) shows the variation of $S(\tau)$ with τ for a delay of 1 and dimension of 5. The slope of the plot gives the value of MLCE=0.00036/iterations.

4.5.7 Random motion

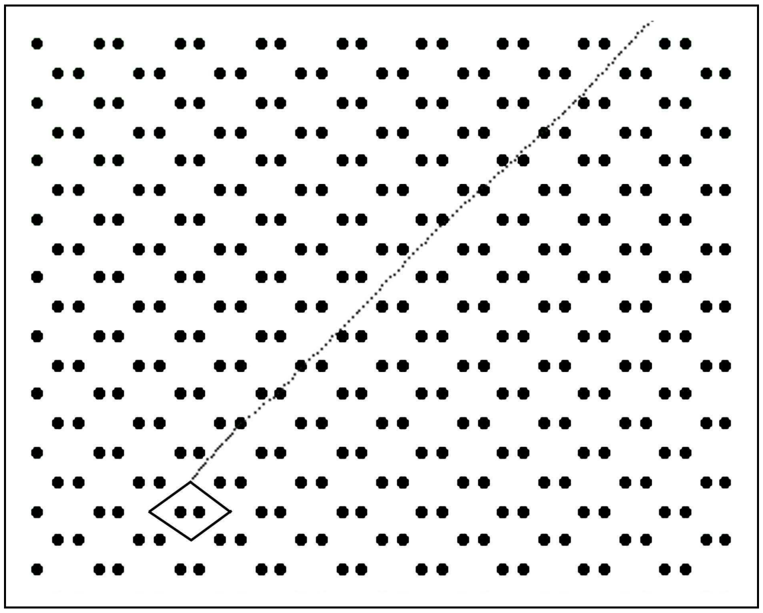

Figure 4.65: Random trajectory for $\psi = 1^0$ and $\theta = -40^0$

A random trajectory is simulated for $\psi_{in} = 1^0$ and $\theta_{in} = -40^0$. Figure(4.66) to (4.73) shows $S(tau)$ plotted against tau for delay between 1 and 8 and dimension 2 to 5.

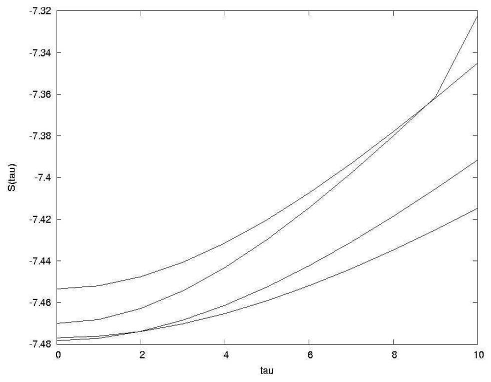

Figure 4.66: delay=1

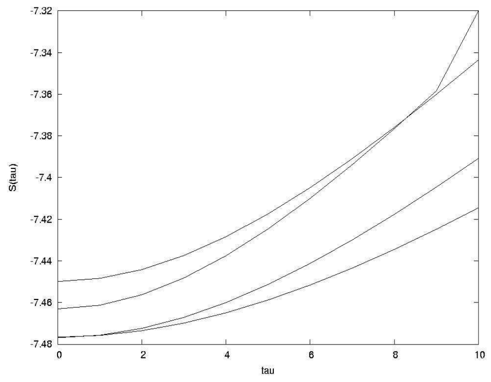

Figure 4.67: delay=2

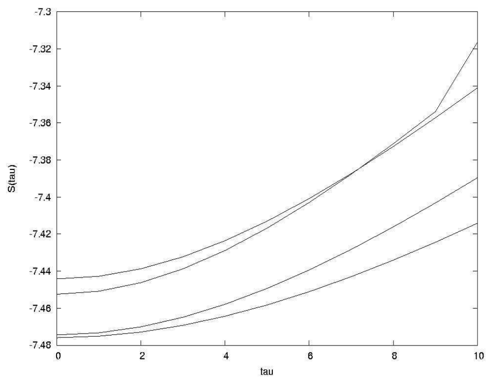

Figure 4.68: delay=3

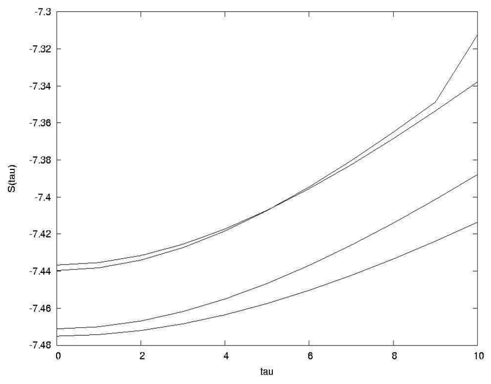

Figure 4.69: delay=4

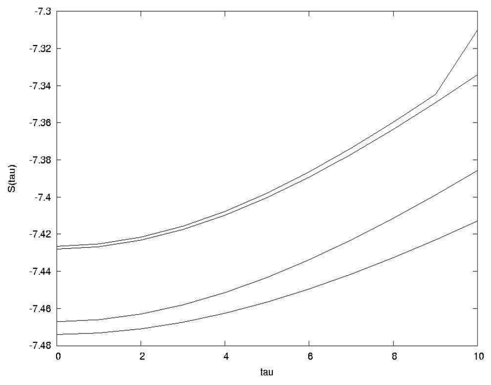

Figure 4.70: delay=5

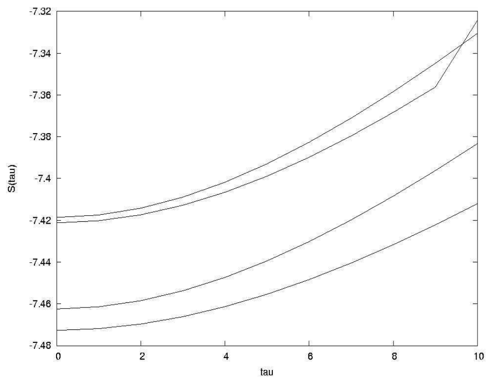

Figure 4.71: delay=6

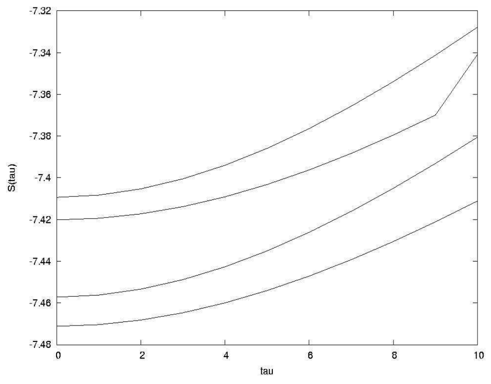

Figure 4.72: delay=7

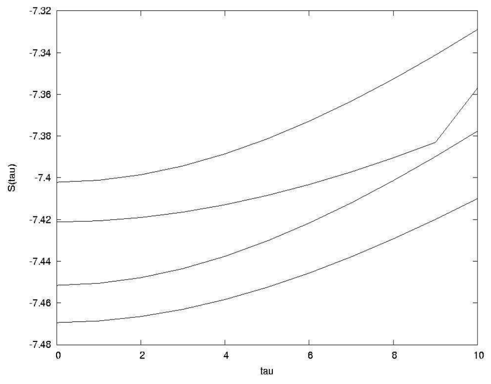

Figure 4.73: delay=8

Figure(4.74) shows the curve for delay=2 and dimension=4. The slope of the selected region of the above curve yields the value of MLCE as 0.015/iterations.

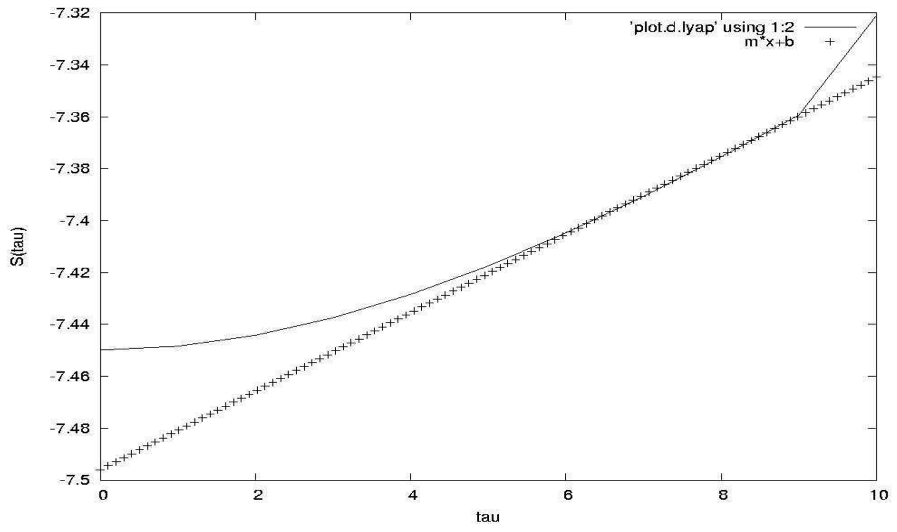

Figure 4.74: Plot of $S(\tau)$ versus τ for random trajectory in fig(4.65) with delay=2 and m=4

In the case of a proton channeled with $\psi = 2^0$ and $\theta = 0^0$ the trajectory is a straight line as shown in figure(4.4). Trajectory is simulated upto 10000 steps and the programme *lyap_k* generates the datafile to plot $S(\tau)$ against τ. Figure(4.75) to (4.82) shows $S(tau)$ plotted against tau for delay between 1 and 8 and dimension from 2 to 5.

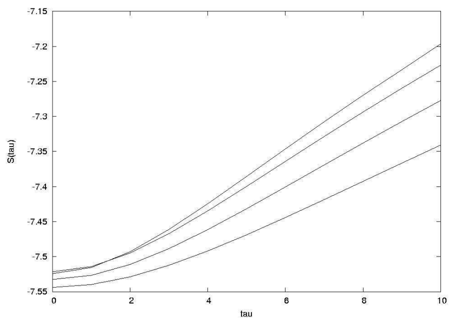

Figure 4.75: delay=1

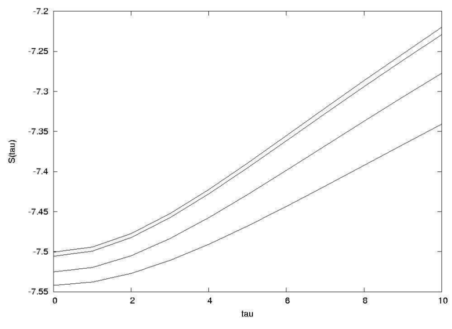

Figure 4.76: delay=2

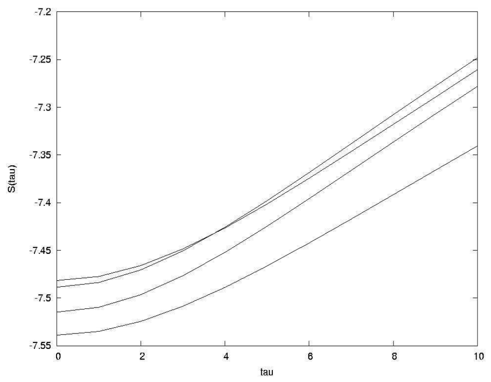

Figure 4.77: delay=3

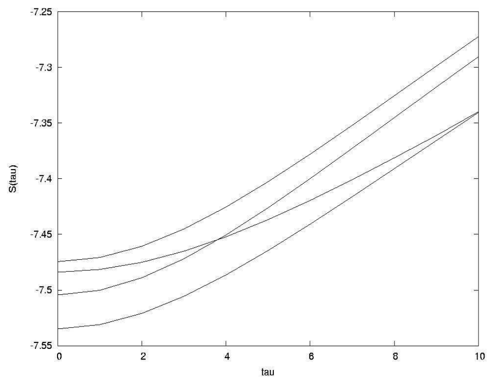

Figure 4.78: delay=4

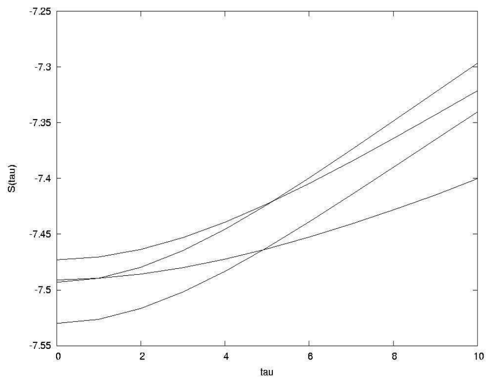

Figure 4.79: delay=5

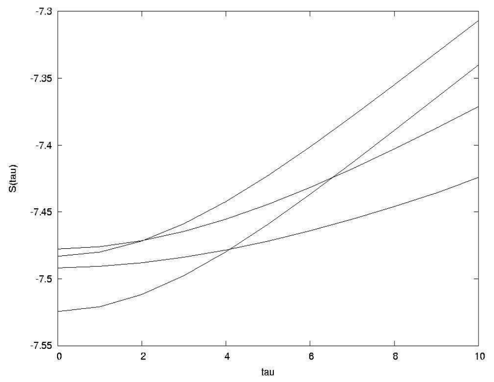

Figure 4.80: delay=6

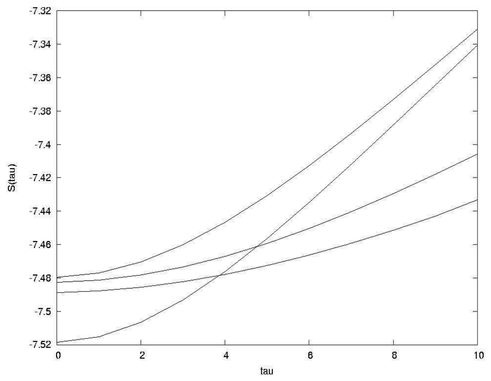

Figure 4.81: delay=7

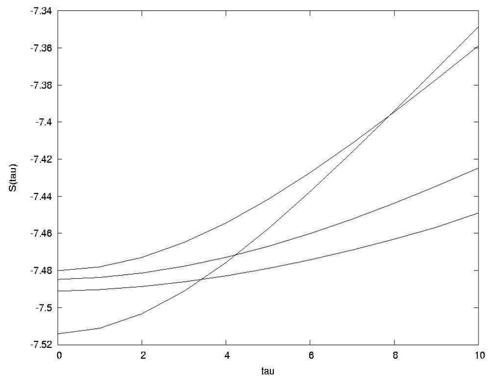

Figure 4.82: delay=8

Figure(4.83) shows the variation of $S(\tau)$ with τ for a delay of 2 and dimension of 4. The slope of the curve gives the value of MLCE as 0.00036/iterations.

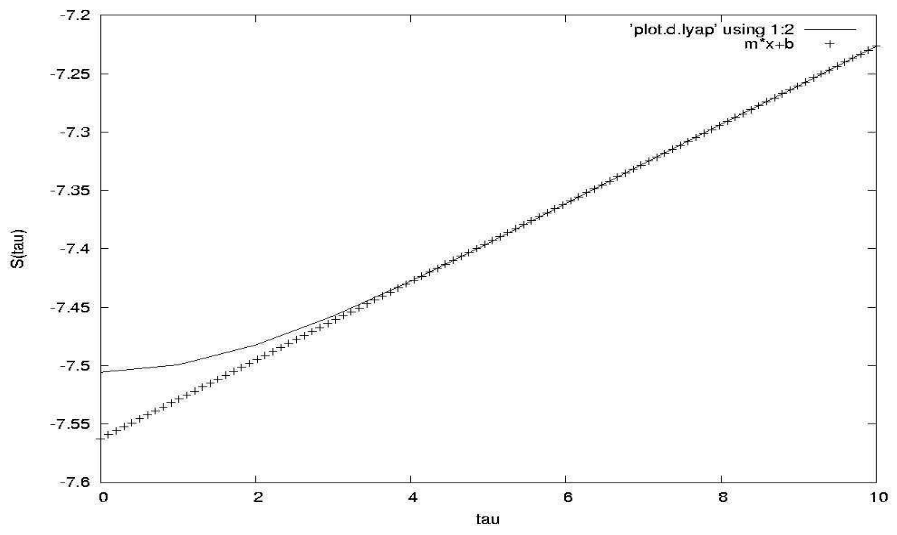

Figure 4.83: Plot of $S(\tau)$ versus τ for straight trajectory in fig(4.4) with delay=2 and m=4

4.6 Conclusion

We have simulated the passage of 1MeV protons in silicon using Runge-kutta method. In the calculations, thermal vibrations of host nuclei have been considered. But energy loss of moving ions due to multiple scattering with electrons in the crystal have been neglected. Using *Tisean* package the MLCE has been determined for every trajectory considered. Table(4.3) gives a consolidated picture of MLCE variation for the different types of trajectories. The results show that hyper channeled trajectories are most chaotic among the various types of motion. Planar and axial channeled trajectories with approximately the same MLCE values are least chaotic. Their MLCE values are about one hundredth of the value for hyper channeled ion. The reason for this can be attributed to the fact that the hyper channeled

Table 4.3: Lyapunov exponents calculated per second for various types of trajectories

ψ_{in}^0	Entry points	θ_{in}^0	Trajectory	λ by TISEAN
0.1	(0,1.92)	30	Hyper channeling	10.24E+14
0.114	(0,1.92)	-30	Quasi-hyper channeling	9.99E+14
0.4	(0,1.92)	-17	Axial cahnneling	0.125E+14
0.538	(0,1.92)	-25	Quasi-axial channeling	0.036E+14
1	(0,0.96)	-83.7	Planar channeling	0.0298E+14
1	(0,0.96)	-83.582	Quasi-planar channeling	0.036E+14
1	(0,1.92)	-40	Random motion	1.5E+14
2	(0,1.92)	0	Straight motion	3.3E+14

ions keep a high distance from the host nuclei throughout its motion. This maximizes the number of host nuclei interacting with the channeled ion, thus increasing the non linearity as well as the collective scattering. For planar and axial channeled trajectories the number of collective scattering is less and this explains their low MLCE values.

For random motion the trajectory is nearly a straight line since no nuclei is involved in any significant interaction with the moving ion. The ion can go on until it meets with a close-encounter scattering with host nucleus. Hence a random moving ion *sees* minimum number of nuclei in the crystal and therefore one should expect a low MLCE. Table(4.3) shows that the MLCE for random motion is at least one order of magnitude less than that for hyper channeled ions. The trajectory in figure(4.4) also keeps a $safe$ distance from a host nucleus. This justifies the low value of MLCE, comparable to random ions for this trajectory also, as can be seen from table(4.3).

Using the latest software $TISEAN$ we have carried out extensive study of chaos in channeling. Results confirm chaos for almost all the ions traversing a crystal. There is also a wide variation in the MLCE among various types of ions moving in a crystal. These results are being communicated[8].

References

[1] Andersen. J. U and Feldman J.C, Phys. Rev. **B1**, 2063(1970)

[2] Rajasekharan. K, and Neelakandan. K, Pramana-J. Phys, **31**, 399(1988)

[3] Alan Wolf, J. B. Swift, Harry L. Swinney and John A. Vastano, Physica, **16D**, 285(1985)

[4] A. Desalvo, S. Giannerini and R. Rosa, Chaos, **16**, 023114(2006)

[5] Sano. M, and Sawada. Y, Phys. Rev. Lett, **55**, 1083(1985)

[6] Eckmann. J. P, Oliffson. K. S, Ruelle. D, and Ciliberto. S, Phys. Rev. A, **34**, 4971(1986)

[7] Brown. R, Bryant. P, and Abarbanel H. D. I, Phys. Rev. A, **43**, 2787(1991)

[8] K. Jeena, K. Rajasekharan and K. Neelakandan, *chaotic behaviour of* 1*Mev proton trajectories in silicon*(under communication)

Appendix-I

Appendix –I

planar.for

```
c       program to calculate Lyapunov exp for planar channelling
c       n= number of variables in nonlinear map
c       nn=n*(n+1)=total number of variables

        parameter(n=2,nn=6)
        implicit double precision(a-h,o-z)
        external fcn
        external vect
        dimension x(nn),xnew(nn),v(nn),tot(n)
        dimension znorm(n),gsc(n)

        data io,irate,mf,iopt/200,3,1000,1/
        data ivst,ivfn/1,3/

c       initial conditions for nonlinear maps
        write(*,*)'Enter initial p and q:'
        read(*,*)v(2),v(1)

        open(unit=1,file='pq.d',status='new')
        open(unit=3,file='vect.d',status='new')
c       initial conditions for linearised maps
        do 10 i=n+1,nn
          v(i)=0.0
10      continue
        tme=0.0
        iv = 0
        do 20 i=1,n
          v((n+1)*i)=1.0
          tot(i)=0.0
20      continue

        do 100 m=1,mf
         do 25 j=1,irate
         do 26 i=1,nn
          x(i)=v(i)
26      continue
         iv = iv + 1
                  if(iv.ge.ivst.and.iv.le.ivfn) call vect(x,xnew,nn)
         if(iopt.eq.1) write(1,*)(x(ii), ii = 1,n)
         call fcn(x,xnew,n)
         do 27 i=1,nn
         v(i)=xnew(i)
27      continue
         tme=tme+1.0
25      continue

c       construct a new orthonormal basis by gram schmidt method
c       normalise first vector
          znorm(1)=0.0
          do 38 j=1,n
          znorm(1)=znorm(1)+v(n*j+1)**2
```

```
38         continue
            znorm(1)=dsqrt(znorm(1))
            do 40 j=1,n
             v(n*j+1)=v(n*j+1)/znorm(1)
40         continue

c          generate the new orthonomal set of vectors

c          generate j-1 gsr coefficients
            do 80 j=2,n
            do 50 k=1,j-1
            gsc(k)=0.0
            do 50 l=1,n
            gsc(k)=gsc(k)+v(n*l+j)*v(n*l+k)
50         continue

c          construct a new vector
            do 60 k=1,n
            do 60 l=1,j-1
            v(n*k+j)=v(n*k+j)-gsc(l)*v(n*k+l)
60         continue

c          calculate the vectors norm
            znorm(j)=0.0
            do 70 k=1,n
            znorm(j)=znorm(j)+v(n*k+j)**2
70         continue
            znorm(j)=dsqrt(znorm(j))

c          normalise the new vector
            do 80 k=1,n
            v(n*k+j)=v(n*k+j)/znorm(j)
80         continue

c          update running vector magnitudes
           do 90 k=1,n
           tot(k)=tot(k)+dlog(znorm(k))/log(2.0)
90        continue

c          normalise exponent and print every io iterations
            if(mod(m,io).ne.0) go to 100
            write(*,334)tme,(tot(k)/tme,k=1,n)
100        continue
           close(unit=1)
           close(unit=3)

           open(unit=2,file='lambda.d',status='new')
           write(2,*) (tot(k)/tme,k=1,n)
           close(unit=2)

334        format(1x,f7.1,3x,f8.4,3x,f8.4)
335        format(1x,f6.2,3x,f6.2)

           end
```

```
        subroutine fcn(x,xnew,n)
        implicit double precision(a-h,o-z)
        dimension x(n),xnew(n),alpha(3),beta(3)
        data alpha/0.1,0.55,0.35/,beta/6.0,1.2,0.3/
c       a - screening length and dp - inter planar distance (both in Ang.)
        data ak/0.7007/, np/3/, a/0.109/, dp/2.039/,ifist/1/


c       nonlinear map equations
        if (ifist.eq.1)then
         np = (np-1)*2 + 1
         ifist = 0
        endif
        sum1 = 0.0
        sum2 = 0.0
        do 10 j = -np, np, 2
                do 10 i = 1,3
         r = x(1)-j
         if (r.lt.0) then
         idir = -1
         else
         idir = 1
         endif
         arg = -beta(i)*dp*dabs(r)/(2.0*a)
         term1 = idir*alpha(i)*dexp(arg)
         term2 = (-beta(i)*dp/(2.0*a))*term1
         sum1 = sum1 + term1
         sum2 = sum2 + term2
10      continue
        xnew(2) = x(2) + ak * sum1
        xnew(1) = x(1) + xnew(2)

c       linearised eqns of motion
        sum2 = ak*sum2
        xnew(3) = (1 + sum2)*x(3) + x(5)
        xnew(4) = (1 + sum2)*x(4) + x(6)
        xnew(5) = sum2*x(3) + x(5)
        xnew(6) = sum2*x(4) + x(6)

        return
        end

        subroutine vect(x,xnew,nn)
        implicit double precision(a-h,o-z)
        dimension x(nn),xnew(nn)

        write(3,*)(x(i), i = 1,nn)

        return
        end
```

pqgen.c

```
#include <stdio.h>
```

```
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>
#include <graphics.h>
#define MAXNUM 4010
#define MXN 250

int main()
          {
          FILE *fp;
          int i,c,ilim[20],j,jm,ij,titlgth; //size approx. => MAXNUM/MXN
          float q[MAXNUM],p[MAXNUM],qm,pm,qmx,pmx,lamda[2];
          float qq[4],pp[4],qx[4][4],px[4][4],qv,pv;
          float qr,pr,yinc,xinc,zr,yr;  //Maxima of coordinates q,p
          int xf,yf,xc,yc,imax,jmax,key;
//xf,yf - Max of x and y coordinates of the screen.
                        //(xc,yc) - Coordinate of the centre of the screen.
          int gd = DETECT,gm,iqm,im,shifty;
          char lbltiks[4],title[40],c1;
          float dep1,dep2,b = 1.0;
// Inter-atomic distance (in Ang.) in the channelling direction

          clrscr();
//        Reads from PLOT.D
          clrscr();
          fp = fopen("pq.d","r");
          i = -1;
          while(!feof(fp))
                    {
                    i++;
                    fscanf(fp,"%g %g",&q[i],&p[i]);
                    }
          imax = i-1;
          fclose(fp);
//        Reads from lambda.d
          fp = fopen("lambda.d","r");
          fscanf(fp,"%g %g",&lamda[0],&lamda[1]);
          fclose(fp);
//        Determine maxima of all p and q
          i = 1;
          qm = fabs(q[0]);
          pm = fabs(p[0]);
//        qm = 1.0;
//        pm = 0.5;
          do
                    {
                    qm = max(qm,fabs(q[i]));
                    pm = max(pm,fabs(p[i]));
                    i++;
                    }
          while(i<=imax);
//        Labels X
          do
                    {
```

```
//      Select projection plane
                do
                        {
                        clrscr();
                        gotoxy(20,6);
                        printf("SELECT YOUR PROJECTION PLANE:");
                        gotoxy(20,9);
                        printf("1  :: Trajectory in phase space");
                        gotoxy(20,12);
                        printf("2  :: Trajectory in real space");
                        gotoxy(20,15);
                        printf("3  :: QUIT");
                        gotoxy(20,18);
                        printf("Hit 1/2/3 to select : ");
                        c = getch();
                        }
                while(c < 49 || c > 53);
                switch (c)
                        {
                        case '1':
                                initgraph(&gd, &gm,"c:\\tc\\bgi");
 //Initialising graphic screen
                                xf = getmaxx();
                                yf = getmaxy();
                                xc = xf/2;
                                yc = yf/2;
//Drawing coordinate axes
                                setcolor(LIGHTGREEN);
                                line(0,yc,xf,yc);
                                line(xc,0,xc,yf);
//Puts X ticks and Y ticks
                                setcolor(CYAN);
                                settextstyle(DEFAULT_FONT,HORIZ_DIR,0);
                                settextjustify(1,2);
                                im = yc*0.75;
                                shifty = 0.5*textheight("|");
                                for(i=-2; i < 3; i++)
                                        {
                                        if (i != 0)
                                        outtextxy(xc+(i*im),yc-shifty,"|");
// X - ticks
                                        }
                                settextstyle(DEFAULT_FONT,□ERT_DIR,0);
                                settextjustify(1,1);
                                for(i=-2; i < 2; i++)
                                        {
                                        if (i != 0)
                                        outtextxy(xc+0.5*shifty,yc+(i*im),"|");
//Y - ticks
                                        }
//Labelling ticks
                                settextstyle(DEFAULT_FONT,HORIZ_DIR,0);
                                settextjustify(1,2);
                                setcolor(LIGHTGREEN);
                                for(i=1; i < 2; i++)
```

```
                                        {
                                        sprintf(lbltiks,"%2.1f",qm);
                                        outtextxy(xc+(i*im),yc+2*shifty,lbltiks);
                                        }
                                for(i=1; i < 2; i++)
                                        {
                                        sprintf(lbltiks,"%3.1f",-qm);

                                outtextxy(xc(i*im),yc+1.3*textheight(lbltiks),lbltiks);
                                        }
                                settextjustify(0,1);
                                for(i=1; i < 2; i++)
                                        {
                                        sprintf(lbltiks,"%2.1f",pm);
                                outtextxy(xc-1.3*textwidth(lbltiks),yc-(i*im),lbltiks);
                                        }
                                for(i=1; i < 2; i++)
                                        {
                                        sprintf(lbltiks,"-%2.1f",pm);
                                outtextxy(xc-1.3*textwidth(lbltiks),yc+(i*im),lbltiks);
                                        }
//□ rites title
                                setcolor(YELLO□ );
                                settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
                                sprintf(title,"    TRAJECTORY IN");
                                outtextxy(2,3*textheight(title),title);
                                sprintf(title,"   P-Q PHASE SPACE:");
                                outtextxy(2,5*textheight(title),title);
                                setcolor(LIGHTCYAN);
                                settextstyle(SMALL_FONT,HORIZ_DIR,5);
                                sprintf(title,"Initial (p,q) = (%9.7f,%5.2f)",p[0],q[0]);
                        outtextxy(xf-1.1*textwidth(title),2*textheight(title),title);
                                sprintf(title, "Lyapunov Coefficients:");
                        outtextxy(xf-1.5*textwidth(title),3.5*textheight(title),title);
                                for(j=0; j<2; j++)
                                        {
                                        sprintf(title, ": %6.3f",lamda[j]);
                outtextxy(xf-1.5*textwidth(title),(1.5*j+5)*textheight(title),title);
                                        }
//      Labels X and Y axes
                                setcolor(LIGHTCYAN);
                                settextstyle(SMALL_FONT,HORIZ_DIR,5);
                                sprintf(title,"            Q --->");
                                outtextxy(xc+textwidth(title)+5,yc+textheight(title)+5,title);
                                settextstyle(SMALL_FONT,□ERT_DIR,5);
                                sprintf(title,"             P --->");
        outtextxy(xc-2*textheight(title),textwidth(title)/2,title);
//      Plots trajectory
                                i = 0;
                                do
                                        {
                                        qr = q[i]*(yc*0.75/qm)+xc;
//0.75 - number same as in line 32 of SLATE1.C
                                        pr = -p[i]*(yc*0.75/pm)+yc;
                                        putpixel(qr,pr,□ HITE);
```

```
//                                      delay(5);
// Delay by 10ms between two points
                                        i++;
                                        }
                                while(i<=imax);
//      Get data for vectors
                                fp = fopen("vect.d","r");
                                i = -1;
                                while(!feof(fp))
                                        {
                                        i++;
                fscanf(fp,"%g %g %g %g g%g",
                                &qq[i],&pp[i],&qx[i][1],&qx[i][2],&px[i][1],&px[i][2]);
//printf("\n\t%d %f %f %f %f\n",i,qx[i][1],qx[i][2],px[i][1],px[i][2]);
                                        }
                                jmax = i-1;
                                fclose(fp);
//      Draw vectors
                                i = 0;
                                qmx = 10.0*max(fabs(qx[i][1]),fabs(qx[i][2]));
                                pmx = 10.0*max(fabs(px[i][1]),fabs(px[i][2]));
                                do
                                        {
                                        qr = qq[i]*(yc*0.75/qm)+xc;
 //0.75 - number same as in line 32 of SLATE1.C
                                        pr = -pp[i]*(yc*0.75/pm)+yc;
                                        qv = qx[i][1]*(yc/qmx)+qr;
                                        pv = -px[i][1]*(yc/pmx)+pr;
                                        setcolor(LIGHTRED);
                                        line(qr,pr,qv,pv);
                                        qv = qx[i][2]*(yc/qmx)+qr;
                                        pv = -px[i][2]*(yc/pmx)+pr;
                                        setcolor(YELLO□ );
                                        line(qr,pr,qv,pv);
                                        setcolor(LIGHTGRAY);
                                        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
                                        sprintf(lbltiks,"%d",i+1);
                                        outtextxy(qr-1.3*textwidth(lbltiks),pr,lbltiks);
                                        i++;
                                        }
                                while(i<=jmax);

//      Exit instruction
                                setcolor(LIGHTRED);
                                settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
                                sprintf(title,"Hit any key to return to menu");
                                outtextxy(xc-textwidth(title)/2,yf-textheight(title)-5,title);
                                getch();
                                closegraph();
                                break;
                        case '2':
                                if(qm>10)
                                        {
                                        printf("\n\tERROR! Random Trajectory");
                                        printf("\n\tHit any key to exit option\n");
                                        getch();
                                        }
                                else
                                        {
                                        initgraph(&gd, &gm,"c:\\tc\\bgi");       //Initialising
graphic screen
                                        xf = getmaxx();
                                        yf = getmaxy();
                                        xc = xf/2;
                                        yc = yf/2;
//Drawing Atomic planes
                                        setcolor(LIGHTGREEN);
                                        qm = fabs(qm);
                                        iqm = qm;
                                        if(iqm<qm) iqm+= 1;
                                        if((iqm % 2) == 0)
                                                iqm = iqm + 1;
                                        yinc = 1.3*yc/(2*iqm);
                                        for(i=-iqm; i<=iqm; i+=2)
                                                {
                                                line(xf*0.05,yc+i*yinc,xf*0.95,yc+i*yinc);
                                                }
//□ rites title
                                        setcolor(YELLO□ );
                                        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
                                        sprintf(title,"    TRAJECTORY IN");
                                        outtextxy(2,3*textheight(title),title);
                                        sprintf(title,"      REAL SPACE:");
                                        outtextxy(2,5*textheight(title),title);
                                        setcolor(LIGHTCYAN);
                                        settextstyle(SMALL_FONT,HORIZ_DIR,5);
                                        sprintf(title, "Initial (p,q) : (%5.2f,%5.2f)",p[0],q[0]);
                            outtextxy(xf-1.1*textwidth(title),2*textheight(title),title);
//        Plots trajectory
                                        if(imax>MXN)
                                                {
                                                jm = (imax+1)/MXN;
                                                if(((imax+1) % MXN) > 0)
                                                        {
                                                        jm = jm + 1;
                                                        for(j=0; j<jm; j++)
                                                                ilim[j] = j*MXN;
                                                        ilim[jm] = imax - ((imax+1)/MXN);
                                                        }
                                                else
                                                        {
                                                        for(j=0; j<=jm; j++)
                                                                ilim[j] = j*MXN;
                                                        }
                                                }
                                        else
                                                {
                                                ilim[0] = 0;
                                                ilim[1] = imax;
                                                jm = 1;
                                        j = 0;
                                        }
                              j=0;
                              xinc = xf*0.9/ilim[1];
                              key = 0;
                              do
                                        {
                                        setcolor(LIGHTBLUE);

settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
                                        dep1 = b*ilim[j];
                                        setfillstyle(SOLID_FILL,getbkcolor());
                                                  bar(0, yc+iqm*yinc+textheight(title),
xf,yc+iqm*yinc+2*textheight(title));
                                        sprintf(title,"%7.1f",dep1);

outtextxy(xf*0.005,yc+iqm*yinc+textheight(title),title);
                                        dep2 = b*ilim[j+1];
                                        sprintf(title,"%7.1f",dep2);
                                                  outtextxy(xf*0.95-
textwidth(title),yc+iqm*yinc+textheight(title),title);
                                        sprintf(title,"Depth (in Ang.) --->");
outtextxy(xc-textwidth(title)/2,yc+iqm*yinc+textheight(title),title);
                                        setcolor(□ HITE);
                                        zr = xf*0.05;
                                        yr = -(q[ilim[j]-1]*yinc)+yc;
                                        if(j == 0) yr = -(q[ilim[j]]*yinc)+yc;
                                        bar(0,yc-yinc+1,xf,yc+yinc-1);
                                        moveto(zr,yr);
                                        for(i=ilim[j]; i<=ilim[j+1]; i++)
                                                  {
                                                  zr = zr + xinc;
                                                  yr = -(q[i]*yinc)+yc;
                                                  lineto(zr,yr);
                                                  }
                              setcolor(LIGHTCYAN);
                              settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
                              sprintf(title,"Hit:-          ");
                              titlgth = textwidth(title)/2;
                    outtextxy(xc-titlgth,yf-6.0*textheight(title)-5,title);
                              if(j > 0)
                                        {
                                        sprintf(title,"    1 for BACK");
                    outtextxy(xc-titlgth,yf-4.5*textheight(title)-5,title);
                                        }
                              else
                                        {
          bar(0, yf-5.5*textheight(title), xf,yf-4*textheight(title));
                                        }
                              if(j < jm-1)
                                        {
                                        sprintf(title,"    2 for FOR□ ARD");
                    outtextxy(xc-titlgth,yf-3.0*textheight(title)-5,title);
                                        }
                              else
```

```
                                        {
                    bar(0,yf-4*textheight(title), xf,yf-2.5*textheight(title));
                                        }
                                sprintf(title,"    Any key to return to menu");
                        outtexttxy(xc-titlgth,yf-1.5*textheight(title)-5,title);
keyin:
                                        c1 = getch();
                                        if(c1=='1')
                                                {
                                                if(j==0)key = 1;
                                                j--;
                                                }
                                        else if(c1=='2')
                                                {
                                                if(j==jm-1)key = 1;
                                                j++;
                                                }
                                        else
                                                key = 1;
                                        }
                                while(key == 0);
//      Exit instruction
                                closegraph();
                                }
                        break;
                case '3':
//                      QUIT
                        clrscr();
                }
        }
        while(c != '3');

//      End
        return 0;
        }
```

gax.for

```
c       Programme to calculate lyapunov spectrum (exponents)
c       Programme generates trajectories by computing accl. in all three
c       directions but for Lyapunov exponents consideres only four dimensional
c       phase space namely qx, qy, psix and psiy.
c       n=number of nonlinear odes
c       nn=n*(n+1)=total number of odes
c       nn1=nn+1(for compilers that start arrays at element 0)


        block data
c       implicit double precision(a-h,o-z)
        common /cnvr/acr
        common /fcndat/vt
        common /option/rcrt,iopt
        data iopt/1/
        data rcrt,acr/1.0e-03,1.0e+10/
```

```
      end
      program ode
      parameter(n=6,mm=4,nn=42,nn1=43,mm1=22)
      parameter(pi=3.14159265)
      external fcn
      double precision znorm,cum
      common /cnvr/acr
      common /fcndat/vt
      common /option/rcrt,iopt
      dimension y(nn1),yprime(nn1),v(nn1),A(nn1),B(nn1),C(nn1)
c     dimension yprime(nn1),v(nn1),A(nn1),B(nn1),C(nn1)
      dimension D(nn1),cum(n),znorm(n),gsc(n),ci(nn1),tm(n)

c    NSTEP is the total number of reorthonormalization steps that will be
c          performed. Set this to any very large value.
c    IRATE is the number of numerical integration time steps per
c          reorthonormalization
c    STPSZE is the integration stepsize in "seconds". Choose a # of seconds
c          small compared to the 'mean orbital period'.
c    IO is the rate at which output is generated. Rather than outputing
c          the spectrum for every IOth reorthonorm.


      open(unit=1,file='ion.d',status='old')
      read(1,*)z1,pm
      close(unit=1)
      write(*,111)
111   format(1x,'Enter NSTEP, STEP SIZE, IRATE, IO :')
      read(*,*)nstep,stpsze,irate,io


c     Initial conditions for nonlinear ODEs
c     (** Choose within the system's basin of attraction **)
c     Initial position coordinates
      write(*,*)'Enter energy(Mev),psi0,theta(deg)'
        read(*,*)e,psi,theta
        e1=e*1.602177e-13
        psi1=psi*pi/180.
        theta1=theta*pi/180.
        pv=sqrt(2.*e1/pm)
        tv=pv*sin(psi1)
        vzii=pv*cos(psi1)
        sth=sin(theta1)
        cth=cos(theta1)
        if(abs(sth).lt.1.0e-06)sth=0
        if(abs(cth).lt.1.0e-06)cth=0
        vxii=-tv*sth
        vyii=tv*cth
      v(1) = 0.
c     v(2) = 1.92
      v(2)=0.96
      v(3) = 0.0
c    Initial velocity coordinates
      v(4) = vxii
      v(5) = vyii
```

```
          v(6) = vzii

        xm = amax1(v(4),v(5))*nstep*stpsze*acr
        vt = sqrt(v(4)*v(4)+v(5)*v(5))
        vzm = sqrt(vt*vt+v(6)*v(6))
        write(*,*)'xm=',xm
        write(*,*)'vx,vy,vt=',v(4),v(5),vt
        write(*,*)'vz,vzm=',v(6),vzm
        open(unit=2,file='plo.d',status='new')
        write(2,*)'NSTEP, STEP SIZE, IRATE, IO:'
        write(2,*)'              ',nstep,stpsze,irate,io
        write(2,*)'e,psi,theta:',e,psi,theta
        if(iplot.eq.1)then
                open(unit=1,file='plot.d',status='new')
                write(1,*)v(3),v(1),v(2),v(4),v(5),v(6)
        endif
        tme = 0.0

c       Initial conditions for linearized ODEs

        do 10 i = n+1,mm1
        v(i) = 0.0
10      continue
        do 20 i = 1,mm
        v((mm+1)*i+2) = 1.0
        cum(i) = 0.0
20      continue

c-------------          Computation Begins       -------------------
        do 100 m = 1,nstep

c       Integration of ordinary differential equation begins (for IRATE steps)
                do 25 j = 1,irate

c       Application of Runge-Kutta for one step of integraton
c***************************************************************
          do 26 i = 1,mm1
         y(i) = v(i)
26       continue
          t = tme
          call fcn(t,y,yprime)

          do 27 i = 1,mm1
         A(i) = yprime(i)
27       continue
         do 28 i = 1,3
         y(i) = v(i) + (stpsze*acr*A(i))/2.0
28       continue
         do 128 i = 4,mm1
         y(i) = v(i) + (stpsze*A(i))/2.0
128       continue
          t = tme + stpsze/2.0
         call fcn(t,y,yprime)
         do 29 i = 1,mm1
         B(i) = yprime(i)
```

```
29      continue
        do 30 i = 1,3
        y(i) = v(i) + (stpsze*acr*B(i))/2.0
30      continue
        do 130 i = 4,mm1
        y(i) = v(i) + (stpsze*B(i))/2.0
130       continue
          t = tme + stpsze/2.0
          call fcn(t,y,yprime)
        do 31 i = 1,nn
        C(i) = yprime(i)
31      continue
        do 32 i = 1,3
        y(i) = v(i) + (stpsze*acr*C(i))
32      continue
        do 132 i = 4,mm1
        y(i) = v(i) + (stpsze*C(i))
132    continue
          t = tme + stpsze
          call fcn(t,y,yprime)
        do 33 i = 1,mm1
        D(i) = yprime(i)
33      continue

c*****************************************************************
          do 34 i = 1,3
          v(i) = v(i)+ stpsze*acr*(A(i)+D(i)+2.0*(B(i)+C(i)))/6.0
34      continue
        do 134 i = 4,mm1
        v(i) = v(i) + stpsze * (A(i) +D(i) + 2.0*(B(i)+C(i)))/6.0
134    continue
c      One step of Runge-Kutta ends here
c*****************************************************************
          tme = tme + stpsze
          if(iplot.eq.1)write(1,*)v(3),v(1),v(2), v(4),v(5),v(6)

25      continue

c      Integration of ODE over

c      Construct new orthonormal basis by Gram-Schmidt

c      Normalize first vector
          znorm(1) = 0.0
        do 38 j = 1,mm
        znorm(1) = znorm(1) + v(mm*j+3)**2
38      continue
        znorm(1) = dsqrt(znorm(1))
        do 40 j = 1,mm
        ci(mm*j+3) = v(mm*j+3)
        v(mm*j+3) = v(mm*j+3)/znorm(1)
40      continue
c      Generate the new (mm-1) orthonormal set of vectors
        do 80 j = 2,mm
c      Generate (j-1) GSR coefficients
```

```
         do 50 k = 1,(j-1)
        gsc(k) = 0.0
         do 50 l = 1,mm
         gsc(k) = gsc(k) + v(mm*l+j+2)*v(mm*l+k+2)
50      continue
c       Construct a new vector
         do 60 k = 1,mm
        do 60 l = 1,j-1
         v(mm*k+j+2) = v(mm*k+j+2) -gsc(l)*v(mm*k+l+2)
60       continue
c       Calculate the vector's norm
        znorm(j) = 0.0
         do 70 k = 1,mm
         znorm(j) = znorm(j) + v(mm*k+j+2)**2
70      continue
        znorm(j) = dsqrt(znorm(j))
c       Normalize the new vector
         do 80 k = 1,mm
        v(mm*k+j+2) = v(mm*k+j+2)/znorm(j)
80      continue
c       Update running vector magnitudes
        do 90 k = 1,mm
        tmm = dlog(znorm(k))/dlog(2.0d0)
        cum(k) = cum(k) + tmm
c        cum(k) = cum(k) + alog(znorm(k))/alog(2.0)
c       write(*,*)'k,znorm : ',k,znorm(k)
90      continue
         write(*,335)  (cum(k)/tme, k = 1,mm)
        write(2,335)  (cum(k)/tme, k = 1,mm)
100      continue
         if(iplot.eq.1)then
        write(1,*)'100.0 100.0 100.0 100.0 100.0 100.0'
        close(unit=1)
        close(unit=2)
         endif
c---------------       Computation Over       -------------------

55       format(5x,'gsc(',i2,') = gsc(',i2,') + v(',i2,')*v(',i2,')')
61      format(3x,'Components of ',i1,'th □ector:')
62      format(3x,'v(',i2,') = v(',i2,') - gsc(',i2,')v(',i2,')')
63      format(3x,'znorm(',i1,') = znorm(',i1,') + v(',i2,')ˆ2')
334     format(1x,e13.6,2x,e13.6,2x,e13.6,2x,e13.6,2x, e13.6,2x,e13.6,2x,e13.6)
335     format(1x,e13.6,2x,e13.6,2x,e13.6,2x,e13.6)
120     format(1x,e13.6,3x,e13.6,3x,e13.6,3x,e13.6)
         stop
         end
c---------------------------------------------------------------

c       Sub-routine containing nonlinear functions

         subroutine fcn(t,y,yprime)
c        implicit double precision(a-h,o-z)
c        real acr,y(43),ax,ay,az
         common /cnvr/acr
         common /fcndat/vt
         common /pda/cj(10),axbdx,axbdy,axbdz,aybdx,aybdy,aybdz,
                                                   azbdx,azbdy,azbdz
         common /posn/xu1,yu1,zu1
         dimension y(43), yprime(43)
         data apx/5.431/, apy/3.84/
c       dimension yprime(43)

c       Nonlinear Lorenz equations
         yprime(1) = y(4)
         yprime(2) = y(5)
         yprime(3) = y(6)
         vz2 = y(6)*y(6)
         call field(y(1),y(2),y(3),vz2,ax,ay,az,*20,*30)
         yprime(4) = ax
         yprime(5) = ay
         yprime(6) = az
         call jacobs(vz2)
         do 10 i = 0,3
         yprime(7+i)  = y(15+i)*vt*acr/apx
        yprime(11+i) = y(19+i)*vt*acr/apy
         yprime(15+i) = (cj(1)*y(7+i) + cj(2)*y(11+i)) *(apx/(vt*acr))
         yprime(19+i) = (cj(4)*y(7+i) + cj(5)*y(11+i)) *(apy/(vt*acr))
10     continue
         go to 40
20     continue
30     write(*,*)'Unexpected error in field calculations'
         stop
40     continue
         return
         end
```

eloss.for

```
        subroutine elos(x,y,v2,zlos)
        zlos=0
        return
        end
```

closs.for

```
        subroutine closf(fx,fy,fz)
        parameter(id=23,pi=3.14159265)
        common /posn/xu1,yu1,zu1
c      The above line supplies the ion  position from calling programmes
c      such as dmtst.for or bdimt.for

        common /podis/x2(50),y2(50),za2,m,n
        common /jacbs/z1,zpmc,dist,dist2,distc2,xq,dxq,con,s
        character*1 qk,qi(id),qj(id)
        dimension idiv(id),jdiv(id)
        dimension xi(50),yi(50),xsq(id),ysq(id)
        data engy/1.0e+04/
c       Crystal data :- u1 value for 293K (Ref.- Gemmel,1974)
        data u1,br,z2,a/0.075,0.529177,14.0,5.43089/
        data ec/1.602177e-19/,eps/8.854188e-12/
```

```
c          calculating/assigning constant terms
c          if(ien.eq.0)then
c          ien=1

c        ------ Inputs ion data ---------
         open(unit=4,file='ion.d',status='old')
         read(4,*)z1,pm
         close(unit=4)

           open(unit=4,file='dftem.d',status='old')
           read(4,*)nf,distc
           close(unit=4)

c        Specification data for execution of this programme:
          dist=3.0
           distc2=distc*distc
          zpmc=z1*(1.672623e-27)/pm
           xq=a*0.25
           idim=(a+dist)/xq
           idim=idim*2+1
           if(idim.gt.id)then
            write(*,*)'Increase value of id to',idim
           stop
           endif
           if(id.gt.50)then
           write(*,*)'Error !; Change size of xi,yi,and zi'
           write(*,*)'          Size should be at least id in'
           write(*,*)'          the parameter statement'
           stop
           endif
           dxq=1./xq
           dist2=dist*dist
           cond=4.0*pi*eps
          if(z1.eq.1.0)then
          at=0.8853*br/(z2**0.3333333)
          else
             sqz=sqrt(z1)+sqrt(z2)
          at=0.8853*br/(sqz**0.6666667)
          endif
          con=3.*at*at
           s=z1*z2*(ec/pm)*(ec/cond)*(1.0e+20)
c          endif

c        -------    Field calculation begins    ------------
c        Obtains upper and lower bounds of i,j,k
           xll=xu1-dist
           xul=xu1+dist
           ixl=xll*dxq
           ixu=xul*dxq
           yll=yu1-dist
           yul=yu1+dist
           jyl=yll*dxq
           jyu=yul*dxq
           zll=zu1-dist
           zul=zu1+dist
           kzl=zll*dxq
           kzu=zul*dxq
           sumx=0
           sumy=0
           sumz=0
           mn=0

c        Generating the crystal structure

           l=0
           do 40 k=kzl,kzu
           l=l+1
           m=0
           call fact(k,qk,kdiv)
           zi=k*xq
           za2=zu1-zi
           zsq=za2*za2
           do 50 j=jyl,jyu
           m=m+1
           if(l.eq.1)then
           call fact(j,qj(m),jdiv(m))
           yi(m)=j*xq
           y2(m)=yu1-yi(m)
           ysq(m)=y2(m)*y2(m)
           endif
           if(qk.ne.qj(m)) go to 50
           n=0
           do 60 i=ixl,ixu
           n=n+1
           if(mn.eq.0)then
           call fact(i,qi(n),idiv(n))
           xi(n)=i*xq
           x2(n)=xu1-xi(n)
           xsq(n)=x2(n)*x2(n)
           endif
c        Selelects valid atomic sites
           if(qk.ne.qi(n)) go to 60
           if(kdiv.eq.4) then
           if(jdiv(m).ne.idiv(n)) go to 60
           else
           if(jdiv(m).eq.idiv(n)) go to 60
           endif

c        Checks whether valid sites are within limiting 'dist'
           dint2=zsq+ysq(m)+xsq(n)
           if(dint2.gt.dist2)then go to 60
c         write(*,*)'Atom very far away'
           endif
c         write(*,*)'xyz',x2(n),y2(m),za2
c         Determines contribution to the field
          if(dint2.lt.distc2)then
c         write(*,*)'clos'
           call fted(afx,afy,afz)
           if((x2(n).gt.0).and.(afx.lt.0))afx=-afx
           if((x2(n).lt.0).and.(afx.gt.0))afx=-afx
           if((y2(m).gt.0).and.(afy.lt.0))afy=-afy
           if((y2(m).lt.0).and.(afy.gt.0))afy=-afy
           if((za2.gt.0).and.(afz.lt.0))afz=-afz
           if((za2.lt.0).and.(afz.gt.0))afz=-afz
          if(z1.ne.1.0)then
          afx=afx*zpmc
          afy=afy*zpmc
          afz=afz*zpmc
          endif
c         write(*,*)'ax,ay,az',afx,afy,afz
          else
c         write(*,*)'far'
           den1=sqrt(dint2)*dint2
           den2=dint2+con
           pg=(1./den1)-(1./(den2*sqrt(den2)))
           afx=x2(n)*pg*s
           afy=y2(m)*pg*s
           afz=za2*pg*s
           endif
c        write(*,*)afx,afy,afz
           sumx=sumx+afx
           sumy=sumy+afy
           sumz=sumz+afz
60       continue
           mn=1
50       continue
40       continue

c        Obtaining the final accl.
           fx=sumx
           fy=sumy
           fz=sumz
           return
           end

c        Subroutine to determine whether the given integer 'i'
c        is odd or even and whether (i-1) or i is divisible by 2 or 4

           subroutine fact(i,qi,idiv)
           character*1 qi
           if((i-(i/2)*2).eq.0)then
           qi='e'
           idiv=multip(i)
           else
           qi='o'
           idiv=multip(i-1)
           endif
           return
           end

c        Function subprogramme to find whether
c        given 'i' is multiple of 2 or 4

           function multip(i)
           if((i-(i/4)*4).eq.0)then
           multip=4
           else
           multip=2
           endif
           return
           end


           subroutine fted(afx,afy,afz)
           parameter(jd=10)
           common /podis/x2(50),y2(50),za2,m,n
           dimension fx(jd,jd,jd),fy(jd,jd,jd),fz(jd,jd,jd)

           if(ien.eq.0)then
          ien=1
           open(unit=4,file='dftem.d',status='old')
           read(4,*)nf,distc
           id=ifix(sqrt(float(nf)))
           do 10 i=1,id
           do 20 j=1,id
          read(4,*)(fx(i,j,k),fy(i,j,k),fz(i,j,k), k=1,id)
20      continue
10      continue
          close(unit=4)
           a10=distc/id
           ainc=a10*0.1
           endif
           x=x2(n)
           if(x.lt.0)x=-x
           y=y2(m)
           if(y.lt.0)y=-y
           z=za2
           if(z.lt.0)z=-z
           i=1+(x/a10)
           if(i.gt.id)i=id
           j=1+(y/a10)
           if(j.gt.id)j=id
           k=1+(z/a10)
           if(k.gt.id)k=id
           afx=fx(i,j,k)
           afy=fy(i,j,k)
           afz=fz(i,j,k)
           return
           end
```

jacobs.for

```
c       ------ Subprogramme to generate the Jacobians ---------------

          subroutine jacobs(vz2)
          parameter(id=23,pi=3.14159265)
          common /pda/cj(10),axbdx,axbdy,axbdz,aybdx,aybdy,aybdz,
                                                          azbdx,azbdy,azbdz
          common /posn/xu1,yu1,zu1
          common /jacbs/z1,zpmc,dist,dist2,distc2,xq,dxq,con,s
```

```
c         The above line supplies the ion  position from calling programmes
c         such as dmtst.for or bdimt.for
          common /podis/x2(50),y2(50),za2,m,n
          character*1 qk,qi(id),qj(id)
          dimension idiv(id),jdiv(id)
          dimension xi(50),yi(50),xsq(id),ysq(id)
          data engy/1.0e+04/
c         Crystal data :- u1 value for 293K (Ref.- Gemmel,1974)
          data u1,br,z2,a/0.075,0.529177,14.0,5.43089/
          data ec/1.602177e-19/,eps/8.854188e-12/
          xll=xu1-dist
          xul=xu1+dist
          ixl=xll*dxq
          ixu=xul*dxq
          yll=yu1-dist
          yul=yu1+dist
          jyl=yll*dxq
          jyu=yul*dxq
          zll=zu1-dist
          zul=zu1+dist
          kzl=zll*dxq
          kzu=zul*dxq
          axbdx=0
          axbdy=0
          axbdz=0
          aybdx=0
          aybdy=0
          aybdz=0
          azbdx=0
          azbdy=0
          azbdz=0
          mn=0

c       Generating the crystal structure
c       Generates all combinations of i,j,k within the lower & upper bounds
c       and determines their factors
          l=0
          do 40 k=kzl,kzu
          l=l+1
          m=0
          call fact(k,qk,kdiv)
          zi=k*xq
          za2=zu1-zi
          zsq=za2*za2
          do 50 j=jyl,jyu
          m=m+1
          if(l.eq.1)then
          call fact(j,qj(m),jdiv(m))
          yi(m)=j*xq
          y2(m)=yu1-yi(m)
          ysq(m)=y2(m)*y2(m)
          endif
          if(qk.ne.qj(m)) go to 50
          n=0
          do 60 i=ixl,ixu
```

```
      n=n+1
      if(mn.eq.0)then
      call fact(i,qi(n),idiv(n))
      xi(n)=i*xq
      x2(n)=xu1-xi(n)
      xsq(n)=x2(n)*x2(n)
      endif

c    Selelects valid atomic sites
      if(qk.ne.qi(n)) go to 60
      if(kdiv.eq.4) then
      if(jdiv(m).ne.idiv(n)) go to 60
      else
      if(jdiv(m).eq.idiv(n)) go to 60
      endif
      dint2=zsq+ysq(m)+xsq(n)
      if(dint2.gt.dist2)then
      go to 60
c     write(*,*)'Atom very far away'
      endif
c     Determines contribution to the Jacobian; axbdx, axbdy etc. are
c     the Jacobian elements in the 'Unit Cell' frame
      if(dint2.lt.distc2)then
c     if(idata.eq.0)write(5,*)'clos'
c     write(*,*)'clos'
      call getf(afx,afy,afz)
      if(x2(n).ge.0)then
      x2(n)=x2(n)+0.1
      ds=0.1
      else
      x2(n)=x2(n)-0.1
      ds=-0.1
      endif
      call getf(afx1,afy1,afz1)
      dfxx=(afx1-afx)/ds
      dfyx=(afy1-afy)/ds
      dfzx=(afz1-afz)/ds
      if(x2(n).ge.0)then
      x2(n)=x2(n)-0.1
      else
      x2(n)=x2(n)+0.1
      endif
      if(y2(m).ge.0)then
      y2(m)=y2(m)+0.1
      ds=0.1
      else
      y2(m)=y2(m)-0.1
      ds=-0.1
      endif
      call getf(afx1,afy1,afz1)
      dfxy=(afx1-afx)/ds
      dfyy=(afy1-afy)/ds
      dfzy=(afz1-afz)/ds
      if(y2(m).ge.0)then
      y2(m)=y2(m)-0.1
```

```
          else
          y2(m)=y2(m)+0.1
          endif
          if(za2.ge.0)then
          za2=za2+0.1
          ds=0.1
        else
        za2=za2-0.1
        ds=-0.1
        endif
        call getf(afx1,afy1,afz1)
        dfxz=(afx1-afx)/ds
        dfyz=(afy1-afy)/ds
        dfzz=(afz1-afz)/ds
        if(za2.ge.0)then
        za2=za2-0.1
        else
        za2=za2+0.1
        endif
        else
        r3=sqrt(dint2)*dint2
        r5=r3*dint2
          rc2=dint2+con
        rc3=rc2*sqrt(rc2)
        rc5=rc3*rc2
        dfxx=(1./r3)-(1./rc3)-(3.0*x2(n)*x2(n)/r5)+ (3.0*x2(n)*x2(n)/rc5)
          dfxx=dfxx*s
        adf=(3.0/r5)-(3.0/rc5)
        dfxy=-x2(n)*y2(m)*s*adf
        dfxz=-x2(n)*za2*s*adf
        dfyy=(1./r3)-(1./rc3)-(3.0*y2(m)*y2(m)/r5)+ (3.0*y2(m)*y2(m)/rc5)
          dfyy=dfyy*s
        dfyz=-y2(m)*za2*s*adf
        dfyx=dfxy
        dfzz=(1./r3)-(1./rc3)-(3.0*zsq/r5)+ (3.0*zsq/rc5)
          dfzz=dfzz*s
        dfzx=dfxz
        dfzy=dfyz
          endif
          axbdx=axbdx+dfxx
        axbdy=axbdy+dfxy
        axbdz=axbdz+dfxz
        aybdx=aybdx+dfyx
        aybdy=aybdy+dfyy
        aybdz=aybdz+dfyz
        azbdx=azbdx+dfzx
        azbdy=azbdy+dfzy
        azbdz=azbdz+dfzz
60      continue
          mn=1
50      continue
40      continue
c         Obtaining the final derivative of accl. axbdx, axbdy etc.
c         in the Lab frame, viz. <110>
          call cjacob()
```

```
            return
            end

c           Subprogramme to determine ax,ay and az at a point (x2,y2,za2)
c           due to an atom at (0,0,0)
            subroutine getf(afx,afy,afz)
            common /podis/x2(50),y2(50),za2,m,n
            common /jacbs/z1,zpmc,dist,dist2,distc2,xq,dxq,con,s
            if((abs(x2(n)).ge.1.0).or.(abs(y2(m)).ge.1.0).or. (abs(za2).ge.1.0)) then
           dint2=za2*za2+y2(m)*y2(m)+x2(n)*x2(n)
           den1=sqrt(dint2)*dint2
           den2=dint2+con
           pg=(1./den1)-(1./(den2*sqrt(den2)))
           afx=x2(n)*pg*s
           afy=y2(m)*pg*s
           afz=za2*pg*s
            else
           call fted(afx,afy,afz)
          if((x2(n).gt.0).and.(afx.lt.0))afx=-afx
          if((x2(n).lt.0).and.(afx.gt.0))afx=-afx
          if((y2(m).gt.0).and.(afy.lt.0))afy=-afy
          if((y2(m).lt.0).and.(afy.gt.0))afy=-afy
          if((za2.gt.0).and.(afz.lt.0))afz=-afz
          if((za2.lt.0).and.(afz.gt.0))afz=-afz
          if(z1.ne.1.0)then
          afx=afx*zpmc
          afy=afy*zpmc
          afz=afz*zpmc
          endif
          endif
          return
          end
```

tr110.for

```
            subroutine pstran(x,y,z,x1,y1,za1)
c           Lab. coordinates: Z - axis along <110>
c           xc=a*0.125
c           c45=1./sqrt(2.0)
            data xc,c45/0.6788612,0.7071067/
            y45=y*c45
            z45=z*c45
            x1=-y45+z45
            y1=x+xc
            za1=y45+z45
            return
            end

            subroutine ftrans(sumx,sumy,sumz,axi,ayi,azi)
            data c45/0.7071067/
            axi=sumy
            axc45=sumx*c45
            azc45=sumz*c45
            ayi=-axc45+azc45
```

```
      azi=axc45+azc45
      return
      end

c     Subprogramme to transform derivatives of accl. in 'Unit Cell' frame
c     to <110> Lab frame
      subroutine cjacob()
c     Determines spatial derivatives of accl. in (/s^2)

      common /pda/cj(10),axbdx,axbdy,axbdz,aybdx,aybdy,aybdz,
                                                  azbdx,azbdy,azbdz
      data c45/0.7071067/
      c452=c45*c45
      cj(1)=aybdy
c     cj(2)=(aybdz-aybdx)*c45
      cj(2)=(aybdy-aybdx)*c45
      cj(3)=(aybdx+aybdz)*c45
c     cj(4)=(azbdy-axbdy)*c45
      cj(4)=(aybdy-axbdy)*c45
c     cj(5)=(-azbdx+axbdx+azbdz-axbdz)*c452
      cj(5)=(-aybdx+axbdx+aybdy-axbdy)*c452
      cj(6)=(azbdx-axbdx+azbdz-axbdz)*c452
      cj(7)=(axbdy+azbdy)*c45
      cj(8)=(-axbdx-azbdx+axbdz+azbdz)*c452
      cj(9)=(axbdx+azbdx+axbdz+azbdz)*c452
c     Next one is zero if eloss = 0
      cj(10)=0
      return
      end
```

gnplot.c

```
/*    ------  Main Programme of Project "GNPLOT.PRJ"  --------
      Project prepared from TRBASGN which processes o/p from GAX.FOR only.
      This one handles o/p of both STRAJ.FOR and GAX.FOR.
      STARJ.FOR:-
              Plots trajectory in real space transverse to channel axis.
      GAX.FOR:-
              Programme to plot projection of ion trajectory in 4-D phase
              space to 2-D r1-vr1, r2-vr2 and r3-vr3 OR x-vx, y-vy and z-vz.
*/
#include <stdio.h>
//#include <conio.h>
#include <stdlib.h>
#include <math.h>
//#include <dos.h>
#include <graphics.h>
//#include <errno.h>
//#include <string.h>

float x,y,z,vx,vy,vz;
float x1,y1;
float xm,ym,zm,vxm,vym,vzm;  //Maxima of coordinates x,y,z,vx etc.
int xf,yf,xc,yc;  //xf,yf - Max of x and y coordinates of the screen.
```

```
                        //(xc,yc) - Coordinate of the centre of the screen
float r1,r2;

void slate();
void projr();
void projx();
void projy();
void projz();
void depth();
void projun();
void conv();
void gentrn(float x, float y);
int main()
          {
          unsigned long int i;
          int c,sorc;
          char string[200];
          FILE *fp;
          clrscr();
//        Determines the source of PLOT.D
          fp = fopen("plot.d","r");
          do c=fgetc(fp); while(c < 45);
          fclose(fp);
          if(c>48) sorc = 1; else sorc = 2; //1 - o/p of STRAJ; 2 - o/p of GAX
//        Reads from PLOT.D
          i=0;
          fp = fopen("plot.d","r");
          if (sorc == 1)
                    {
                    fscanf(fp,"%d",&c);
                    if(fscanf(fp,"%g %g",&z,&x));      //Final depth, theta
                    else
                              {
                              printf("ERROR in input on second line;\nSystem message:
                              %s\n",i,strerror(errno));
                              getch();
                              exit(1);
                              }
                    if(fscanf(fp,"%g %g %g",&z,&x,&y));
                    else
                              {
                              printf("ERROR in input at %l;\nSystem message:
                              %s\n",i,strerror(errno));
                              getch();
                              exit(1);
                              }
                    }
          else
                    {
                    if(fscanf(fp,"%f %f %f %f %f %f",&z,&x,&y,&vx,&vy,&vz));
                    else
                              {
                              printf("ERROR in input at %l;\nSystem message:
                              %s\n",i,strerror(errno));
                              getch();
                        exit(1);
                        }
                }

//      Determine maxima of all coordinates
        xm = fabs(x);
        ym = fabs(y);
        if(sorc==2)
                {
                zm = fabs(z);
                vxm = fabs(vx);
                vym = fabs(vy);
                vzm = fabs(vz);
                }
        if(sorc==1)
                {
                do
                        {
                        xm = max(xm,fabs(x));
                        ym = max(ym,fabs(y));
                        i++;
                        if(fscanf(fp,"%g %g %g",&z,&x,&y));
                        else
                                {
                                printf("%f %f\n",x,y);
                                printf("ERROR in input at %l;\nSystem message:
                                %s\n",i,strerror(errno));
                                getch();
                                exit(1);
                                }
                        }
                while(x != 100.0 && y != 100.0);
                }
        else
                {
                do
                        {
                        xm = max(xm,fabs(x));
                        ym = max(ym,fabs(y));
                        zm = max(zm,fabs(z));
                        vxm = max(vxm,fabs(vx));
                        vym = max(vym,fabs(vy));
                        vzm = max(vzm,fabs(vz));
                        i++;
                        if(fscanf(fp,"%f %f %f %f %f %f",&z,&x,&y,&vx,&vy,&vz));
                        else
                                {
                                printf("ERROR in input at %l;\nSystem message:
                                %s\n",i,strerror(errno));
                                getch();
                                exit(1);
                                }
                        }
                while(x != 100.0 && y != 100.0);
                }
        fclose(fp);
//      □ arning screen if no. of atoms to be displayed is too high
        if((xm > 30.0) || (ym > 30.0))
                {
                do
                        {
                        clrscr();
                        gotoxy(20,6);
                        printf("□ ARNING! xm and ym are %f and %f",xm,ym);
                        gotoxy(20,8);
                        printf("Hit 1 - To EXIT");
                        gotoxy(24,10);
                        printf("2 - To continue with another xm/ym");
                        gotoxy(24,12);
                        printf("3 - To continue any how");
                        c = getch();
                        }
                while(c < 49 || c > 51);
                switch (c)
                        {
                        case '1':
                                exit(1);
                                break;
                        case '2':
                                gotoxy(20,14);
                                printf("Enter xm/ym: ");
                                scanf("%f",&xm);
                                ym = xm;
                                break;
                        case  3 :
                                clrscr();
                        }
                }
//      Menu to plot or 'quit'for GAX alone
        if(sorc==2)
                {
                do
                        {
//      Select projection plane
                        do
                                {
                                clrscr();
                                gotoxy(20,6);
                                printf("SELECT YOUR PROJECTION PLANE:");
                                gotoxy(20,8);
                                printf("1  :: (110) plane");
                                gotoxy(20,10);
                                printf("2  :: X - □X plane");
                                gotoxy(20,12);
                                printf("3  :: Y - □Y plane");
                                gotoxy(20,14);
                                printf("4  :: Z - □Z plane");
                                gotoxy(20,16);
                                printf("5  :: TRAJECTORY IN UNIT CELL");
                                gotoxy(20,18);
```

```
                    printf("6  :: QUIT");
                    gotoxy(20,20);
                    printf("Hit 1/2/3/4/6 to select : ");
                    c = getch();
                    }
                while(c < 49 || c > 54);
                switch (c)
                    {
                    case '1':
                        clrscr();
    //                  gotoxy(20,6);
    //              printf("Enter Maximum x :");
    //              scanf("%f",&xm);
                        projr();
                        break;
                    case '2':
                        clrscr();
    //                  gotoxy(20,6);
    //              printf("Enter Maximum □elocity in X direction :");
    //              scanf("%f",&vxm);
                        slate();
                        projx();
                        break;
                    case '3':
                        clrscr();
    //                  gotoxy(20,6);
    //              printf("Enter Maximum □elocity in Y direction :");
    //              scanf("%f",&vym);
                        slate();
                        projy();
                        break;
                    case '4':
                        clrscr();
    //                  gotoxy(20,6);
    //              printf("Enter Maximium □elocity in Z direction :");
    //              scanf("%f",&vzm);
                        slate();
                        projz();
                        break;
                    case '5':
                        clrscr();
                        projun();
                        break;
                    case '6':
    //                  QUIT
                        clrscr();
                    }
                }
            while(c != '6');
            }
        else
            {
            clrscr();
            projr();
            }
```

```
//      End
        return 0;
        }
//      -------------       END OF MAIN-------------------------

//      -------------       SUBROUTINE 'PROJX()'       -----------------
//      Subprogramme to plot in X-□X plane

void projx()
        {
        int som;
        float xr,vxr;
        char title[30];
        FILE *fp;

//□ rites title
        setcolor(YELLO□ );
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"    PROJECTION");
        outtextxy(2,1.5*textheight(title),title);
        sprintf(title,"    IN X-□X PLANE:");
        outtextxy(2,3.5*textheight(title),title);

//      Labels X and Y axes
        setcolor(LIGHTCYAN);
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        sprintf(title,"Generalized X --->");
        outtextxy(xc+textwidth(title)+5,yc+textheight(title)+5,title);
        settextstyle(SMALL_FONT,□ERT_DIR,5);
//      settextjustify(LEFT_TEXT,RIGHT_TEXT);
        sprintf(title,"Generalized □X --->");
        outtextxy(xc-textheight(title)*3,textwidth(title)/2,title);

//      Plots trajectory
        fp = fopen("plot.d","r");
        fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
        do
                {
                vxr = -(vx/vxm)*(yc/2)+yc;
                gentrn(x,y);
                r1 = r1*(yc/2)+xc;
                putpixel(r1,vxr,□ HITE);
//              delay(100);         // Delay by 100ms between two points
        fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
                }
        while(x != 100.0 && y != 100.0);
        fclose(fp);

//      Exit instruction
        setcolor(LIGHTRED);
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"Hit any key to return to menu");
        outtextxy(xc-textwidth(title)/2,yf-textheight(title)-5,title);
        getch();
        closegraph();
        }
//      -------------       SUBROUTINE 'PROJY()'        -----------------
//      Subprogramme to plot in Y-□Y plane

void projy()
        {
        float yr,vyr;
        char title[30];
        FILE *fp;

//□ rites title
        setcolor(YELLO□ );
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"    PROJECTION");
        outtextxy(2,1.5*textheight(title),title);
        sprintf(title,"    IN Y-□Y PLANE:");
        outtextxy(2,3.5*textheight(title),title);

//      Labels X and Y axes
        setcolor(LIGHTCYAN);
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        sprintf(title,"Generalized Y --->");
        outtextxy(xc+textwidth(title)+5,yc+textheight(title)+5,title);
        settextstyle(SMALL_FONT,□ERT_DIR,5);
//      settextjustify(LEFT_TEXT,RIGHT_TEXT);
        sprintf(title,"Generalized □Y --->");
        outtextxy(xc-textheight(title)*3,textwidth(title)/2,title);

//      Plots trajectory
        fp = fopen("plot.d","r");
        fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
        do
                {
                yr = (y/ym)*(yc/2)+xc;
                vyr = -(vy/vym)*(yc/2)+yc;
                putpixel(yr,vyr,□ HITE);
//              delay(100);       // Delay by 100ms between two points
        fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
                }
        while(x != 100.0 && y != 100.0);
        fclose(fp);

//      Exit instruction
        setcolor(LIGHTRED);
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"Hit any key to return to menu");
        outtextxy(xc-textwidth(title)/2,yf-textheight(title)-5,title);
        getch();
        closegraph();
        }
//      -------------       SUBROUTINE 'PROJZ()'        -----------------
//      Subprogramme to plot in Z-□Z plane

void projz()
```

```
        {
        float zr,vzr;
        char title[30];
        FILE *fp;

//□ rites title
        setcolor(YELLO□ );
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"    PROJECTION");
        outtextxy(2,1.5*textheight(title),title);
        sprintf(title,"    IN Z-□Z PLANE:");
        outtextxy(2,3.5*textheight(title),title);

//      Labels X and Y axes
        setcolor(LIGHTCYAN);
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        sprintf(title,"Generalized Z --->");
        outtextxy(xc+textwidth(title)+5,yc+textheight(title)+5,title);
        settextstyle(SMALL_FONT,□ERT_DIR,5);
        sprintf(title,"Generalized □Z --->");
        outtextxy(xc-textheight(title)*3,textwidth(title)/2,title);

//      Plots trajectory
        fp = fopen("plot.d","r");
        fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
        do
                {
                zr = (z/zm)*(yc/2)+xc;
                vzr = -(vz/vzm)*(yc/2)+yc;
                putpixel(zr,vzr,□ HITE);
//              delay(100);        // Delay by 100ms between two points
        fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
                }
        while(x != 100.0 && y != 100.0);
        fclose(fp);

//      Exit instruction
        setcolor(LIGHTRED);
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"Hit any key to return to menu");
        outtextxy(xc-textwidth(title)/2,yf-textheight(title)-5,title);
        getch();
        closegraph();
        }

//      -------------        SUBROUTINE 'PROJR()'        -----------------
//      Subprogramme to plot in real (110) plane

void projr()
        {
        int i,j,k;
        float a = 5.43089;          //Si lattice constant in Ang.
        float a8,dp;
        float xr,yr,xym,sc;    //xym - maximum value of x and y combined
                                //sc - A scale factor such that trajectory is
```

```
                         // within 90% of 'yc'
        char title[30],titcl[30];

        int gd = DETECT,gm;
        FILE *fp;

//      Opens graphic screen and defines the values of parameters a8,dp,...etc.
        initgraph(&gd, &gm,"c:\\tc\\bgi");      //Initialising graphic screen
        xf = getmaxx();
        yf = getmaxy();
        xc = xf/2;
        yc = yf/2;
//      xm = 5.0;       //Lines to test (Do not discard this and next lines)
//      ym = 5.0;       //Changing xm and ym will allow you
                       //to include as many rows more or less
                       // in the (110)plane.
        xym = max(xm,ym);
        sc = yc*0.9;    //Curve to come up to 40% of screen border. Indirectly
                         //decrease of the coefficient of 'yc' increases the
                         // number of atomic rows on the screen.
        a8 = a/8.0;
        dp = a/(2.0*sqrt(2.0));  //Interplanar separation in (110)
        a = (a/xym)*sc;          //  |
        a8 = (a8/xym)*sc;        //  | Scaling to screen coordinates
        dp = (dp/xym)*sc;        //  

//      Generate the atomic rows in the plane
        setcolor(MAGENTA);
        setfillstyle(SOLID_FILL,MAGENTA);   //fill style for atomic rows
        for(j=-1; j<2; j+=2)      //-1 Left of screen-centre
                {
                for(i=-1;i<2;i+=2)  //-1 Above screen-centre
                        {
                        k = -i;     //k fixes centre of the basis
                                    //At hte beginning j=-1,i=-1 and k=1
                                    //so that yr = yc. ie. The origin (screen
                                    //centre is the mid-point of two atoms
                                    // in a basis.
                        yr = yc + (i-1)*dp/2.0;
                        do
                                {
                                xr = xc + ((k+1)*a/4.0);
                                do
                                {
                                pieslice(xr+a8,yr,0,360,2);  //Draws atomic rows
                                pieslice(xr-a8,yr,0,360,2);  //Draws atomic rows
                                xr = xr + j*a;
                                }
                                while(xr >= 0 && xr <= xf);
                                k = -k;
                                yr = yr + i*dp;
                                }
                        while(yr >= 0 && yr <= yf);
                        }
                }
//      Draws unit cell
        setcolor(GREEN);
        line(xc-a*0.5,yc,xc,yc-dp);
        line(xc,yc-dp,xc+a*0.5,yc);
        line(xc+a*0.5,yc,xc,yc+dp);
        line(xc,yc+dp,xc-a*0.5,yc);

//□ rites title
        setcolor(YELLO□ );
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"    PROJECTION");
        outtextxy(2,1.5*textheight(title),title);
        sprintf(title,"    IN (110) PLANE:");
        outtextxy(2,3.5*textheight(title),title);

//      Plots trajectory
        setcolor(LIGHTCYAN);
        settextstyle(SMALL_FONT,HORIZ_DIR,4);
        i = 0;
        fp = fopen("plot.d","r");
        fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
        do
                {
                xr = (x/xym)*sc+xc;
                yr = -(y/xym)*sc+yc;
                putpixel(xr,yr,□ HITE);
                depth();
//              delay(50);      // Delay by 100ms between two points
                fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
                }
        while(x != 100.0 && y != 100.0);
        fclose(fp);

//      Exit instruction
        setcolor(LIGHTRED);
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"Hit any key to return to menu");
        outtextxy(xc-textwidth(title)/2,yf-textheight(title)-5,title);
        getch();
        closegraph();
        }

void depth()
        {
        char title[30];

        setfillstyle(SOLID_FILL,BLACK);
        sprintf(title,"  Depth: %g Ang.",z);
        settextstyle(0,HORIZ_DIR,1);
        bar(0,yf-1.1*textheight(title),1.1*textwidth(title),yf);
        setcolor(□ HITE);
        outtextxy(0,yf-textheight(title),title);
        }

void projun()
```

```
          {
          extern float x,y,z,vx,vy,vz;
          extern float x1,y1;
          int i,j,k;
          float a = 5.43089;          //Si lattice constant in Ang.
          float a8,dp;
          float xr,yr,xym,sc;   //xym - maximum value of x and y combined
                                 //sc - A scale factor such that trajectory is
                                 // within 90% of 'yc
          char title[30],titcl[30];

          int gd = DETECT,gm;
          FILE *fp;

//        Opens graphic screen and defines the values of parameters a8,dp,...etc.
          initgraph(&gd, &gm,"c:\\tc\\bgi");       //Initialising graphic screen
          xf = getmaxx();
          yf = getmaxy();
          xc = xf/2;
          yc = yf/2;
//        xm = 5.0;       //Lines to test (Do not discard this and next lines)
//        ym = 5.0;       //Changing xm and ym will allow you
                         //to include as many rows more or less
                         // in the (110)plane.
//        xym = max(xm,ym);
          xym=2.0;
          sc = yc*0.9;    //Curve to come up to 40% of screen border. Indirectly
                           //decrease of the coefficient of 'yc' increases the
                           // number of atomic rows on the screen.
          a8 = a/8.0;
          dp = a/(2.0*sqrt(2.0));  //Interplanar separation in (110)
          a = (a/xym)*sc;           //  |
          a8 = (a8/xym)*sc;         //  | Scaling to screen coordinates
          dp = (dp/xym)*sc;         //  □

//       Generate the atomic rows in the plane
          setcolor(MAGENTA);
          setfillstyle(SOLID_FILL,MAGENTA);   //fill style for atomic rows
          for(j=-1; j<2; j+=2)       //-1 Left of screen-centre
                    {
                    for(i=-1;i<2;i+=2)  //-1 Above screen-centre
                              {
                              k = -i;     //k fixes centre of the basis
                                          //At hte beginning j=-1,i=-1 and k=1
                                          //so that yr = yc. ie. The origin (screen
                                          //centre is the mid-point of two atoms
                                          // in a basis.
                              yr = yc + (i-1)*dp/2.0;
                              do
                                        {
                                        xr = xc + ((k+1)*a/4.0);
                                        do
                                        {
                                        pieslice(xr+a8,yr,0,360,2);  //Draws atomic rows
                                        pieslice(xr-a8,yr,0,360,2);  //Draws atomic rows
                                        xr = xr + j*a;
                                    }
                                    while(xr >= 0 && xr <= xf);
                                    k = -k;
                                    yr = yr + i*dp;
                                    }
                            while(yr >= 0 && yr <= yf);
                            }
                    }
//      Draws unit cell
        setcolor(GREEN);
        line(xc-a*0.5,yc,xc,yc-dp);
        line(xc,yc-dp,xc+a*0.5,yc);
        line(xc+a*0.5,yc,xc,yc+dp);
        line(xc,yc+dp,xc-a*0.5,yc);

//□ rites title
        setcolor(YELLO□ );
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"    PROJECTION");
        outtextxy(2,1.5*textheight(title),title);
        sprintf(title,"    IN (110) PLANE:");
        outtextxy(2,3.5*textheight(title),title);

//      Plots trajectory
        setcolor(LIGHTCYAN);
        settextstyle(SMALL_FONT,HORIZ_DIR,4);
        i = 0;
        fp = fopen("plot.d","r");
        fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
        conv();
        do
                {
                xr = (x1/xym)*sc+xc;
                yr = -(y1/xym)*sc+yc;
                putpixel(xr,yr,□ HITE);
                depth();
//              delay(50);      // Delay by 100ms between two points
                fscanf(fp,"%g %g %g %g %g %g",&z,&x,&y,&vx,&vy,&vz);
                conv();
                }
        while(x != 100.0 && y != 100.0);
        fclose(fp);

//      Exit instruction
        setcolor(LIGHTRED);
        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        sprintf(title,"Hit any key to return to menu");
        outtextxy(xc-textwidth(title)/2,yf-textheight(title)-5,title);
        getch();
        closegraph();
        }

void conv()
        {
```

```
        extern float x,y,z,vx,vy,vz;
        extern float x1,y1;
        float a = 5.43089;          //Si lattice constant in Ang.
        float calp,salp,salp1,ax,ay;
        calp=a*0.5;
        salp=a/(2*sqrt(2.0));
        salp1 = sqrt(2.0)/a;
        ax = 0.5 + x/a;
        ay = y*salp1;
        r1 = ax-ay;
        r2 = ax+ay;
        if(r1 >= 0)
                r1 = r1 - (int)(r1);
        else
                r1 = r1+(int)(1.0+abs(r1));
        if(r2 >= 0)
                r2 = r2 - (int)(r2);
        else
                r2 = r2+(int)(1.0+abs(r2));

//      printf("%f,%f\n",r1,r2);
//      getch();
        x1=(r1+r2-1.)*calp;
        y1=(r2-r1)*salp;
        }
```

gentrn.c

```
/*      Programme to convert x,y in <110> frame to generalised coordinates
        r1,r2 in <110> frame                */
#include <stdio.h>
//#include <conio.h>
#include <math.h>
extern float xm,ym,zm,vxm,vym,vzm;
extern float r1,r2;
void gentrn(float x, float y)
        {
//     --------     CODE 1          -----------
        r1 = x/xm;
        r2 = y/ym;

/*     --------     CODE 2          -----------
//      Alternate code to generate generalised coordinates of Ellison et al.
        float ax,ay,salp1;
        float a = 5.43089;

        salp1 = sqrt(2.0)/a;
        ax = 0.5 + x/a;
        ay = y*salp1;
        r1 = ax-ay;
        r2 = ax+ay;
        if(r1 >= 0)
                r1 = r1 - (int)(r1);
        else
                r1 = r1+(int)(1.0+abs(r1));
        if(r2 >= 0)
                r2 = r2 - (int)(r2);
        else
                r2 = r2+(int)(1.0+abs(r2));
*/
        }
```

slate.c

```
//      ---------- SUBROUTINE FOR MAIN IN 'TRBASIN.C'   ------------
//      Subprogramme to open a graphics screen, draws the X-Y axes,
//      and ticks and labels them for further plotting by projx(),... etc.
//#include <conio.h>
#include <graphics.h>
#include <stdio.h>
#include <math.h>
#define PI 3.14159265
extern  int xf,yf,xc,yc;

void slate()
{
        int gd = DETECT,gm,i,im,shifty;
        char lbltiks[3],title[30];

        initgraph(&gd, &gm,"c:\\tc\\bgi");      //Initialising graphic screen
        xf = getmaxx();
        yf = getmaxy();
        xc = xf/2;
        yc = yf/2;

//Drawing coordinate axes
        setcolor(LIGHTGREEN);
        line(0,yc,xf,yc);
        line(xc,0,xc,yf);

//Puts X ticks and Y ticks
        setcolor(CYAN);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,0);
        settextjustify(1,2);
        im = yc/2;
        shifty = 0.5*textheight("|");
        for(i=-2; i < 3; i++)
                {
                if (i != 0)
                        outtextxy(xc+(i*im),yc-shifty,"|"); // X - ticks
                }
        settextstyle(DEFAULT_FONT,□ERT_DIR,0);
        settextjustify(1,1);
        for(i=-2; i < 2; i++)
                {
                if (i != 0)
                        outtextxy(xc+0.5*shifty,yc+(i*im),"|");  //Y - ticks
                }

//Labelling ticks
        settextstyle(DEFAULT_FONT,HORIZ_DIR,0);
        settextjustify(1,2);
        setcolor(MAGENTA);
        for(i=1; i < 2; i++)
                {
                sprintf(lbltiks,"%d",i);
                outtextxy(xc+(i*im),yc+2*shifty,lbltiks);
                }
        for(i=1; i < 2; i++)
                {
                sprintf(lbltiks,"-%d",i);
                outtextxy(xc-(i*im),yc+2*shifty,lbltiks);
                }
        settextjustify(0,1);
        for(i=1; i < 2; i++)
                {
                sprintf(lbltiks,"%d",i);
                outtextxy(xc-1.5*textwidth(lbltiks),yc-(i*im),lbltiks);
                }
        for(i=1; i < 2; i++)
                {
                sprintf(lbltiks,"-%d",i);
                outtextxy(xc-1.5*textwidth(lbltiks),yc+(i*im),lbltiks);
                }
}
```