Université d'Ottawa · University of Ottawa

University of Ottawa
Master's Program in Systems Science

Thesis

Optimization of Device Control for the Flicker Perimeter

by

Jie Wu

Thesis Director
Evanne Casson, Ph. D.

May 15, 2002

Acknowledgments

First and foremost, I thank
Professor Evanne Casson, who has supervised this work.

In the process of researching this project,
I met and talked to many extraordinary people in the field of medical clinic, systems science, modeling and simulation and computer programming.
They provided me with unique knowledge and experience in a complex domain. I wish to thank them all for their advice, courtesy and time they allotted so that this work could come to fruition.

Abstract

To optimize the function of a prototype flicker perimeter, it is important to reduce the amount of time required to change test target location in the visual field while maintaining the random presentation order necessary to ensure accurate results. The purpose of this project is to develop and test a software algorithm that achieves these two goals simultaneously. I have developed an algorithm for the Flicker perimeter that maximizes the efficiency of travel in the motors governing test target projection. This will reduce the time between test trials as well as the entire sequence, thus limiting the fatigue factor in testing. The algorithm incorporates a component that maintains a pseudo-random order of presentation. I have developed and constructed a simulation of the functioning perimeter by designing and reengineering the system program and incorporating both the original presentation algorithm (Random) and the optimizing algorithm (Optimized). Using a comparison technique, I demonstrate that the simulation corresponds exactly to the algorithms for presentation order as well as reflecting the behavioral relationship between the patient and the actual flicker perimeter during a test session. After determining the appropriate number of session iterations for the simulation, the results of the simulation for Random and Optimized algorithms were compared and the impact of restrictions on maximum travel time on the effectiveness of the Optimized algorithm was investigated. The results demonstrate clearly that the Optimized algorithm enhances the efficiency of the program by reducing overall test time and the number of test sequences requiring long travel times between test locations. This efficiency increases with the restrictions on the maximum length of travel time; however such restrictions also increase the number of non-random test sequences. Within a certain range of restrictions, the results indicate that it is possible to reduce test time by a factor of at least two while not significantly increasing the frequency of non-random test sequences. The implications of these results are discussed in terms of both the algorithm and the simulation methods, which both have potential in biomedical instrumentation development.

Table of Contents

Chapter 1 - Background

I propose to modify the testing algorithms of the Flicker Perimeter, a prototype device designed to enhance the early detection of glaucoma.

1.1 Glaucoma

Glaucoma is a disease of the eye marked by increased pressure within the eyeball that can result in damage to the optic nerve and gradual loss of vision. It is a leading cause of irreversible blindness in older Canadians. The disease is the result of gradual loss of retinal nerve fibers due to increased intra-ocular pressure and optic nerve head cupping. This results in gradual visual loss, which progresses from the peripheral visual field towards central vision. Since this loss is painless, it can only be detected through regular medical examination. Once blindness from glaucoma has occurred, no treatment will restore the lost vision. However, in nearly all cases, blindness from glaucoma is preventable if detected and treated at an early stage. This prevention requires early detection and proper treatment. Early detection is very important and is based on the observing evidences of optic nerve head cupping and early loss in the peripheral visual field [Shields, B., 1992].

1.2 Perimetry

Evidence of visual field loss is obtained through perimetric assessment [Casson, E. J., 1992, 1996]. Such assessments are carried out using a perimeter, consisting of an evenly lit hemisphere or flat surface upon which targets of various intensities are presented at different locations in the peripheral visual field to determine localized visual sensitivity. In most cases, this assessment is carried out using an automated perimeter that uses software to control the testing procedure and the recording of patient responses. Perimetry [Haley, M. J., 1997] is a method of assessing visual field loss by testing sensitivity at various location of visual field of the each eye of the patient. This is achieved by looking at fixation target while the test light is presented in visual field. The

location of the test target is determined in a random way, so the patient cannot predict the next test and move their eye to look directly at it. This method of testing peripheral visual field locations can be achieved by either having the patient look at a fixed fixation target and moving the test target to different points in visual field, or by having the patient look at a moving fixation target and by causing the eye to move, changing the position of a fixed test target to be presented at different locations in visual field. The second method is used in the Flicker device, where the test target is too complex to be moved.

1.3 Flicker Perimeter

Based on previous research [Casson, E. J., 1992, 1996] that indicates the perception of flickering targets is reduced early on in the process of glaucomatous damage, we have developed a new method of perimetry and designed and built a Flicker Perimeter. It tests the ability to detect whether or not a test target is flickering or steady, as the modulation amplitude of a sinusoidal flickering light of a constant average luminance is varied to determine patient sensitivity.

However, the current algorithm controlling the presentation of the test targets (via movements of the fixation target) and the determination of sensitivity in Flicker Perimeter is based on standard perimetric algorithms where test time is already quite long. In addition, due to the design of the Flicker Perimeter, the travel time between the test points is significantly slower than other perimeters. Thus, test time will increase even further. As a result, we are unable to obtain a high-resolution representation of the visual field, an essential component in early detection without causing severe fatigue in the patient.

1.4 The goal

The goal of my research is to develop an algorithm that reduces travel time between perimetric locations, thus reducing time for the entire test sequence and the potential fatigue for patient while maintaining the random-like presentation order necessary to ensure accurate results. The project background and goal is illustrated in figure 1. In

following chapters, I describe the system, as well as the design, implementation and assessment of a new algorithm using a software simulation of the Flicker Perimeter.

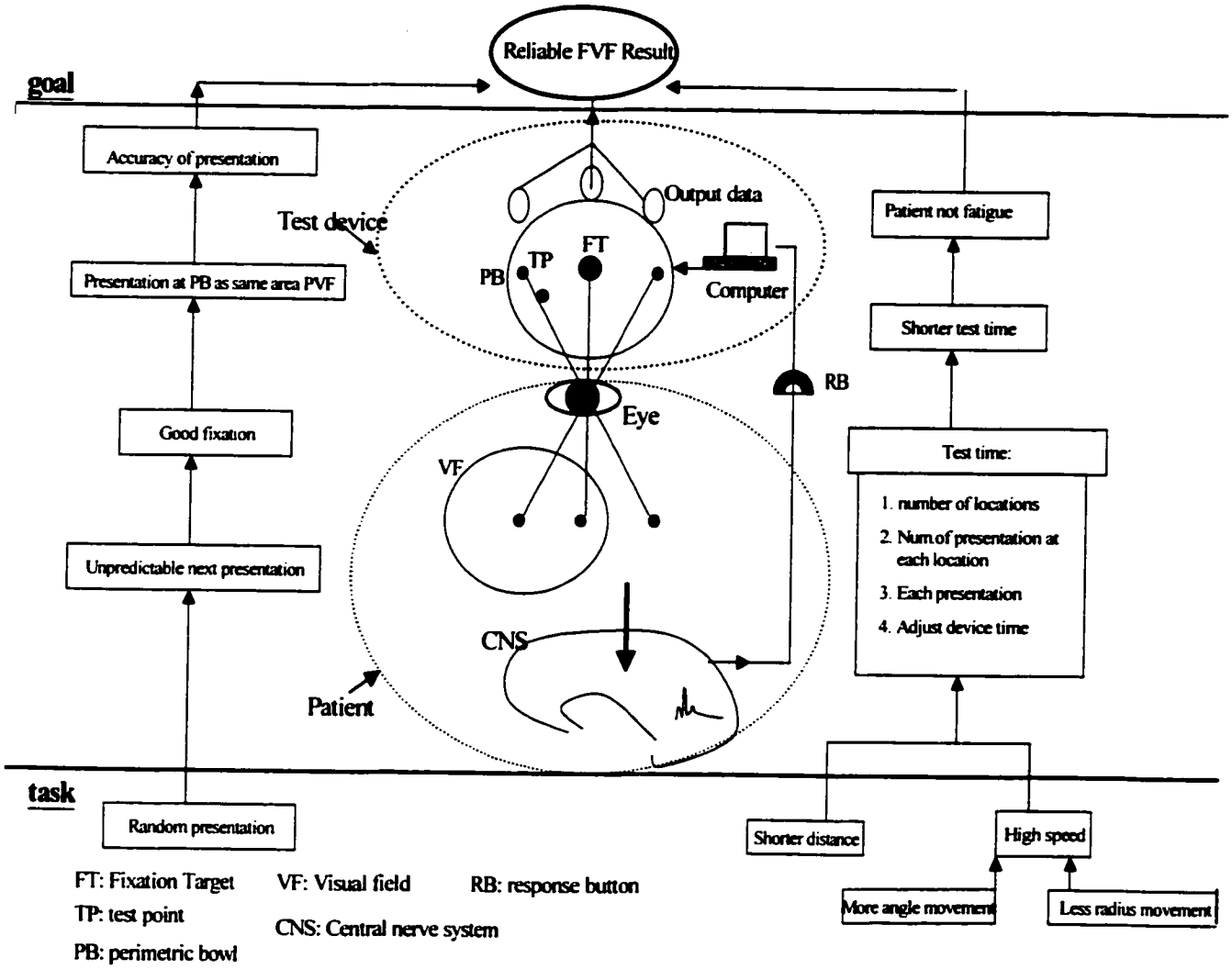


Fig. 1: The scope of the main research goal and task in my project

Chapter 2 - Problem Definition and Methodology

The system I am investigating is a prototype device, which has been designed and developed to meet the requirements of standard perimetric practice [E. Casson, 1995, 1997]. The documentation and references related to this prototype are limited. In order to do further development on the device, it is necessary to first understand the design and measurement method of the system in detail.

2.1 A Description of the Flicker Perimeter and Measurement Method

2.1.1 System Components

The Flicker Perimeter (fig. 2) consists of a large perimetric bowl, 60 cm in diameter, with a chin rest attached to the lower rim so that the patient sits facing into the center of the bowl, with his/her chin and forehead resting comfortably in the chin rest. A response button is provided for the patient to signal when he/she has seen a target. The test target is situated at the farthest point of the sphere facing the patient, i.e., the center of the bowl. This target, which can be altered in diameter between 5 degrees and 0.5 degrees of visual angle, is modulated with various rates of Flicker and various amplitudes. The fixation light is attached to a mechanical arm to allow it to take a variable position within the perimetric bowl. It is intended that the patient look steadily at this light as it is moved from location to location in the perimeter bowl, placing the central target in different locations within the peripheral visual field. The patient then reports when he/she sees the test target flickering using peripheral vision, allowing the determination of visual sensitivity at a number of locations within the visual field. Fixation arm movement allows the presentation of flickering targets within 60 ($\pm$30) degrees of visual field surrounding central vision (fixation).

The patient reports seeing the flickering of the test target via a response button that can be pressed to send a signal to the measurement device that the patient perceives the flicker light at the test target location. The modulation amplitude of the flicker for that location can then be altered for the next presentation depending on the response

(Yes = reduced amplitude, No = increased amplitude). This process is repeated for the next location tested until all locations have been tested in random order. The process is repeated until the modulation amplitude of flicker represented for each location approximates the minimal amount required for the patient to respond (threshold).

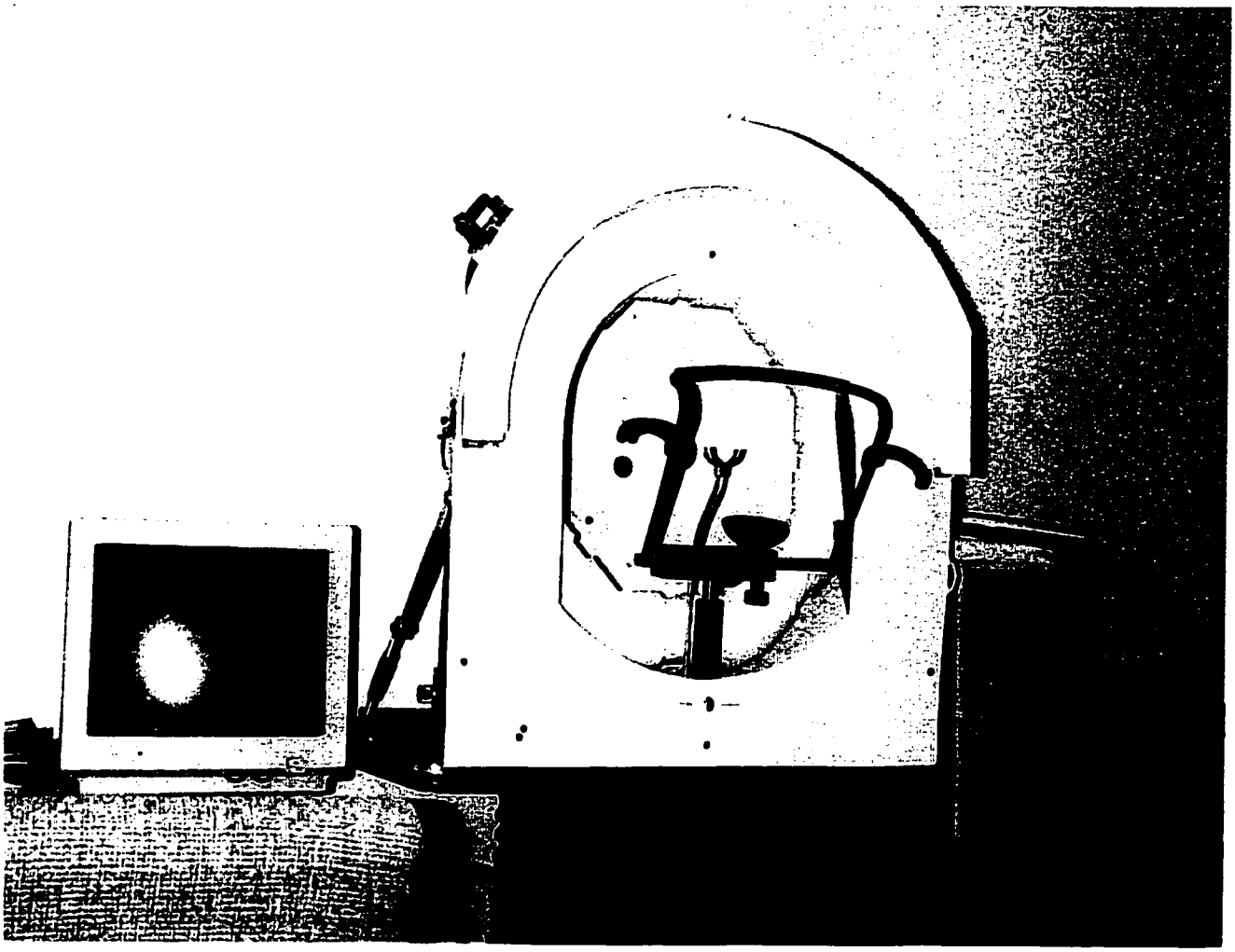


Fig. 2: The Flicker Perimeter

From the software designer's point of view, the challenges are to develop efficient methods for

- moving the fixation light from location to location;
- altering the amplitude of test target light at each location in view of the patient response;
- determining a random sequence of fixation light movements to ensure the random order of target presentation in different locations in the visual field.

Of these, the most difficult part to arrange is the movement of the fixation light. In the device under investigation, the fixation light is on the end of the fixation light arm, which is itself mounted on a semi-circular support that can be moved through 180 degrees about an axis that runs in the patient entrance pupil plane but is perpendicular to the line of vision (Fig. 3). There are two motors controlling the movement of fixation light arm: the polar arm, which moves the fixation target around the circumference of the perimeter bowl, and the radial arm, which moves the fixation target along any arc within the bowl. Figure 4 illustrates the movement of the fixation light arm from the last position, P, to next position, Q, by first performing a polar movement then a radial movement.

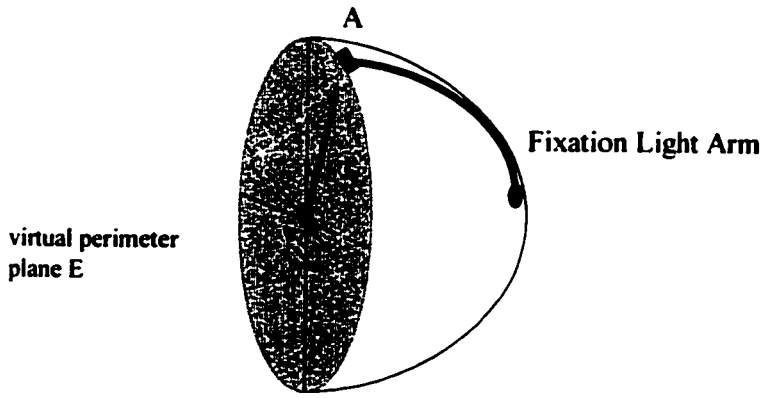


Fig.3: The projection of Fixation Light Arm
on the patient entrance pupil plane E

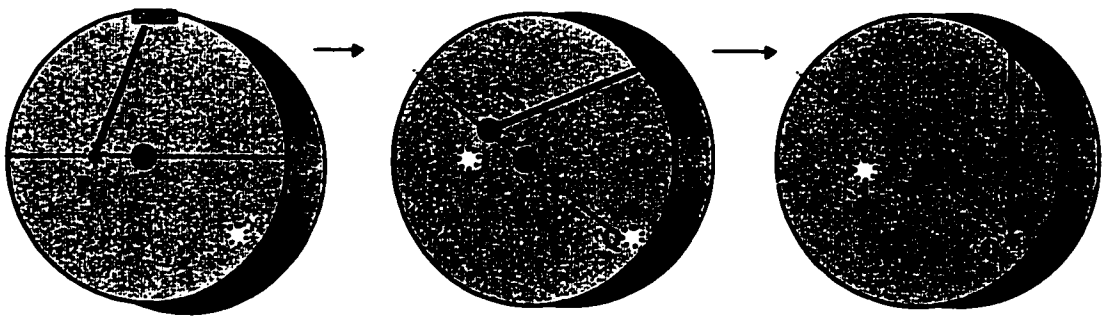


Fig. 4: The movement of fixation target: the polar and radial movement of the fixation light arm
from point P to point Q, first perform the polar movement then the radial movement

From the point of view of the technician, the operation of the equipment is viewed via a computer monitor, since all the commands to operate the equipment and the gathering of data are conducted under program control. Indeed it is a goal of this paper to define and discuss the appropriate properties of that program. The technician can output results and diagnostic information to a printer attached to the computer.

During testing, sensitivity values are obtained for each location, based on the pattern of responses that the patient makes to targets at that location. For example if a flickering test target presented at location X is readily perceived flickering by the patient, he/she presses the response button. The relationship between the flicker amplitude (A_1) and the "seen" response is noted and the next time location X is tested, the flicker amplitude is reduced (A_2). If, on this test presentation the flicker is not perceived, the patient does not respond. This is again noted and the flicker amplitude increased slightly (A_3) for the next presentation at location X. If the patient responds positively, the average of the last

"seen" and not seen" ($(A_2 + A_3)/2$) is taken to represent the reciprocal of the patient's sensitivity for that location and this value is stored in a data array. On average this process requires three iterations of target presentation at each location in the visual field. Once sensitivities have been calculated for each location in the visual field (typically 76 locations for a visual field representing $\pm$ 30 degrees around fixation), the data array is output in a form that is easy to interpret from a diagnostic perspective, using a normative database for comparison.

2.1.2 Test Strategy

Based on previous investigations [3, 4] and the requirements of the testing procedure I used the following general test parameters and strategies as a model for the perimetric testing procedure used in the Flicker Perimeter.

• The test target is fixed at the center of the perimetric bowl. It is capable of three different rates of flicker, namely 4Hz, 8Hz, and either 16 or 32Hz

• The fixation light is movable to any of a set of predefined locations on the sphere of bowl. I based the model on the test pattern used in a standard diagnostic test, the 30-2 test pattern from the Humphrey Field Analyzer [Haley, M. J., 1997] (Fig. 5). Using this pattern, sensitivities must be obtained at each of 76 points in total, with a distance between any two near points being 6°.

• To ensure that the patient does not try to anticipate the movement of the fixation target and thus negatively impact the accuracy of the test results, the movement sequence of the fixation light should be such that the patient is unable to predict it. Such requirements are satisfied if the sequence of test locations is determined in a truly random pattern. For example, by physically drawing a (numbered) ball from some kind container. But this is not very practical for computer simulation, and the software equivalent of this process sometimes requires long intervals between sequential tests. To meet the requirement of unpredictability, it may be sufficient that the patient be unable

to anticipate the movement or presentation of the test target in a reasonable period of time. This does not require a truly random method to determine test locations. In a later section I will introduce and use "pseudo-random" number generators to generate unpredictable sequences of test points for patients.

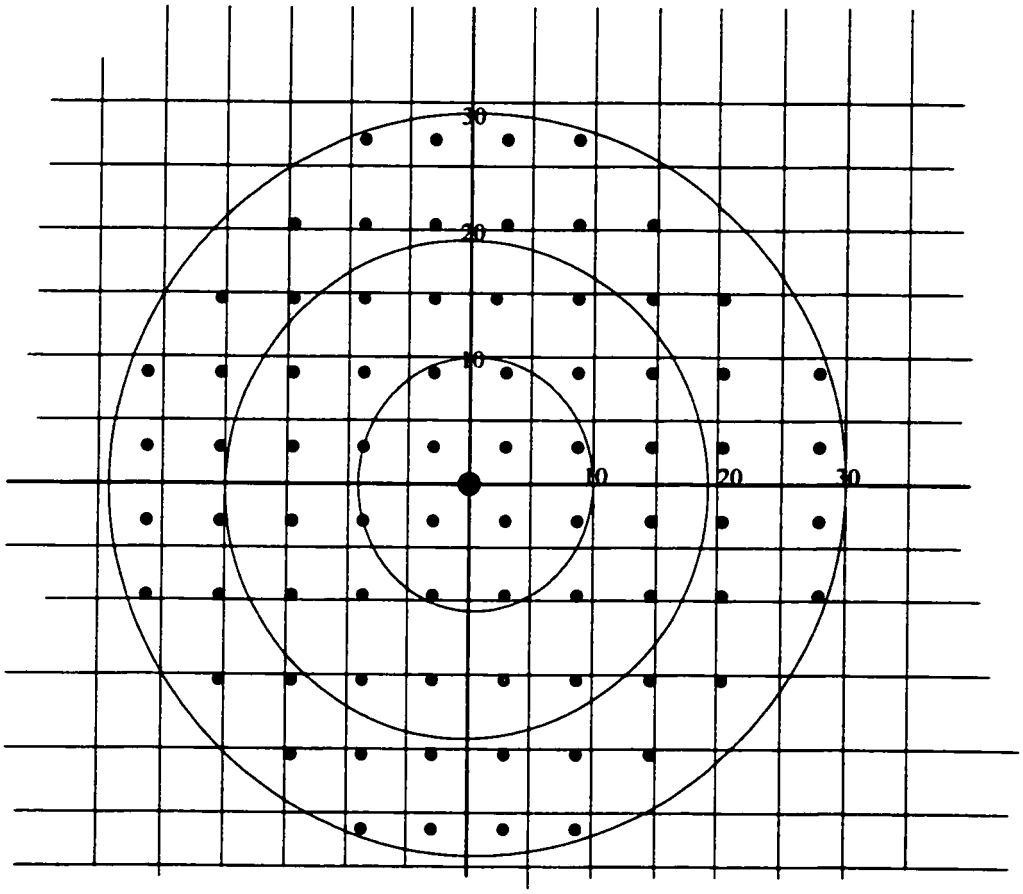


Fig 5: 76 test locations (Humphrey 30-2 pattern)

2.1.3 Description of Movable Objects and Type of Movement

- Movable Objects:

• Fixation Light Arm is described mathematically as an arc with length of R* pi /2 (where R is the radius of the bowl), one end attached at any point (between 0° ~ 180°) on the circumference of the largest section of the bowl – (represented in two dimensions by the virtual perimeter plane (see Fig 3); one end is free;

• Fixation Light is located at the free end of the Fixation Light Arm (Fig 3)

• The Fixation Light is positioned at any location in the visual field by combining two types of movement, polar movement and radial movement. These are described below.

- Type of Movement :

• Polar movement: Fixation Light Arm moves around the circumference of the virtual perimeter plane. The positions of both ends change (Fig. 6) (Fig. 7)

• Radial movement: Fixation Light Arm moving around specific radius, keeping same polar position, only the position of Fixation Light changes (Fig. 8).

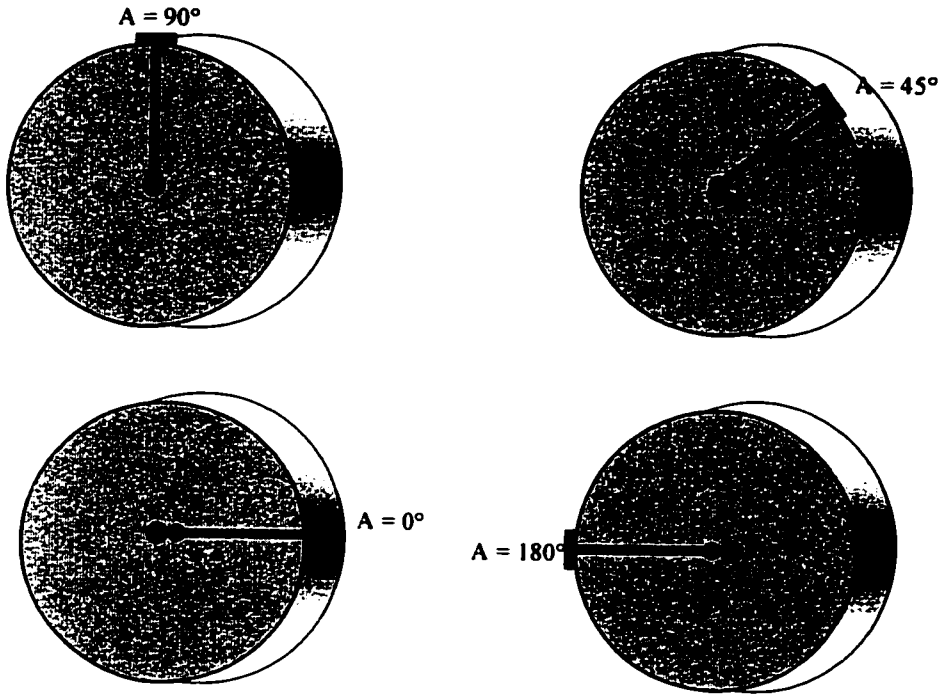


Fig. 6: Polar movement

A simple polar movement with an initial fixation light position, as the arm moves, only the polar value changes, keeps radial value same

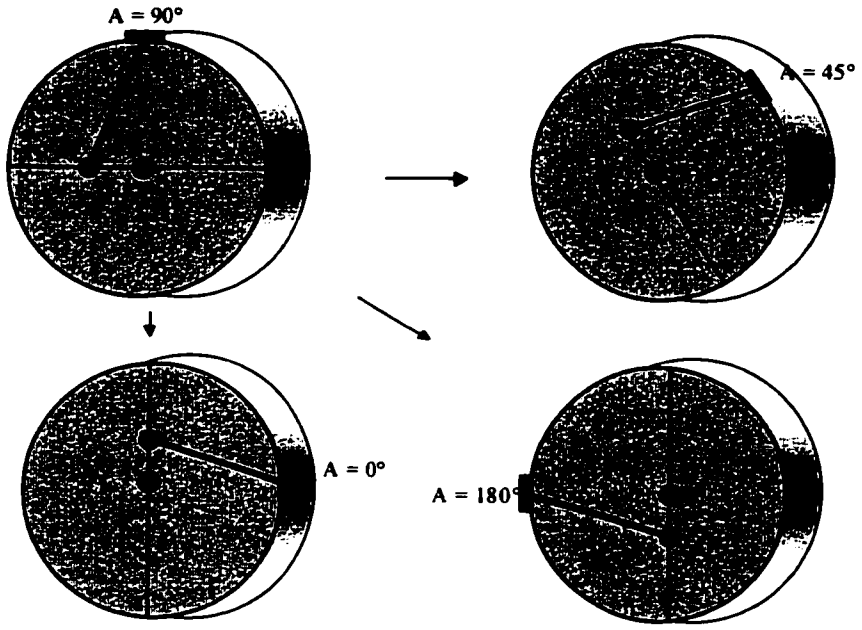


Fig 7: Polar movement(2)

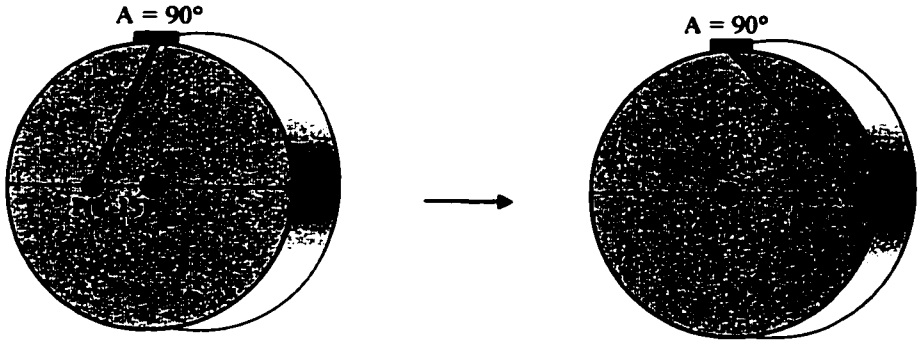


Fig. 6 Radial movement of the fixation light arm
A: the position of the fixation light arm
p, q: the position of the of fixation light

2.1.4 Point to point movement details

How can the Fixation Light be moved from last position *p(x1,y1)* to next position *q(x2,y2)*?

1. *What is P(x,y)?*

First I define the parameters that determine P (x, y) (Fig. 9):

P0 (0, 0): initial position of the Fixation Light

plane X0: the vertical section which passes the center of the bowl

plane Y0: the horizontal section which passes the center of the bowl

Each and every point (Fig. 10) on the sphere is characterized by two parameters x and y, which can be obtained as follows. Given a point P (x1,y1) on the sphere, I can draw a horizontal plane and a vertical plane that both pass through this given point, denoted by Xp and Yp. The intersection of Y0, Xp, and the sphere is a point Px1, and the intersection of X0, Yp, and the sphere is a point Py1. Let x1 be the angle of the arc P0Px1 in the plane Y0 and let y1 be the angle of the arc P0Py1 in the plane X0. These two coordinates determine a unique point on the sphere: P (x1, y1).

Now I can extend above definition:

The initial position of the Fixation Light P0 (0, 0) is a point at sphere with both angles equal zero,

x=0, y=0; X0 is plane on which every point with the angle of the arc P0Px equals zero, ie, x = 0;

Y0 is plane on which every point with the angle of the arc P0Py equals zero, i.e., y = 0;

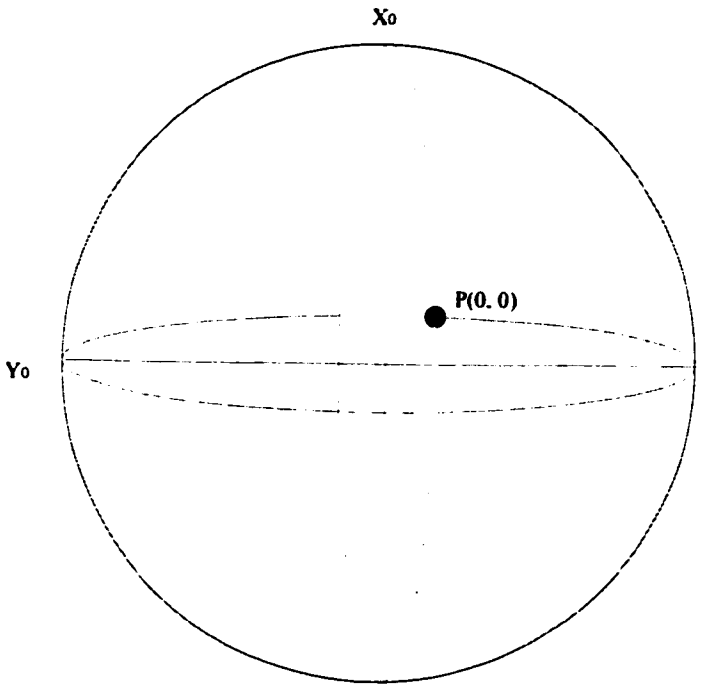


Fig 9: shows initial fixation light p (0, 0)
plane XY1 plane X0, plane Y0

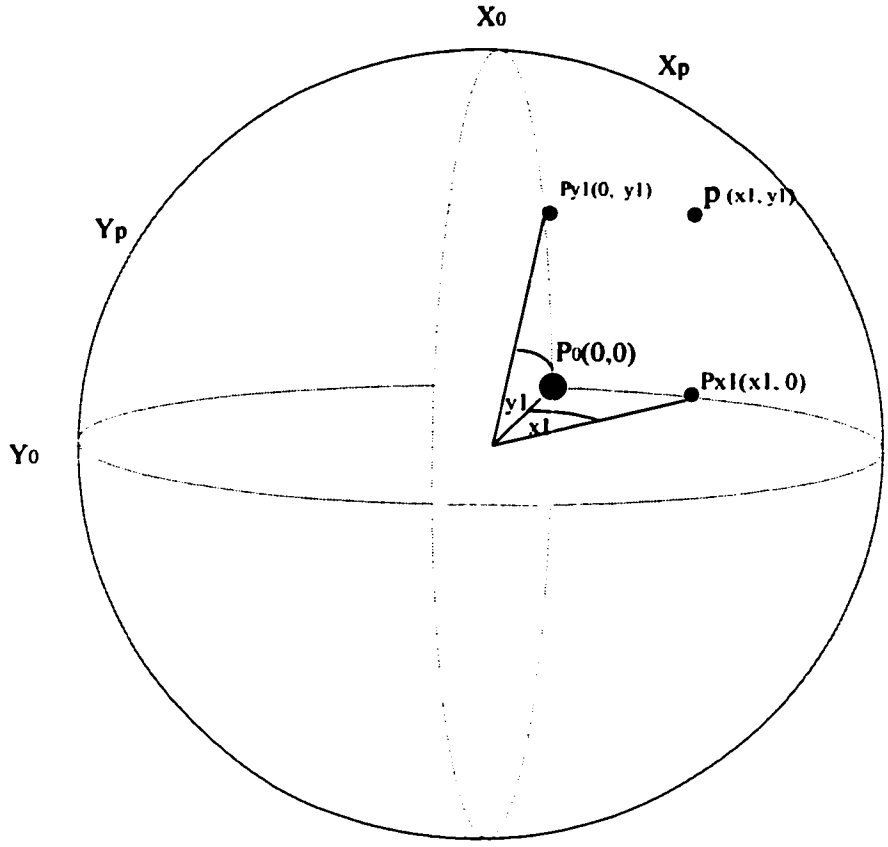


Fig 10: show fixation light at position p (x1, y1)
● is the test target XYl

2. ***Where is the other end of Fixation Light Arm when Fixation Light is at P (x, y)?***

The other end of the Fixation Light Arm is determined by its polar value: **A**. The initial position of the Fixation Light Arm is: **A** = 90°. According to its property of the polar movement, **A** can take any value between 0° ~ 180°(Fig. 4).

As **A** is a point at circumference of the largest section of the bowl – the virtual perimeter plane **E**, I project all the things on the sphere to the virtual perimeter plane (Fig.3) in order to determine value **A** for a given P (x, y). Let P′ be the projection point of P at plane E, the x′ and y′ axis value of P′ can be obtained by plane Y0 and plane X0 (Fig. 9):

$$P'(R^* \sin x, R^* \sin y)$$

and the angle of P′O′x′:

$$\theta = R^* \sin y / R^* \sin x$$

As a result of the properties of the radial movement, I only need to set the Fixation Light Arm in a polar value **A** which its projection line meets the line P′O′ at right angle. With this right polar position A, the radial movement of the fixation light arm is a simple, rapid movement to reach the point P (x, y):

$$A = (\arctan (R^* \sin y / R^* \sin x))^*180/\pi + 90°$$

3. ***Movement of the fixation light arm***

Now I can answer the question "How can the Fixation Light be moved from one position *p (x1, y1)* to next position *q (x2, y2)*?" The calculations below are illustrated in Figures 10-12.

p(x1,y1):

Projection: $P'(R^* \sin x1, R^* \sin y1)$,

$Ap = (\arctan (R^* \sin y1 / R^* \sin x1))^*180/\pi + 90°$ (if $Ap < 0$, $Ap + 180$)

q(x1,y1):

Projection: $q'(R^* \sin x2, R^* \sin y2)$,

$Aq = (\arctan (R^* \sin y2 / R^* \sin x2))^*180/\pi + 90°$ (if $Aq < 0$, $Aq + 180$)

I can determine the polar movement: $\in A = Aq - Ap$, the fixation Light Arm moving around the circumference of plane E counterclockwise with $\in A$ degree if $\in A$ is positive, clockwise if $\in A$ is negative.

To determine the radial movement I need work on another plane, Z0, which passes through this given point,

P0 and O0. The angle φ (P'PO0) represents the radian value PP0 which is the radial distance from

P0 to point P: $\varphi = \arcsin \sqrt{\text{sinx* sinx +siny*siny}} \ *180/pi;$

p(x1,y1): $\varphi p = \arcsin \sqrt{\text{sinx1* sinx1 +siny1*siny1}} \ *180/pi;$

q(x1,y1): $\varphi q = \arcsin \sqrt{\text{sinx2* sinx2 +siny2*siny2}} \ *180/pi.$

I can determine the radial movement: $\in \varphi = \varphi p - \varphi q$, the fixation light moves against to P0, if $\in \varphi$ is positive, towards P0 if $\in \varphi$ is negative.

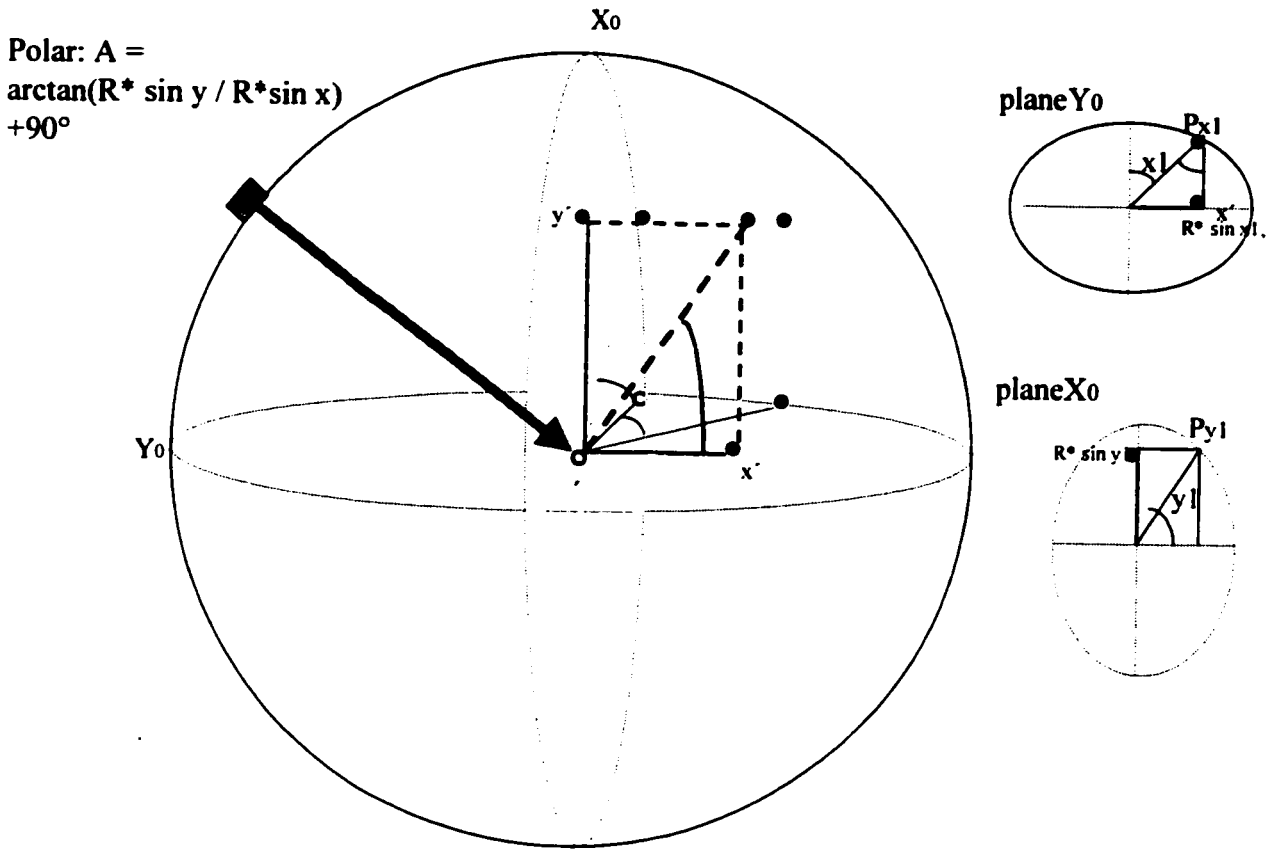
Time consumption of each single point to point movement

Based on a set of measurements obtained from the actual perimeter during operation, the time consumption of the movement of the fixation light arm for polar movement is: 10°/sec, for radial movement is: 90°/sec. The total time consumption of the fixation light moving from last position *p (x1, y1)* to next position *q(x2,y2)* is:

$\in A / (18 \text{ sec} / 180°) + \in \varphi / (2 \text{ sec} / 180°)$

$= (Ap - Aq) / (18 \text{ sec} / 180°) + (\varphi p - \varphi q) / (2 \text{ sec} / 180°)$

The exact calculations for each location are given in Appendix 1: which shows the entire timing matrix for 77X77 locations (including the central point, (0,0)).



XY3

Fig 11: calculation of the movement of the fixation arm

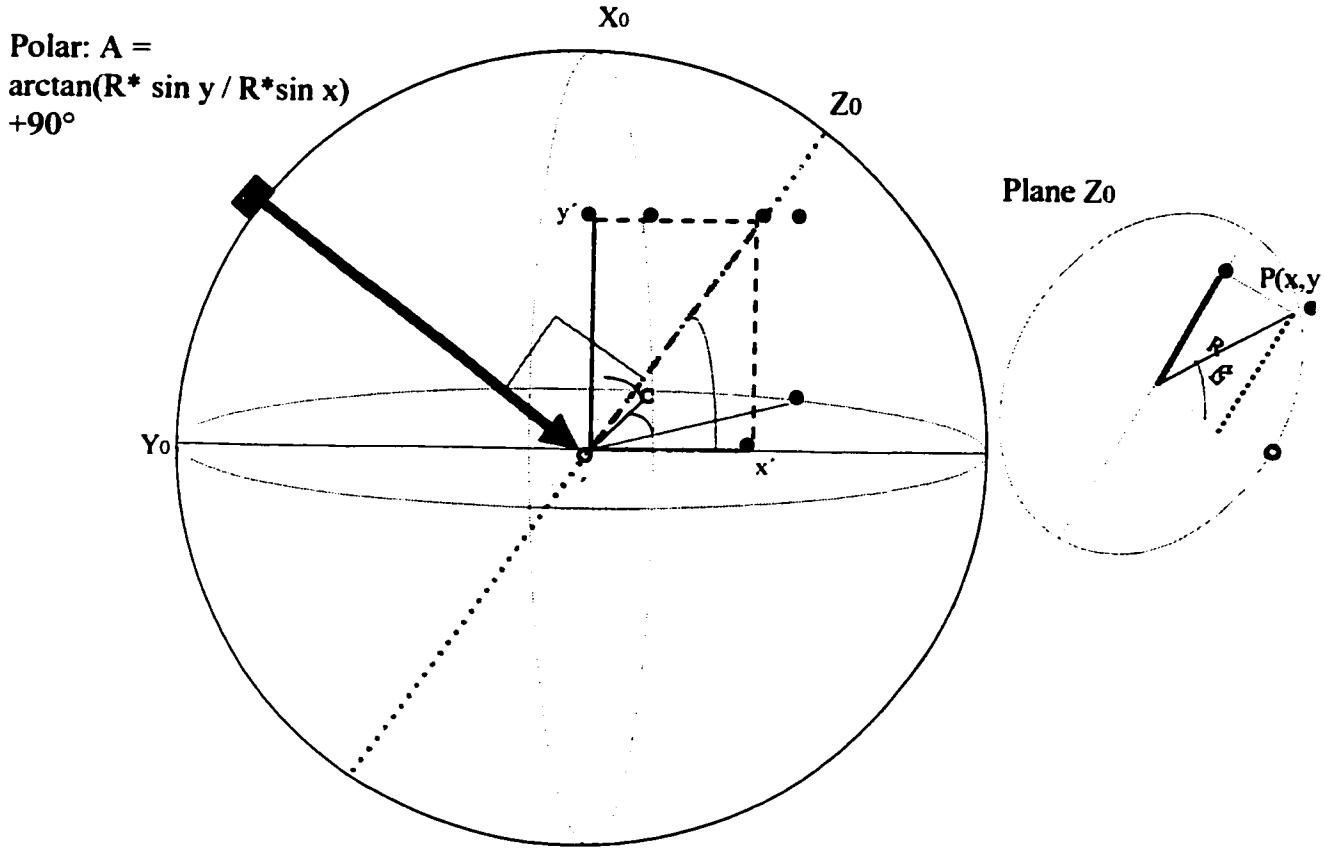


Fig 12: plane Z_0

Presentation of Fixation Light

The ideal situation is that the Fixation Light appears at any one point of the 76 possible positions of Humphry 30-2 pattern [Haley, M. J., 1997] in a pseudo-random way. However the movements described above sometimes take long enough that the overall test and measurement would exceed reasonable time limits. In particular, if the polar value difference of two positions is large, then the movement from these two positions will take longer, since the polar movement takes much more time than the radial movement. According to measurement for the actual perimeter, the time consumption of the movement of the fixation light arm for polar movement is: 10°/sec, for radial movement is: 90°/sec. To understand the differences in times of movement I have labeled the 76 positions as in Figure 5. I then consider all the transition times
T (i, j) = time for movement of fixation arm from position i to position j (Appendix 1)
This does not need to be commutative, i.e., T (i, j) does not need to be equal to T (j, i).

2.2 Problems of the Existing System

The problem: suppose I have a 77X77 matrix T in which T[i][j] represents the travel time from point i to point j. The original system software was designed so that the next test point is chosen at random from the matrix T. This original algorithm provides ideal randomization, but results in long travel time between test points, specially for polar movement. If the difference of the polar values (travel angle) between two successive test points is large, the travel time will also be great, adding up to an unacceptably long total test time. (approx. 10°/sec). How do I find a test sequence such that the selection of the sequence of test locations appears random to patient, but the travel times can be decreased when compared to the times that result from the original testing algorithm.
I described the approach in the next section.

2.3 Theoretical Approach

2.3.1 A Basic algorithm

The goal of the algorithm I am developing is to produce a pseudo-random sequence of test locations that will optimize or at least improve the performance of the testing device by minimizing the time required to display the entire sequence.

The general approach is:
First I use a random number generator to generate a next point, but I need to set a time-limit on the travel time The next point presents only when the travel time from the last point to the next with in this limit. In other words, if the travel time from last to potential next point exceeds to this limit, then the next point simply is skipped, and it keeps generating the another next until next point is generated that will fulfill the predefined travel time-limit requirement. The travel time-limit reflects and is equivalent to the maximum movement of the fixation arm (i.e., travel angle restriction). I hypothesize that the size this restriction directly affects the degree of optimization of the algorithm both in terms of the total test time and the degree of apparent randomization.

2.4 Random Number Generator and Pseudo-random selection of points

2.4.1 Random Number Generator

Before proceeding further, I should first discuss how random numbers (or numbers looking random) can be generated on a computer. Today's practical random number generators are computer programs, which produce a deterministic, periodic sequence of numbers [P. L'Ecuyer, 1988, 1990][G. Marsaglia, 1985]. A deterministic algorithm generating numbers that look, from the outside, as if they were truly random, is called a pseudo-random number generator [P. L'Ecuyer, 1997]. There are many different pseudo-random number generators available in the literature [F. James, 1990] [E. J. Dudewicz and T. G. Ralley,1981] [D. E. Knuth, 1991] [P. L'Ecuyer, 1992]. Here I propose to use a simple and widely used method, called Wichmann Hill [B.A. Wichmann and Hill, 1982]

generator (see Appendix 3 for detailed code). The Wichmann Hill generator is a combination of the following three linear generators:

Generator 1: given an initial value (called seed) x, the algorithm generates a value x′ equal to 171*(x mod 177) – 2*(x/177) if it is nonnegative, otherwise equal to 171*(x mod 177) – 2*(x/177)+ 30269.

Generator 2: given an initial value (called seed) y, the algorithm generates a value y′ equal to 171*(y mod 176) -35*(y/176) if it is nonnegative, otherwise equal to 171*(y mod 176) -35*(y/176) +30307.

Generator 3: given an initial value (called seed) z, the algorithm generates a value z′ equal to 170*(z mod 178) -63*(z/178) if it is nonnegative, otherwise equal to 170*(z mod 178) -63*(z/178) +30323.

The Wichmann Hill generator consists of generating a value w between 0 and 1 using three seeds x, y, and z and the above three generators. w is given by the differential between x′/30269 + y′/30307 + z′/30323 and its integer part, where x′, y′ and z′ are the values generated by Generators 1-3. For production runs, different values (between 1 and 30000) should be used to initialize seeds each time, for example, by some automatic method such as from date and time readings. The value w is considered as a uniformly-distributed random variable on [0, 1]. Since the number generated is on a smaller scale than the digits that I need [1...76], I can scale it by using the modulus operator, ie, (W*100000) / 76 + 1;
Note: the (W*100000) / 76 produces a number between 0 and 75, so I add one to it to get a number between 1 and 76. Based on this principle, a sequence of test locations can be determined recursively by updating the seeds to their respective value generated by Generators 1-3.

2.4.2 Example Measurement Sequences

For an example, let x=1, y=10000, and z=3000. Then I obtain $x' = 171$, $y' = 22808$, and $z' = 24832$ and $w = 1.577131$. Since w minus its integer part equals 0.577131, use modulus operator and get value 6, so the Next (fixation light position) will be at the point labeled as 6. According to the algorithms I have developed, first I check the travel time between the last point to this next point to see if it is within the time limit defined, then perform the test at that point, otherwise simply skip this next point and continue to generate the next potential point. To select the consequent next point, I use $x' = 171$, $y' = 22808$, and $z' = 24832$ as input to calculate a new w and then select a point.

2.4.3 Some Modification for the Randomness

Since my specific aim is to select test points so that a patient is unable to predict subsequent test locations over a reasonable period of time, I do not need to generate a truly random sequence for the entire test. Rather I can use a deterministic approach for the generation of test points. The original system was designed to pick the next test location using random shuffle among all the 76 test locations (the whole candidate set). I can use a similar approach, except I can narrow the candidate set or subset the candidate set for the choice of selecting the next test location. Concretely, I can restrict a group of locations, which are relatively close to the last test location and only pick the next test location using random shuffle among this group (a subset of whole test set). The entire sequence will still appear random to the patient, but this randomness is now relative to the each subset of the test locations. This process, by its nature, will reduce test time, while appearing random as long as a reasonable (not too short) travel time limit is provided. I hypothesize that there will be an Optimal degree of travel time restriction that will reduce total test time without significantly decreasing the apparent randomness of the presentation order. In section 3.1.3 I describe in detail how to simulate the Optimized Flicker system (Optimal) based on the algorithm described and compares it to the performance expected with the original Flicker system (Random) using an unmodified random number generation algorithm.

Chapter 3 - Implementation

3.1 Modeling and Simulation of Flicker Program

In order to study the Flicker system scientifically, I have to make a set of assumptions about how it functions. These assumptions, which usually take the form mathematical or logical relationships, constitute a model that is used to try to gain some understanding of how the corresponding system behaves. Since the relationships in Flicker system that compose the model are not simple and involve stochastic processes, it may not be possible to use analytic solution, i.e.; mathematical methods to obtain exact information on question of interest. As a result, I investigated the model means of simulation. In the simulation, I have implemented a computer program to evaluate the optimized and original models numerically, and gathered data to estimate the desired characteristics of the model. The simulation is also more effective for the assessment of a stochastic process as it is far more efficient to run a simulation a thousand times rather than to assess the results of a thousand patients tested on a real perimeter. In the remainder of this section, the flicker system and model are discussed in considerably greater detail and I describe how the program was developed in C++ to simulate the system of interest.

3.1.1 Objectives of simulation

The primary goals of the simulation program are:

1) Designing and reengineering of the program to correspond precisely to the algorithms as well as to reflect the behavioral relationship presented in real flicker system

2) Analyzing and determining the algorithms (which have been developed) to enhance the efficiency of travel and speed up the whole test.

3.1.2 Systems, Models and Simulation

Our system is defined to be a collection of entities, e.g., the movable device, the device for presenting the flickering target, and the patient and technician that act and interact together toward the accomplishment of some particular test. In practical terms, what is meant by "the system" depends on the objectives of my study, which is producing a pseudo-random determination of test location that maximizes the efficiency of travel in the motors governing target projection.

I am only interested in investigating the movable device rather than the modules governing the flickering target and the patient/technician interactions with the perimeter. The collection of the entities that comprise a system for this study is, in reality, only a subset of the whole Flicker system. I define the state of my subsystem to be that a collection of variables necessary to describe a system at a particular time, relative of the objective of the study. Examples of possible variables are number of locations to be test, number of repeat points, estimated travel time from last to next, Optimal degree (maximum travel angle restriction), etc. These variables are described detail in 3.1.5. Since the state of these variables (1) changes instantaneously at separated points in time, (2) the system evolves over different time periods and (3) random number generator is implied as part of the system input, I can categorize the system as a (1) discrete, (2) dynamic and (3) stochastic simulation model.

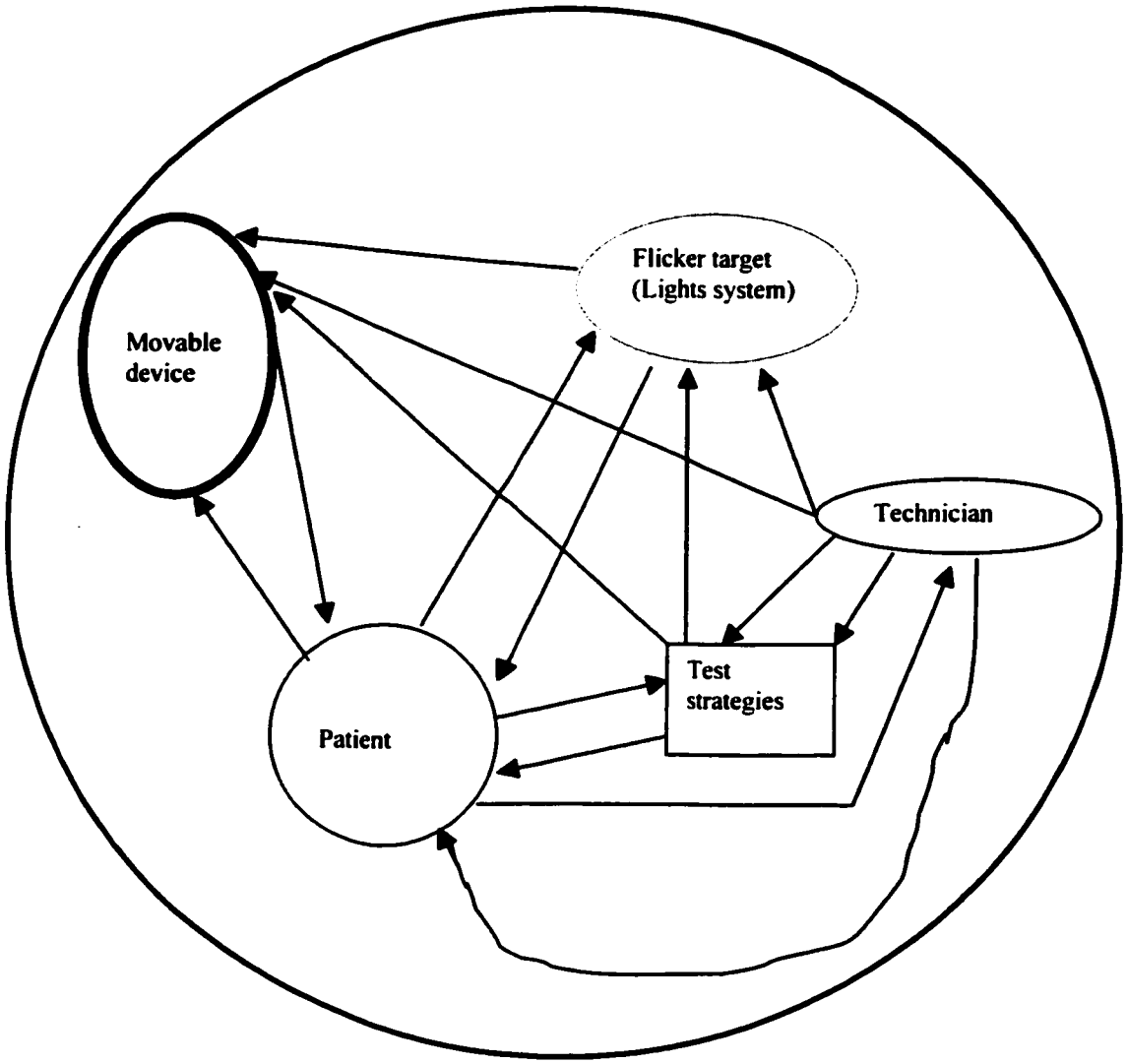


Fig. 13: System decomposition and subsystem

3.1.3 Program Algorithm Development

This section shows in detail how to simulate the Optimized Flicker system based on my basic algorithm (2.3.1). I show every "snapshot" of the simulated program just after each event occurs and explain it intuitively. This section is programming independent.

Now I am ready to describe the basic algorithm in more detail:

A high level algorithm:

A: a set of all points to be visited during a test;
Last: the last position of the fixation light;
Next: the next possible position of the fixation light generated by random generator;
Time-limit: a predefined time period, the fixation light arm moves only if the travel time from the last point to next point with in this limit .

step 0: define a start position
step 1: initialize the Last as the start position;
step 2: while set A is not empty (the test points not all done), generate a random Next position from the all possible positions in set A;
step 3: if the travel time from Last to Next is within the predefined Time-limit initiate the flicker arm movement, present the flicker test and increase the number of visits to that point by 1, if the number of the visits of that Next reached the maximum, delete that point from set A, ie, the test at that point is done;
step 4: repeat from step 2;
step 5: end of the sequence when the test points all done.

This algorithm gives enough detail for us to show a pseudo-random sequence of test locations that will optimize or at least improve the performance of the testing device by minimizing the time required to display the entire sequence.

However, several questions are raised:

1. How can I define or choose the Time-limit? (this is discussed in the next chapter)
2. How can I define the set of the test points? What kind abstract data type will be implemented?
3. What about if the remaining points are all at outside the Time-limit, i.e., how to avoid the infinite loop when the remaining test points are all unreachable according to the Time-limit?

To respond to these questions, I describe the algorithm in more detail. The lower level algorithm can be illustrated by the following pseudocode:

Variables:

- **Set A** implement as a linked list, each node represents a fixation light position, in which there are four variables: label, x, y, record. Label is the numeric symbol of each specific point, range 0-76; x is X-axis value of a test point; y is Y-axis value of a test point, record is the times of visit each point; the maximum node allowed in the list is 77.
- **T [77][77]:** Matrix of type double: size 77X77, element in T [i][j] represents the travel time between test point i and j;
- **Last:** integer, last visit point, using this value I can access the test set A by refer a position label;
- **Next:** integer, next potential visit point generated by random number generator; using the value in Last and Next, access the matrix T, get the travel time
- **Max_visit:** the maximum visit times allowed at that position;
- **Time_limit**: the maximum travel time allowed between two position;
- **MRP: Minimum_Reachable_Percentage**: a predefined ratio that ensure the certain percentage points in set A will be reachable from last position under the current Set of Time-limit.
- **increment:** increase the Time-limit only when all the travel time from the last to all the remaining point in set A exceeded the Time-limit

Functions:

Random-Number-generator(): return a integer (a label of the test point)
randomly generated from all the possible point in set A
The Wichmann Hill random number Generator is implemented here.

Flicker-test(): perform the test;

Update-data-file(): perform the task of recording the test time and sequence;

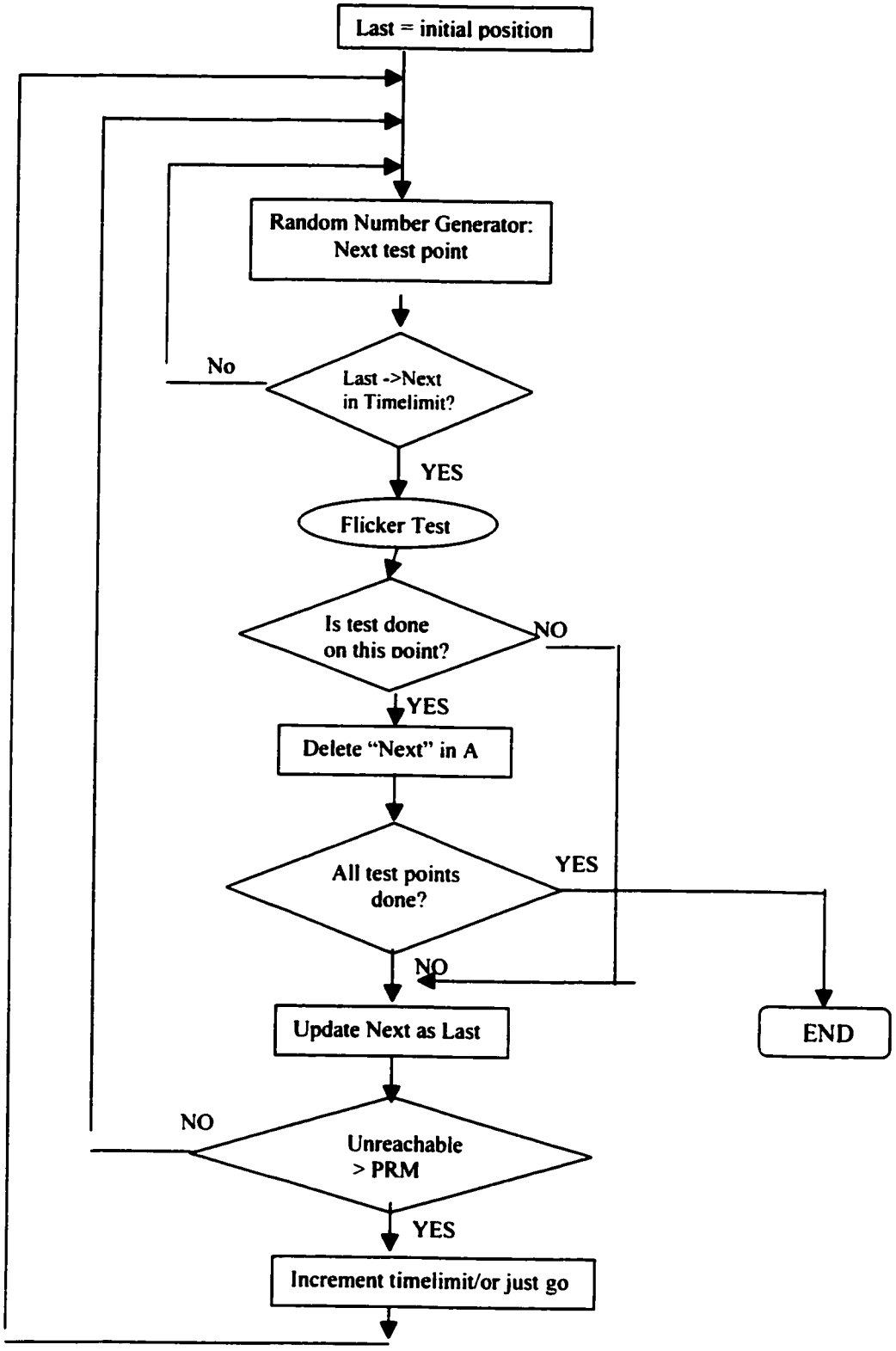
Check-Reachable_Percentage(int MRP, int): it takes a integer which presents the MRP (required reachable_Percentage) and the last test location, return false if the real reachable_Percentage is equal or great than MRP, otherwise return true ;

Reset-Time_limit(): reset back to the initial Time_limit;

Code

```
step1: build the linked list A with 77 nodes: all the 76 predefined test points and
       the initial position;
step2: initialize position last = 0;
step3: while(set A is not empty)
         {
         do
           {
             next = Random-Number-shuffler();     //get next from set A;
             if ( T[last ][ next] < timelimit)
             {
               do Flicker-test();
                 Update-data-file();
                Reset-Time_limit();
               increment the record by 1 at node next;
               if ( record= =Maxvisits or test is done at that point)
                  {delete the next from set A;}
               update last: last  = next;
               repeat from step 3;
              }
             else   repeat from step 3;
       }
        while(Check-Reachable_Percentage() )
           {Time_limit = Time_limit + increment;
            repeat from step 2;}
}
```

Flowchart


Last = initial position
Random Number Generator:
Next test point
No
Last ->Next
in Timelimit?
YES
Flicker Test
Is test done
on this point?
NO
YES
Delete "Next" in A
All test points
done?
YES
NO
Update Next as Last
END
NO
Unreachable
> PRM
YES
Increment timelimit/or just go

3.1.4 Components and Organization of the Simulation Model

- Flicker device control algorithm:
  abstracted from the original flicker program according to the arm movement
- Optimal algorithm:
  has been developed, see 3.1.3
- Database file(time matrix, see Appendix 1):
  database file which stores the travel time between any two test points among 76 points. It was developed in Excel, has been converted to database file format for programming
- Random number generator
  see 2.4

3.1.5 Some constraints and definition for the model

I have defined a set of the system constraints for the models. These include in:

1. **Initial point**: which is the point located in center of the device bowl;
2. **Location set**: all locations of the test points in each test;
3. **Polar movement rate**: 10°/secwhich is calculated from my math model (see1.2.4)
4. **Optimization degree (maximum travel angle restriction)**: a predefined degree for each test in order to set the maximum radial movement and limit the travel time. It can be any number between in 1-180°.
5. **Test time**: since the polar movement takes much longer time than the radial movement and the time of radial movement among all different points is similar, I only try to save travel time on polar movement. So the test time in my model reflects only the polar movement. Obviously, this test time is very close to the real test time.
6. **Long point** which is the any next test point that the travel distance from last test point to the next is longer than optimization degree.
7. **Repeat Points**: any point whose position is same as last test point. Those pair of points is the repeat points.

8. **Mirror points**: which is a set of points that has same polar value but located in opposite location relative to the initial point;

3.2 Model Implementation and Execution

In this chapter, I discuss the details of model implementation in terms of programming, language, interface and execution.

3.2.1 Visual C++ program

Implementation Language: C++

Implementation Specification:

1. Model input:
    - number of test points;
    - test point location;
    - travel time between any two test points(time matrix);
    - optimization degree (maximum travel angle restriction);
2. Model output:
    - total test (travel) time
    - sequence of the all test points
    - number of the special points(long, mirror and repeat points)

Implementation detail:

- **Original system model:**
    1. define initial target point,
    2. randomly generate next test point,
    3. set first test point, second test point, record the travel time t of these two points, virtual test,
    4. add t to total test time,
    5. find next test point, recursively find all test points,
    6. build a test sequence array, calculate the cumulative travel time

- **Optimal system model:**

  1.define initial target point,

  2. find next test point according the Optimal algorithms and the initial point,

  3. set first test point, second test point, record the travel time *t* of these two points, virtual test,

  4. add *t* to total test time,

  5. find next test point, recursively find all test point,

  6. build a test sequence array, calculate the cumulative travel time

Figure 14 provides a detailed description of the modeling and simulation of the Flicker system.

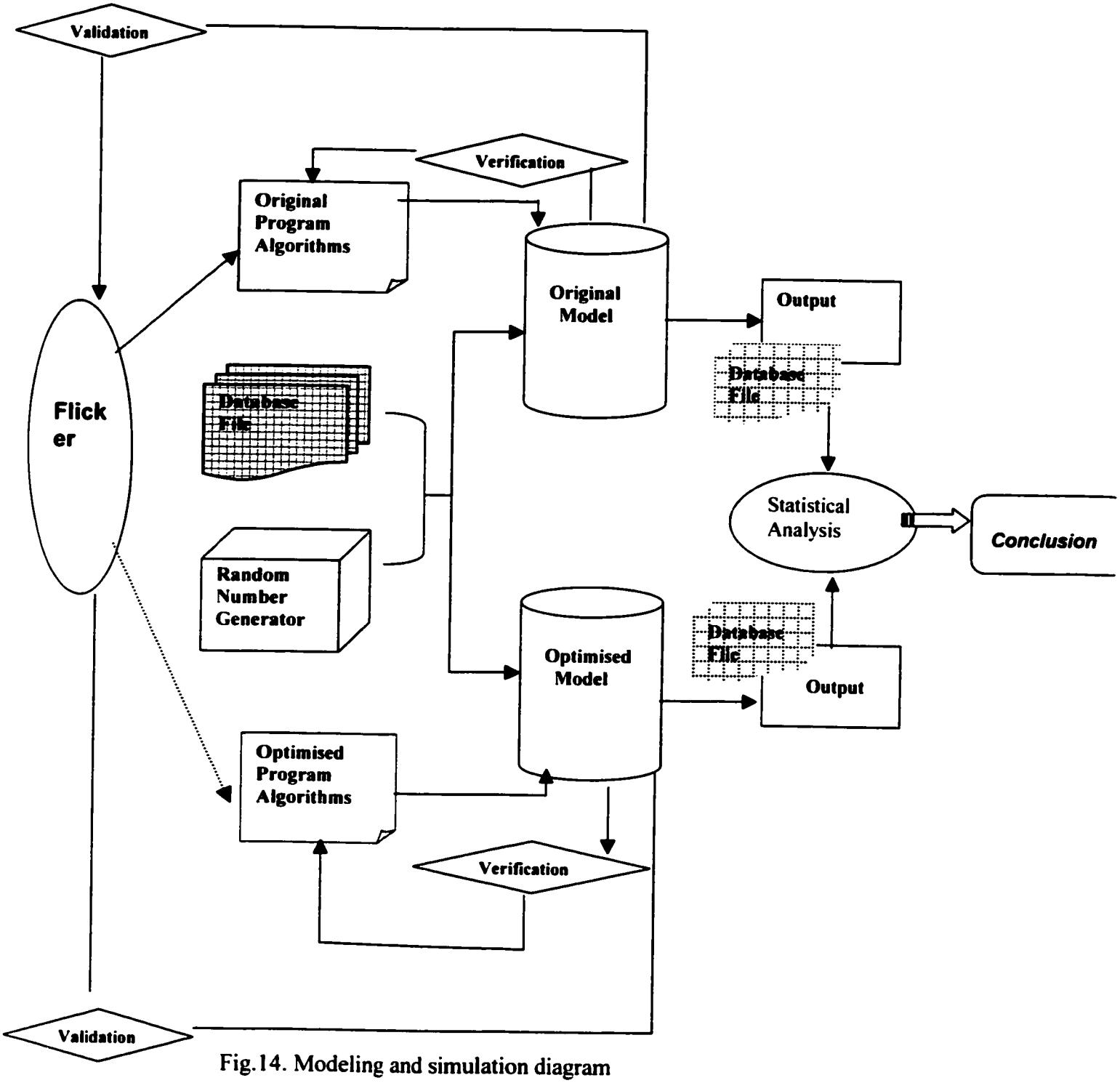
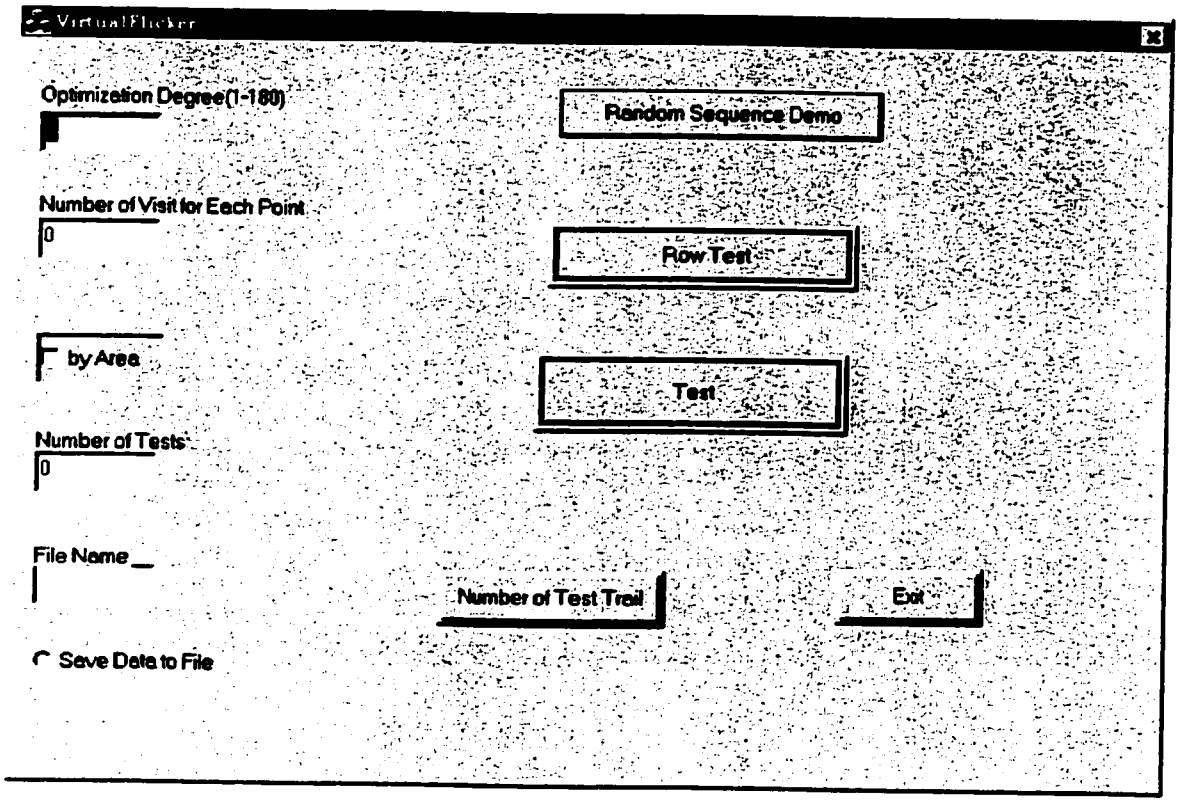


Fig.14. Modeling and simulation diagram

3.2.2 Program execution interface

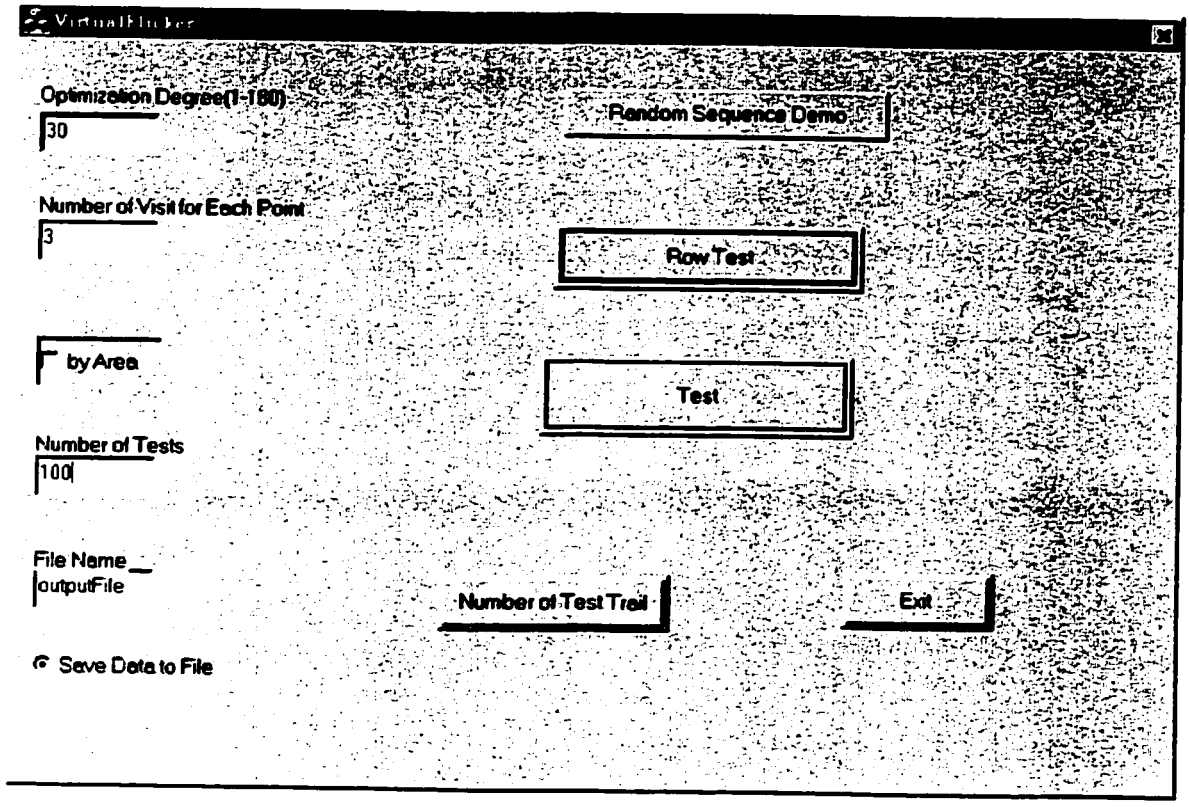
I designed the program interface, which simulates the interface of the actual Flicker. In this interface, one is able to specify any optimization maximum travel angle restriction (Optimal degree), number of visits for each test point, number of tests needed to be done as well as output file name. I have options to do a single test or a specific number of tests for the purpose of statistical analyses.



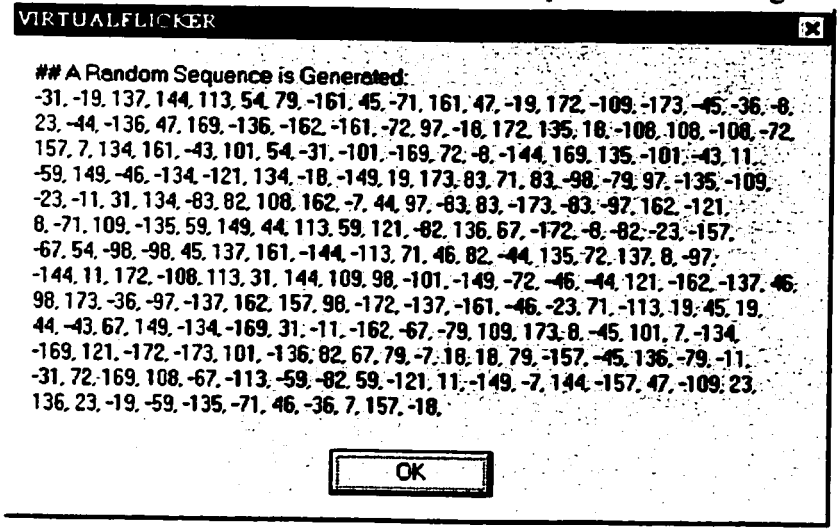
Here is the example of execution of the program:

Input: Optimization maximum travel angle restriction (Optimal degree): 30°, number of visit for each point: 3, number of tests: 100.

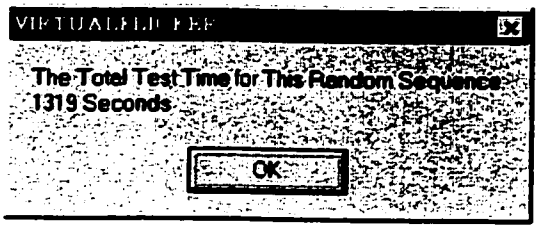
The following demonstrates detail steps. (The program can be run by clicking the option: row test, which complete the all 100 tests and construct the output file.



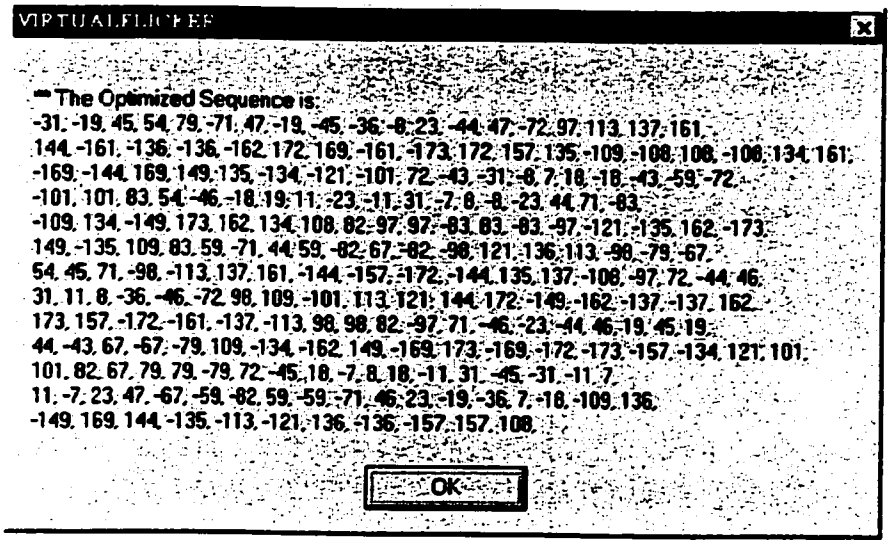
Here is an example of the random test sequence for the original model:



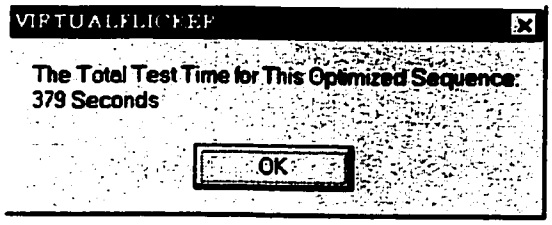
Here is the total test time for the random sequence:



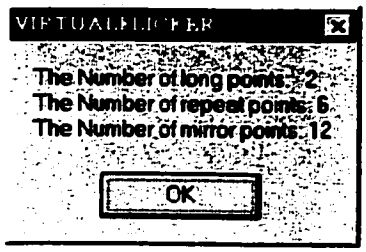
Here is an example of the Optimal test sequence for the Optimal model:



Here is the total test time for the Optimal sequence:



Here is the record for the other important variable values for this simulation:


The Number of long points: 2
The Number of repeat points: 6
The Number of mirror points: 12
OK

3.2.3 Object View of the program

The simulation program is implemented in objected oriented C++ language; that is, the computer program is composed of a group of interacting objects. In order to illustrate the program code behind the VC++ dialogue interface, I explore the object view of the device simulation program as below:

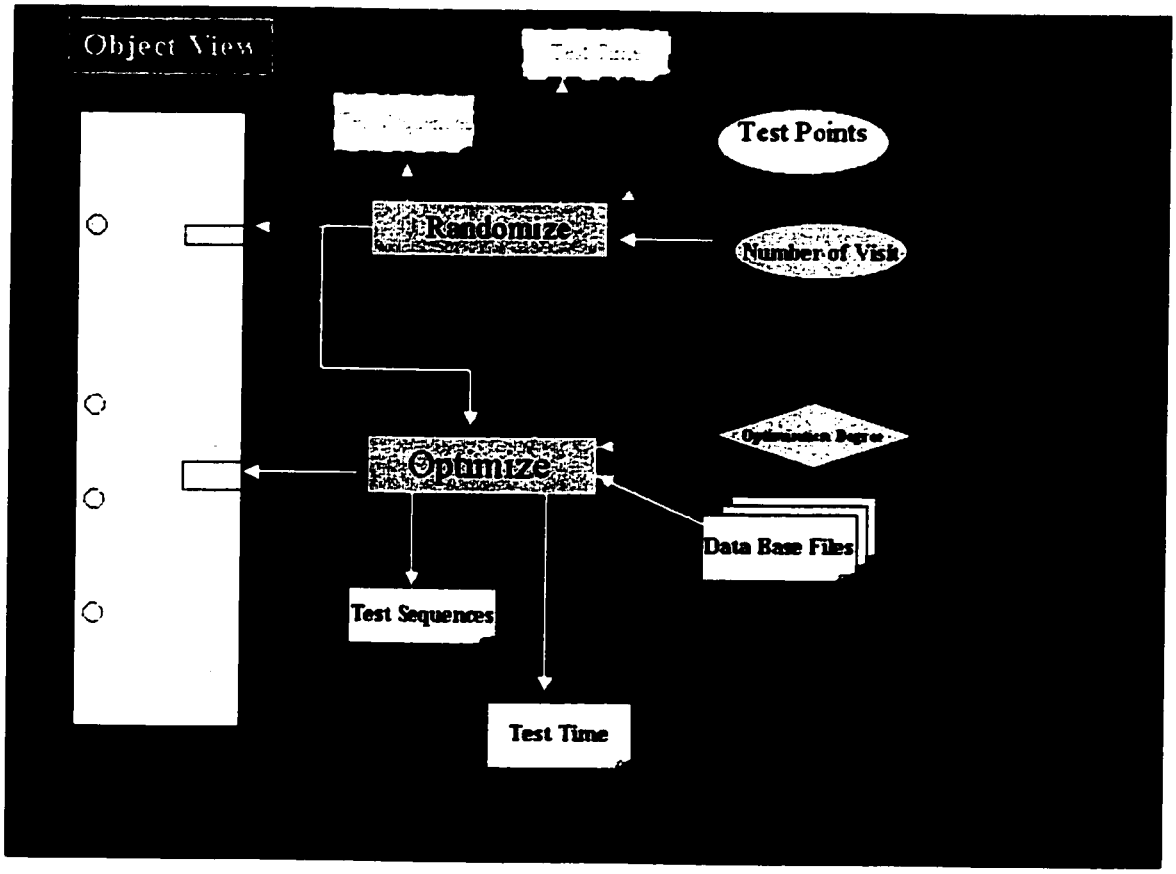


Fig.14. Object view of the simulation program

Chapter 4 – Results and Discussion

4.1 Model Verification and Validation

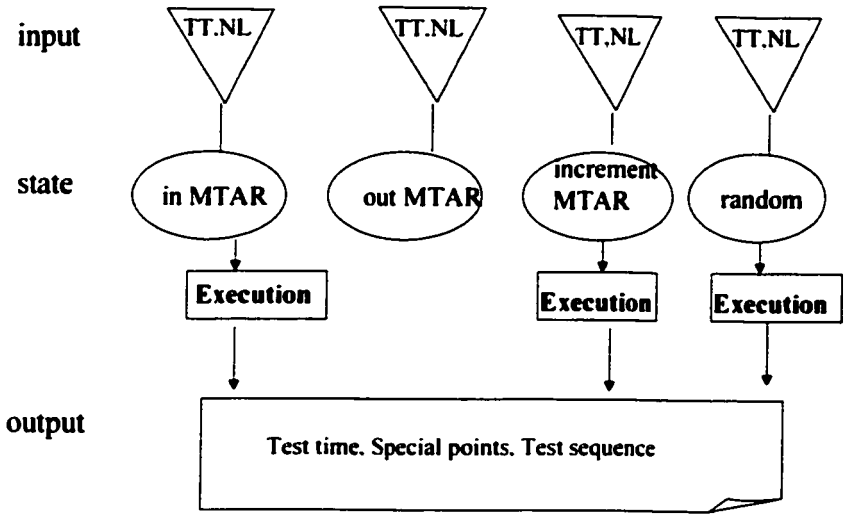
This section describes the verification of the effectiveness of the simulation program (check for program error) and the validation of the accuracy of the simulation program (check for model error).

4.1.1 Model program and the algorithms

Verification is the attempt to establish that a direct relation holds between a simulation program and a model. There are two general approaches [B. P. Zeigler and H. Praehofer, 2000]:

- Formal proofs of correctness
- Extensive testing

Formal proofs employ mathematical and logical formalisms underpinning the concepts in the system's specification to establish the requisite morphism. Unfortunately, such proofs are difficult to carry out for a complex stochastic system. However, the morphism concept comes in handy since I can verify the program with the algorithms (or flow chart) by laying out the combination of the inputs and states that have to be tested for completeness. In these studies, I have carried out all of these steps and demonstrated that the computer programs correspond precisely to the model I defined (see Figure 16)



NL: Next location. MTAR: maximum travel angle restriction. TT: travel time

Fig. 16: Verification the program with the algorithms from the combination of the inputs and states demonstrated that the computer programs correspond precisely to the model I defined

4.1.2 Model program and real system

Validation is the process of testing the program against the model for validity.

I invoke the experimental frame concept to perform the validation process. [Bernard P. Zeigler, 2000] The experimental frame is a specification of the conditions under which the system is observed or experimented with. As such, an experimental frame is the operational formulation of the objectives that motivate a modeling and simulation project. Figure 17 illustrates how the experimental frame is critical in this process. The frame generates input parameter sets (trajectories) to both the original system (Random) and the Optimal model (Optimal). The corresponding output parameter sets (trajectories) of the model and systems are fed back into the frame. According to this approach, validity requires that these trajectories be equal. The experimental frame provides the conditions for the both model and original system to be assessed. The frame in my studies is decomposed into (1) generator –random number generator, (2) acceptor –test

points queue, (3) transducer –turnaround time (whole test time) and (4) failure rate (special points).

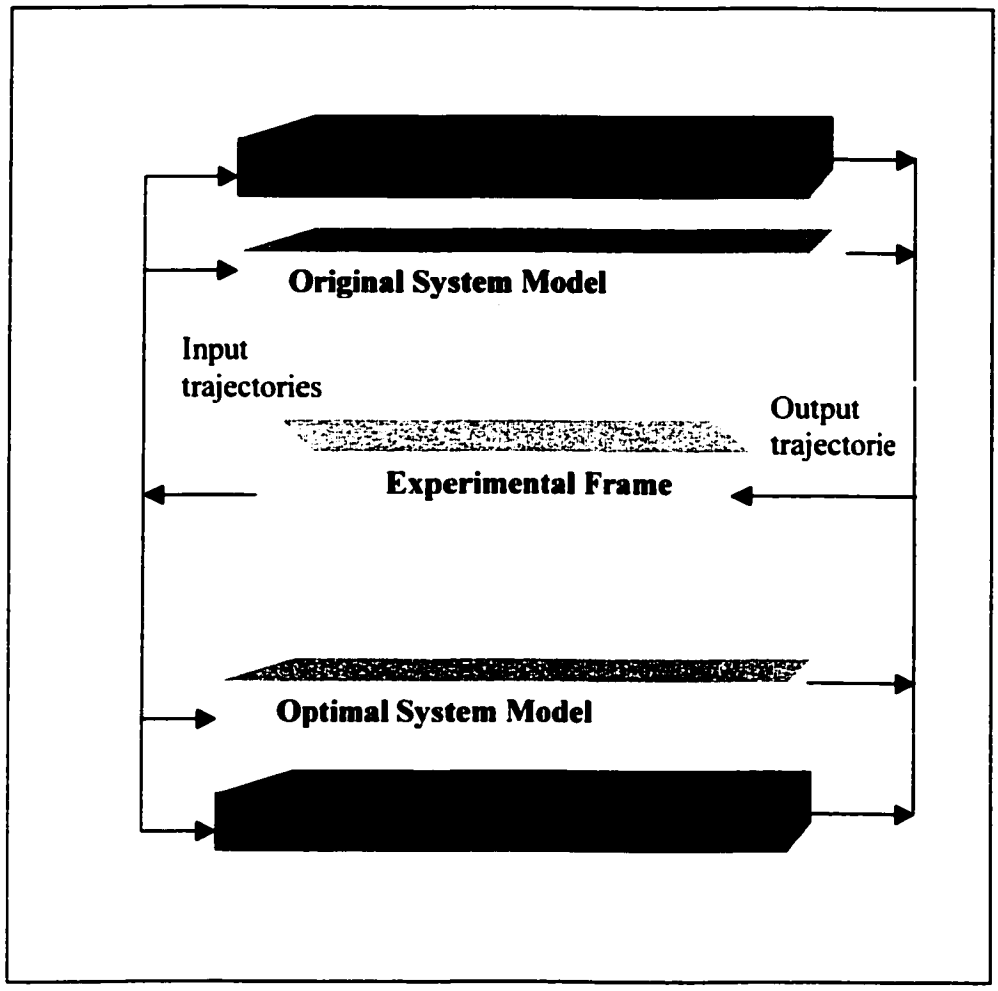


Fig. 17: Validation in experimental frame

Validation requires the comparison of model and source system for both original and optimal systems. The comparison is only performed under the conditions of experimentation specified in the frame: input trajectories generated by the generator, the control condition checked by the acceptor, and the output summaries performed by the transducer. Theoretically, the results from both source and optimal systems, which

followed the same generator, acceptor and transducer, should be compared in my studies.

Unfortunately, clinical data comparing the Optimal and original models is not available; thus I can not access the source system to acquire the result empirically. However, I have abstracted and analyzed the actual system, and then followed the same generator, acceptor and transducer concepts in both models, so the comparison can be made logically.

4.2 Results and Statistical Analyses

In this section, I discuss the empirical results from the simulation program. I have assessed the simulation program extensively using a range of input parameters, i.e., maximum travel angle restriction (Optimal degree), number of visits in each test location, total test points and locations and the number of interactions in each series. The number of simulations done for each group is listed below.

Table 1. The number of simulations done in different groups

Optimal Iterations	10°	30°	50°	90°	150°	Original
30	5	10	10	10	5	40
100	3	3	3	5	3	17
1000			3			3

The results of each simulation are described in detail below.

4.2.1 Sample data output and comments

After each execution of the simulation program, the output data are written to data files. The data output is composed two parts. The output file 1 contains the whole test sequences both for Random and Optimal algorithm. The output file 2 contains number of tests, number of visit per test point, maximum travel angle restriction (Optimal degree), the total test time of random sequence and optimal sequence, as well as the

statistical data for all the special points, i.e., the number of repeat points, long points and mirror points as defined in 2.3.4. These data are used to assess the pseudo-random characteristics of the algorithm.

Here is part of a sample output file from the program. A complete set of information is available in Appendix 2:

Test	Size	O_degree	R_time	O_time	R_Long	R_Repeat	R_Mirror	O_Long	O_Repeat	O_Mirror
1	3x76	30	1356	346	159	3	2	3	7	11
2	3x76	30	1315	361	153	0	1	3	5	19
..										
..										

where O_degree means maximum travel angle restriction (Optimal degree), R_time means test time for random sequence(O_time for Optimal sequence), R_Long, R_Repeat and R_Mirror mean number of long , repeat, mirror points in Random sequence.

4.2.2 Data analyses

The data is analyzed to investigate three specific questions:

1. What is the minimum size of the simulation run that provides a reliable and accurate information about the algorithms?
2. Does the Optimal algorithm save time compare to the Original Random algorithm? What happens to the peudo-randomness of the algorithm with increasing maximum angle restrictions reduces the total time?
3. Within the possible range of maximum angle restrictions (Optimal degree), what value for the maximum angle restriction in the Optimal simulation provides the most reduction in test time while preserving the pseudo-random characteristics of the testing sequence?

1. Simulation size:

In order to determine the impact of the size of test groups (i.e., 30, 100, 1000), I first look at all the series with a maximum travel angle restriction (Optimal degree) of 50° but with varying number of tests in simulation. The results are outlined in figure18 - 21, table 2 – 5:

Table 2. Optimal and Random Test time in different groups

Number of tests	TotalTestTime(± SD) Random	Optimal
30	1362(±52.3)	541(±15.7)
100	1361(±57.6)	542(±21.2)
1000	1369(±60.4)	541(±21.2)

Table 3. Long points in different groups

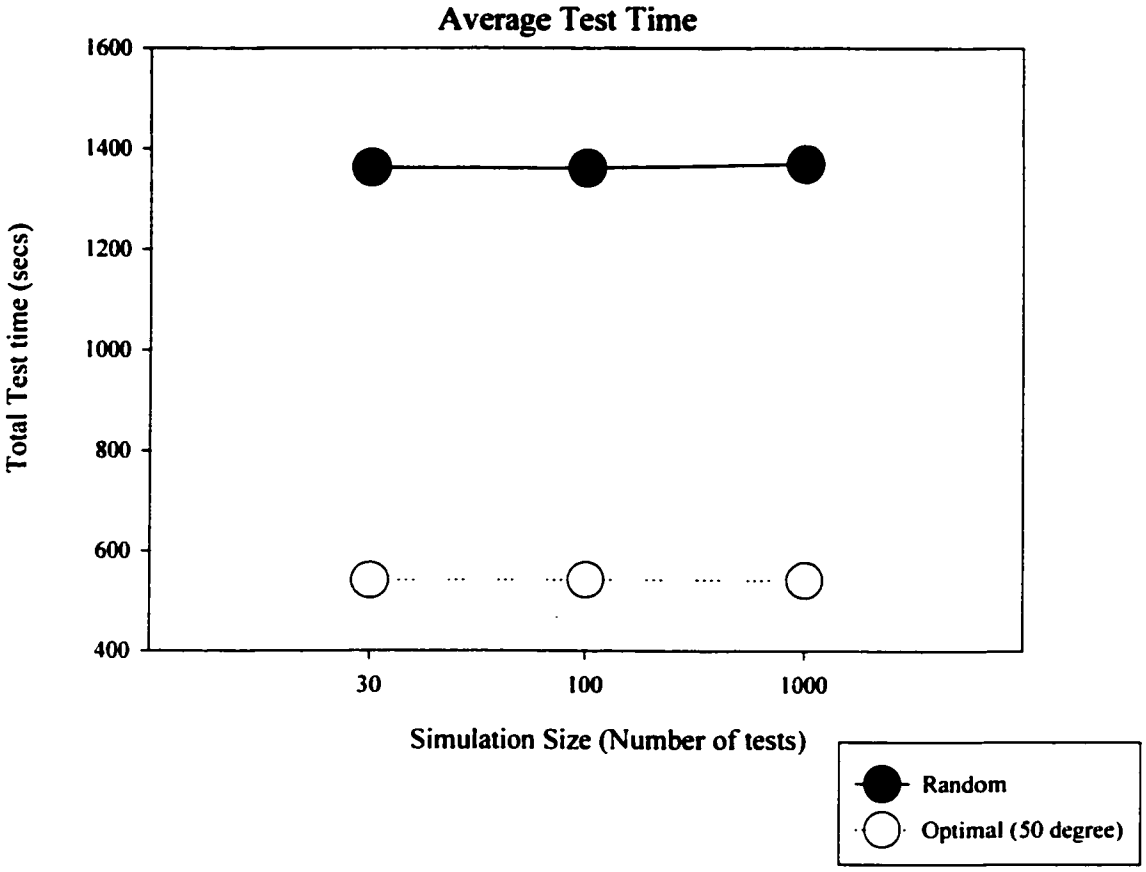
Number of tests	**Total Long Points** Random Seq.	(± SD) Optimal Seq.
30	126(±11.0)	2.4(±0.5)
100	118.4(±12.5)	2.0(±1.8)
1000	120(±12.7)	1.9(±1.9)

Table 4. Repeat points in different groups

Number of tests	**Total Repeat Points** Random Seq.	(± SD) Optimal Seq.
30	1.8(±1.2)	2.4(±0.8)
100	2(±1.4)	4.2(±1.9)
100	2(±1.5)	4.1(±2.0)

Table 5. Mirror points in different groups

Number of tests	**Total Mirror Points** Random .	(± SD) Optimal Seq.
30	1.4(±0.5)	4.3(±0.5)
100	2(±1.8)	6.1(±2.1)
1000	2(±1.6)	6.1(±2.4)



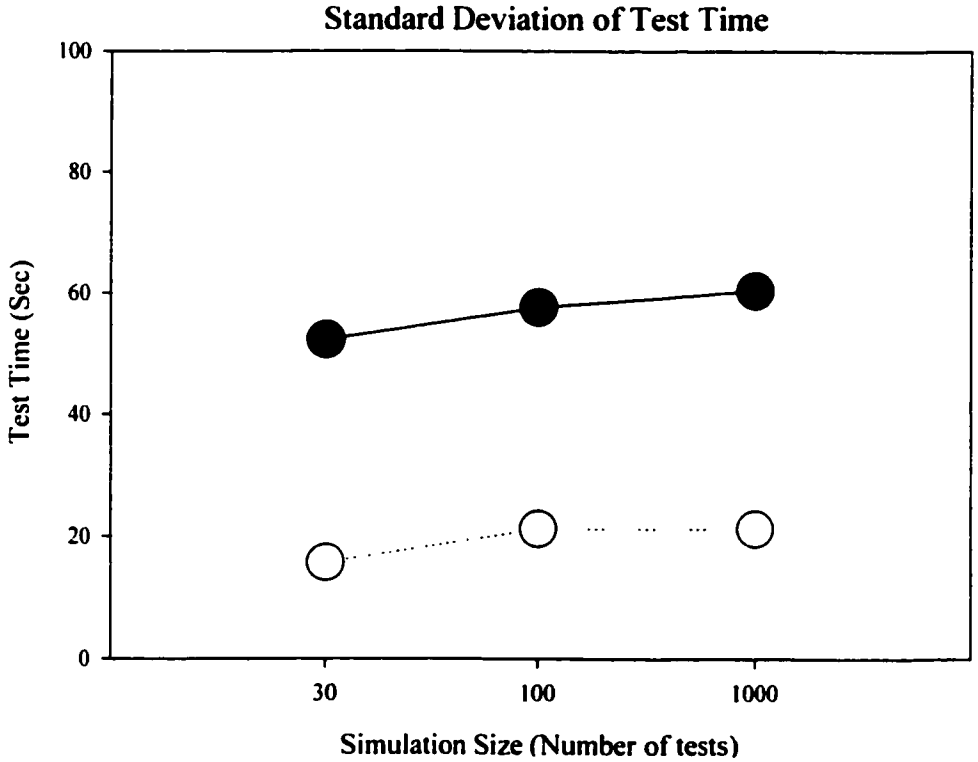
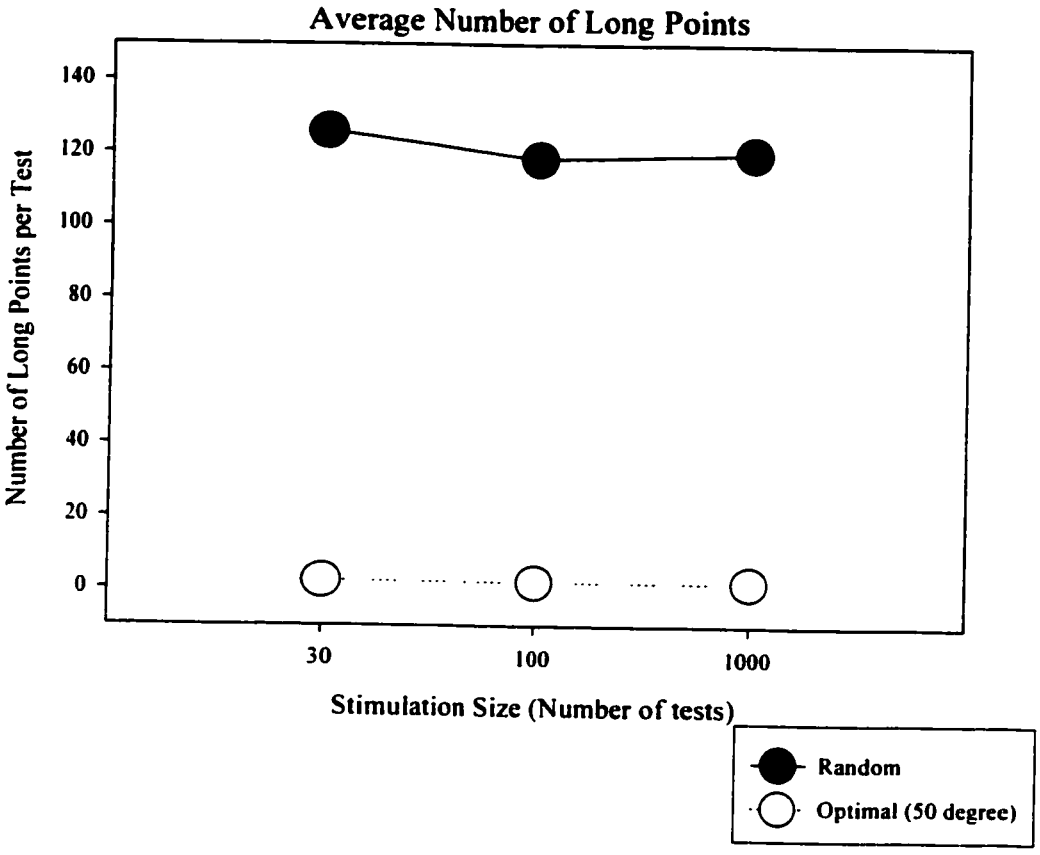


Fig.18: Average test time and standard deviation in different groups



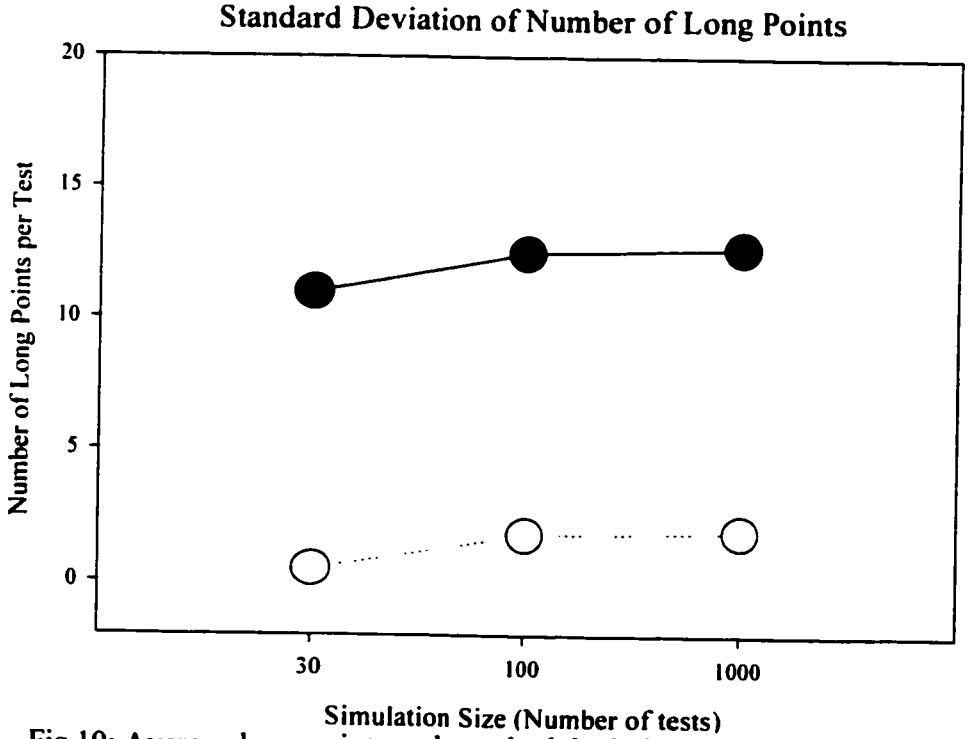


Fig.19: Average long points and standard deviation in different groups

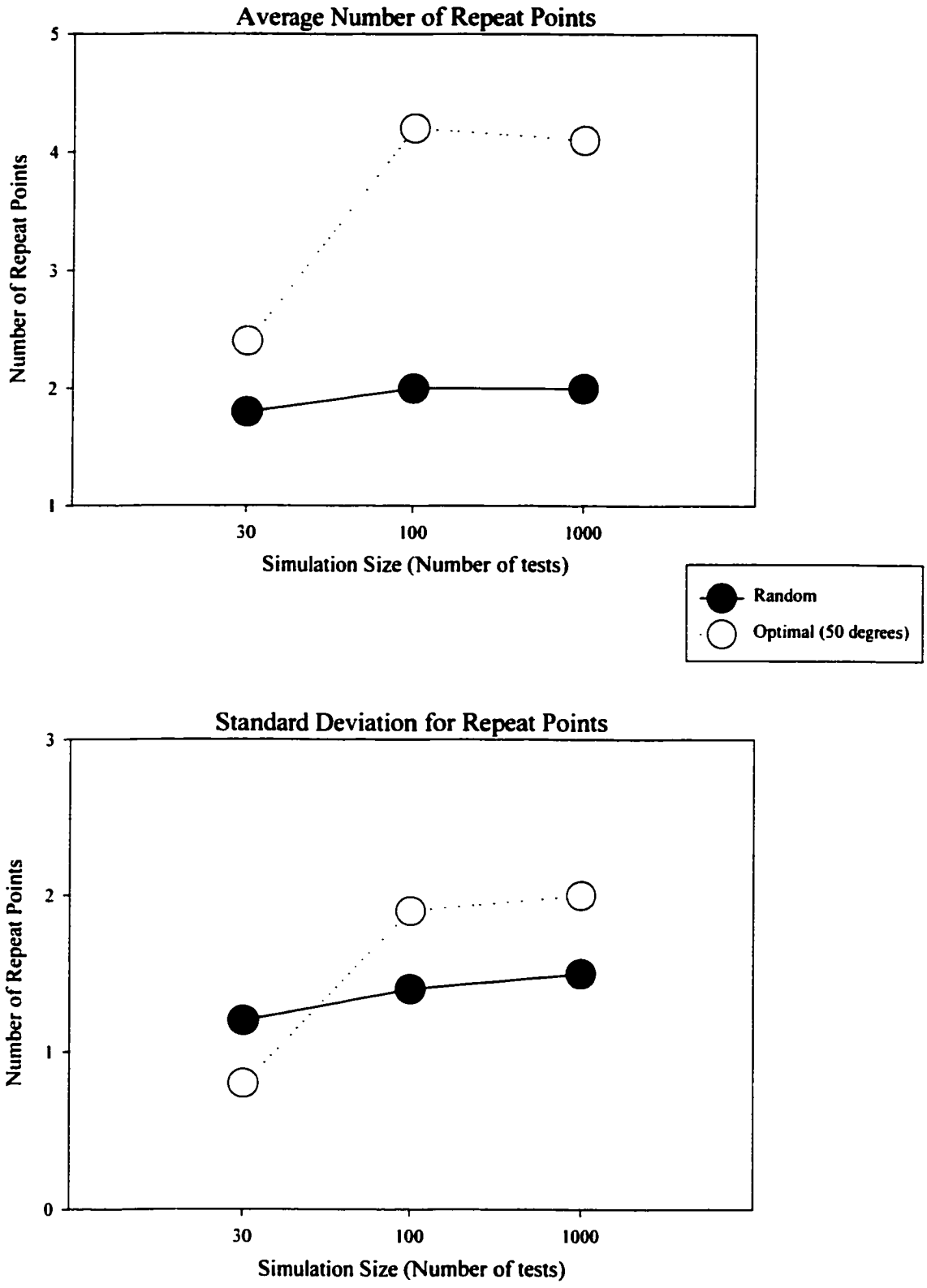


Fig.20: Average repeat points and standard deviation in different groups

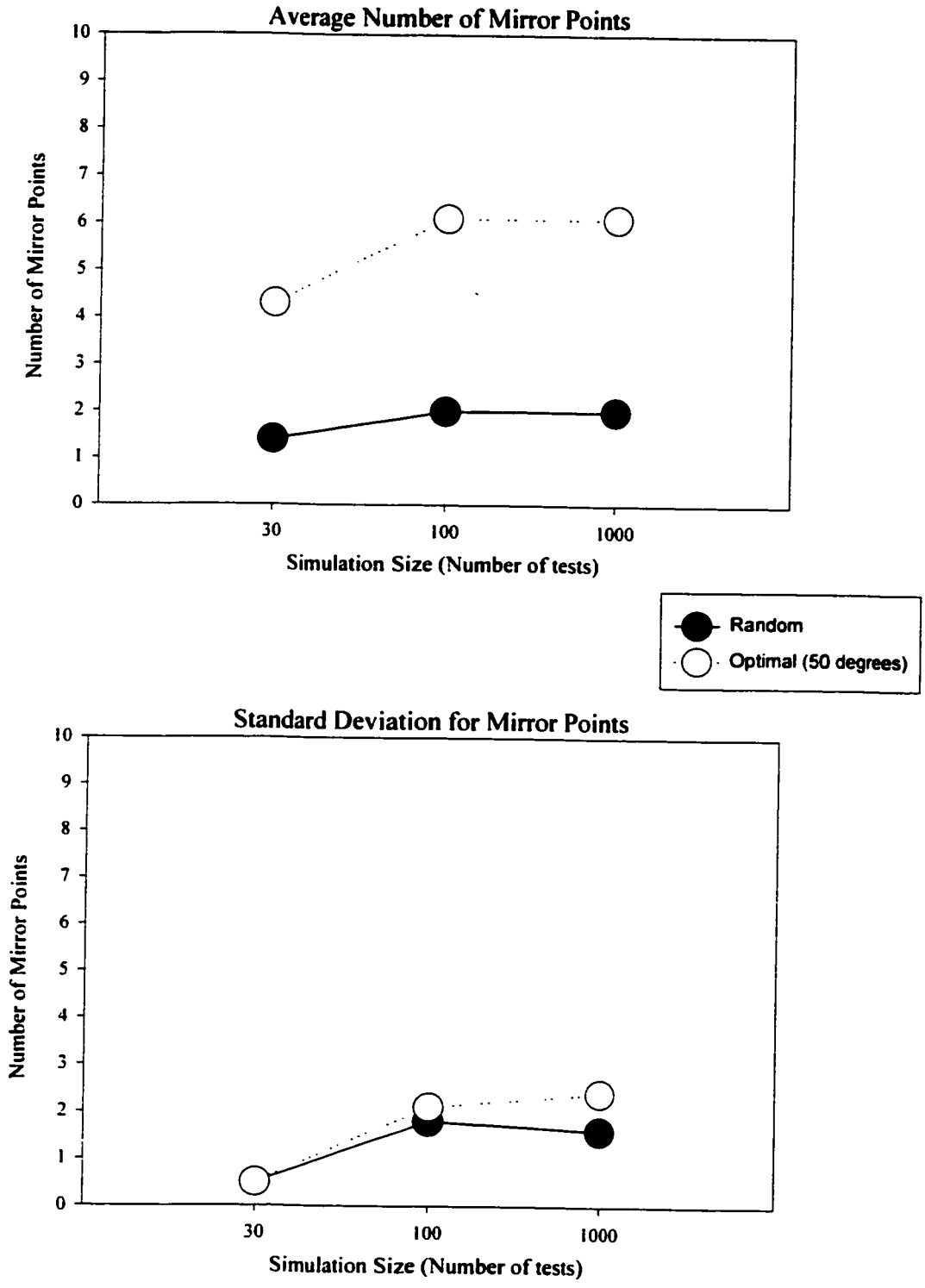


Fig.21: Average Mirror points and standard deviation in different groups

Above tables and figures show the average test time, average number of long points, repeat points and mirror points as well as the standard deviations in each test group with the same maximum travel angle restriction (Optimal degree 50°) but with increasing number of simulation tests. I observe that the results are very stable over the number of iterations from 30, 100, to 1000 runs per simulation. The results demonstrate that a simulation run of 100 is adequate to provide stable information (with SD values similar to those of 1000 iteration runs) on algorithm characteristics. Therefore it is not necessary to run 1000 simulation tests.

2. Impact of Changing Maximum Travel Angle Restriction

In this section, I present the results of all 100 iteration simulations to look the impact of the different maximum travel angle restriction (Optimal degree) on both total test time and pseudo-random characteristics

Comparison of the Random and the Optimal algorithms in 100-iteration simulations:

Table 6. Test time in different maximum travel angle restriction (Optimal degree)

	Test time				
	10°	30°	50°	90°	150°
Random	1360(±64.1)	1359(±57.9)	1361(±57.6)	1372(±52.9)	1364(±54.8)
Optimal	149(±14.6)	360(±14.9)	542(±21.2)	840(±38.6)	1321(±54.9)

Table 7. Long points in different maximum travel angle restriction (Optimal degree) groups

	Number of long points				
	10°	30°	50°	90°	150°
Random	201(±9.1)	155(±12.0)	118.4(±12.5)	71.0(±10.3)	7.5(±4.3)
Optimal	8.3(±3.5)	2.4(±2.1)	2.0(±1.8)	1.4(±1.5)	1.2(±1.6)

Table 8. Repeat points in different maximum travel angle restriction (Optimal degree) groups

	Number of repeat points				
	10°	30°	50°	90°	150°
Random	2.3(±1.5)	2(±1.4)	2(±1.4)	1.9(±1.5)	2.2(±1.6)
Optimal	18.2(±4.4)	5.7(±2.3)	4.2(±1.9)	3(±1.7)	2.2(±1.6)

Table 9. Mirror points in different maximum travel angle restriction (Optimal degree) groups

	Number of mirror points				
	10°	30°	50°	90°	150°
Random	2.9(±1.7)	2(±1.8)	2(±1.8)	1.9(±1.7)	2.2(±1.8)
Optimal	26.7(±5.0)	9.1(±3.0)	6.1(±2.1)	4.1(±1.8)	2.8(±1.9)

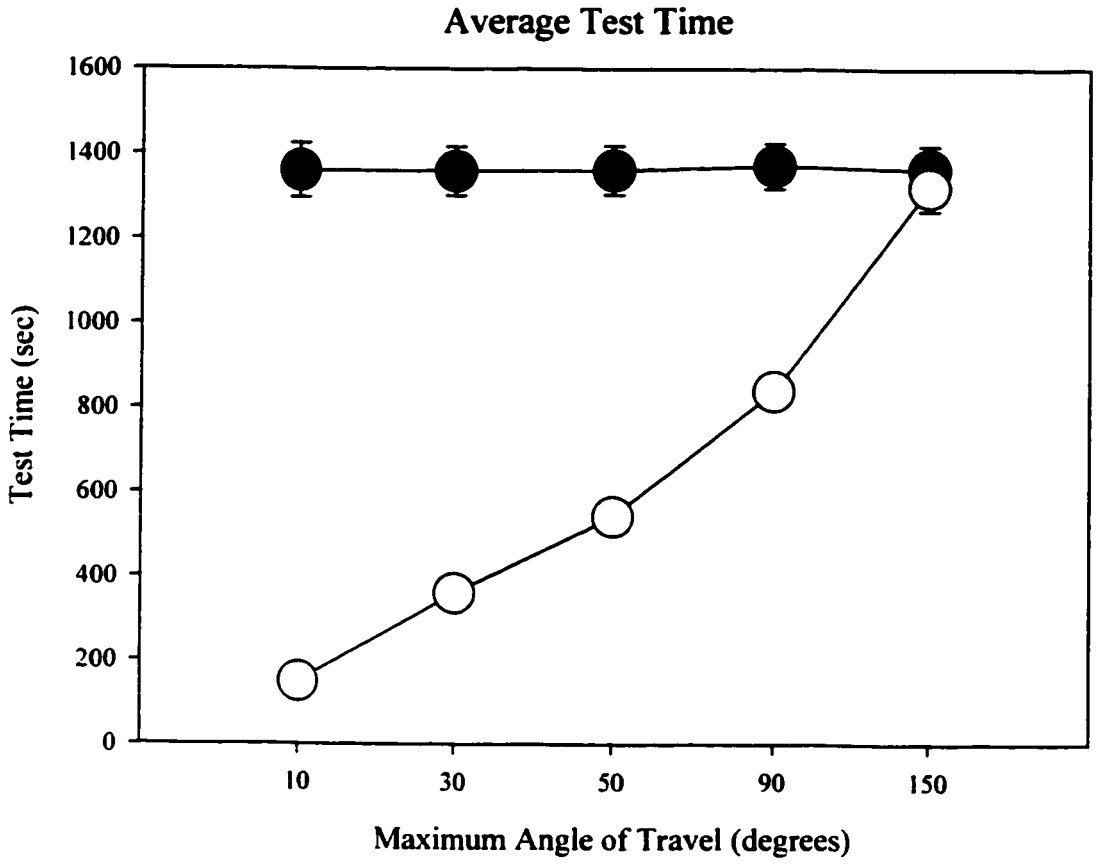


Fig.22: Average test time and standard deviation in the groups with different Maximum angle restriction.

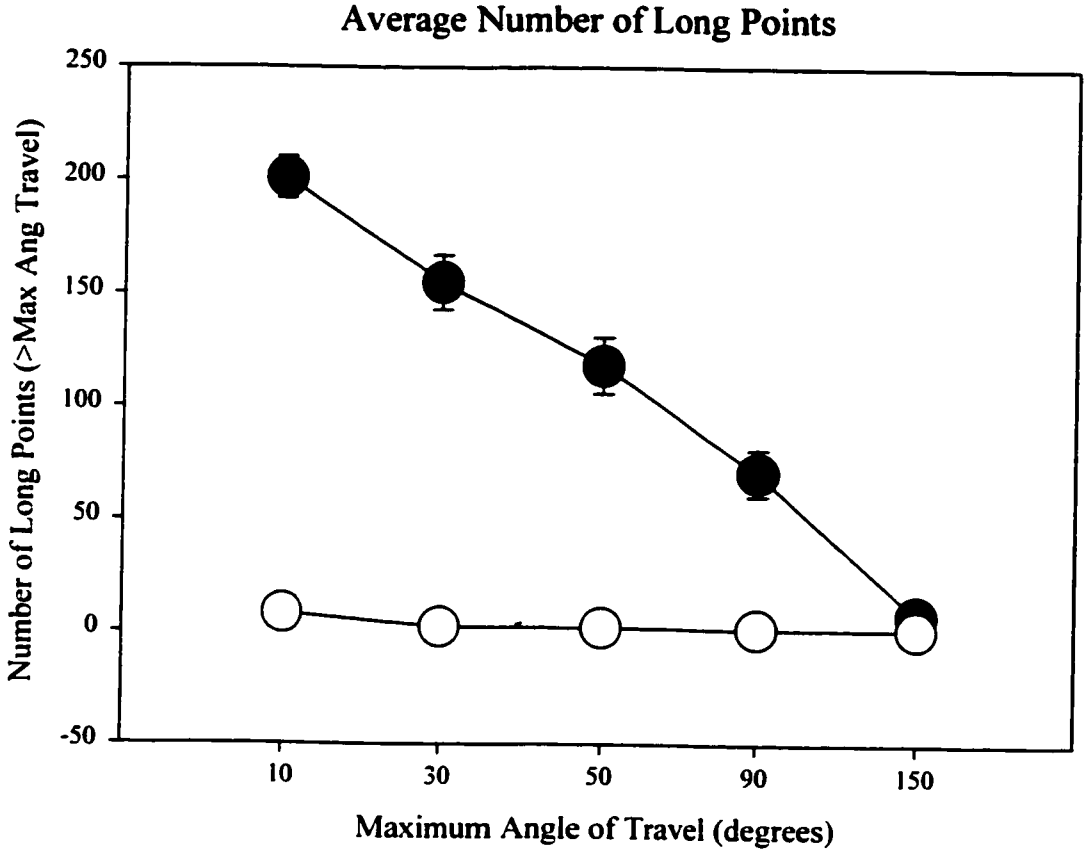
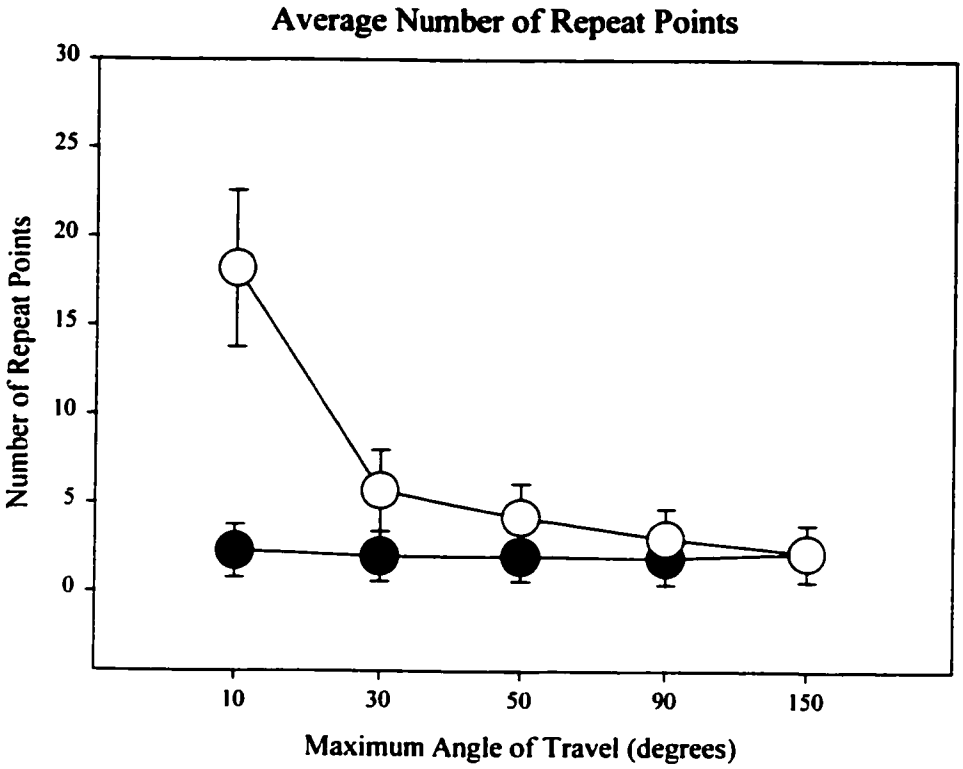


Fig.23: Average long points and standard deviation in the groups with different Maximum angle restriction.



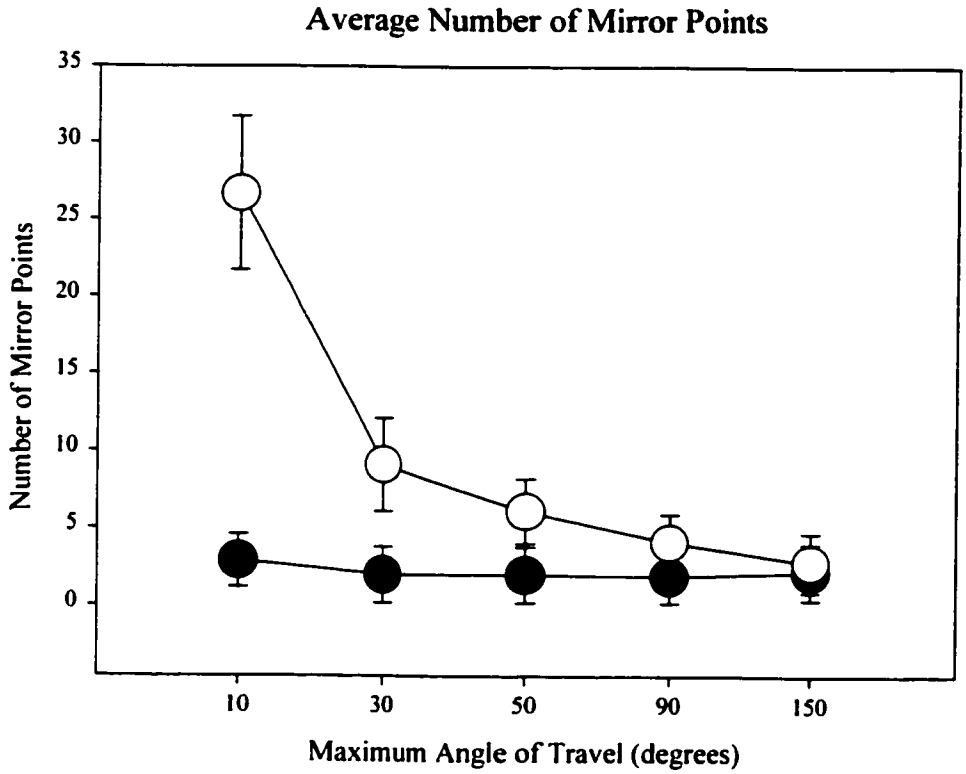


Fig.24: Average Repeat and Mirror points and standard deviation in the groups with different Maximum angle restriction

Above tables and figures show how the average test time, average number of long points, repeat points and mirror points vary with changes in the maximum travel angle restriction (Optimal degree) from 10, to 150 degrees. An estimate of variability is given for each value in terms of the SD from 100 data points.

As I implemented both optimal and random test sequences in the simulation program, each execution results two sets of data, one for the optimal sequence and one for the random sequence. These sets of data are the result of the same input data. However, due to differences between the two algorithms, the maximum travel angle restriction (Optimal degree) applies primarily to the Optimal simulation. Each random sequence is comparison group for the corresponding optimal sequence. For the measurement of the test time in random groups, whatever the input of the maximum travel angle restriction (Optimal degree) is, the Random test simply ran the random sequence. For the measurement of the long points, repeat points and the mirror points in random groups, whatever the input of the maximum travel angle restriction (Optimal degree) is, the test simply recorded these special points according to the corresponding criteria in the optimal sequence of the same test. As demonstrated below, the maximum travel angle restriction does impact the number of long points recorded from the random simulation, as the criterion is defined as those points exceeding the maximum travel angle restriction.

As seen in Figure 22 and Table 6, decreasing the value of the maximum travel angle restriction (Optimal degree) from 150 to 10 degrees decreased the total test time in the Optimal simulation. Figure 23 and Table 7 demonstrate that, as expected, the number of long points; that is the points that exceed the defined time of the maximum angle restriction, increase for the simulation of both the Optimal and more dramatically for the original Random algorithm. Restricting the maximum angle also increases the number of repeat and mirror points in the simulation of the Optimal algorithm, but obviously not a factor in the simulation of the original Random algorithm (Figure 24 and Table 8,9)

Figures 25 to 28 show how the estimates of total test time, long points, mirror and repeat points vary from test run to test run across the 100 iterations of the simulation. The average variability is given in terms of the standard deviation (SD) in Table 10 This demonstrates that my results are very stable across tests. The regression line is also given for each data set. While slopes vary there is no consistent trend in the slopes towards either negative or positive values over repeated simulations of 100 iterations (Table 11).

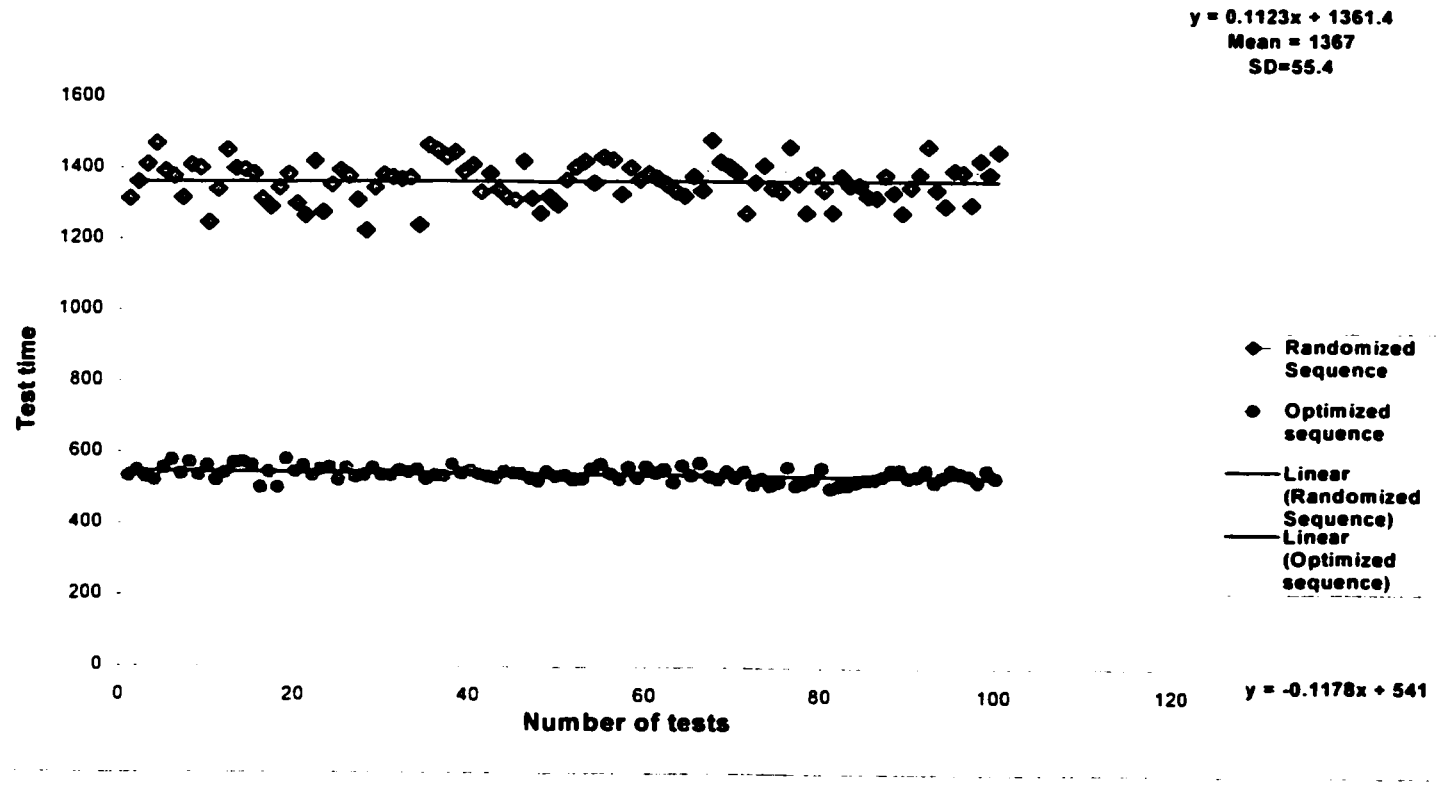


Fig. 25: Optimal sequence test time with maximum travel angle restriction (Optimal degree) 50° Vs random sequence test time.

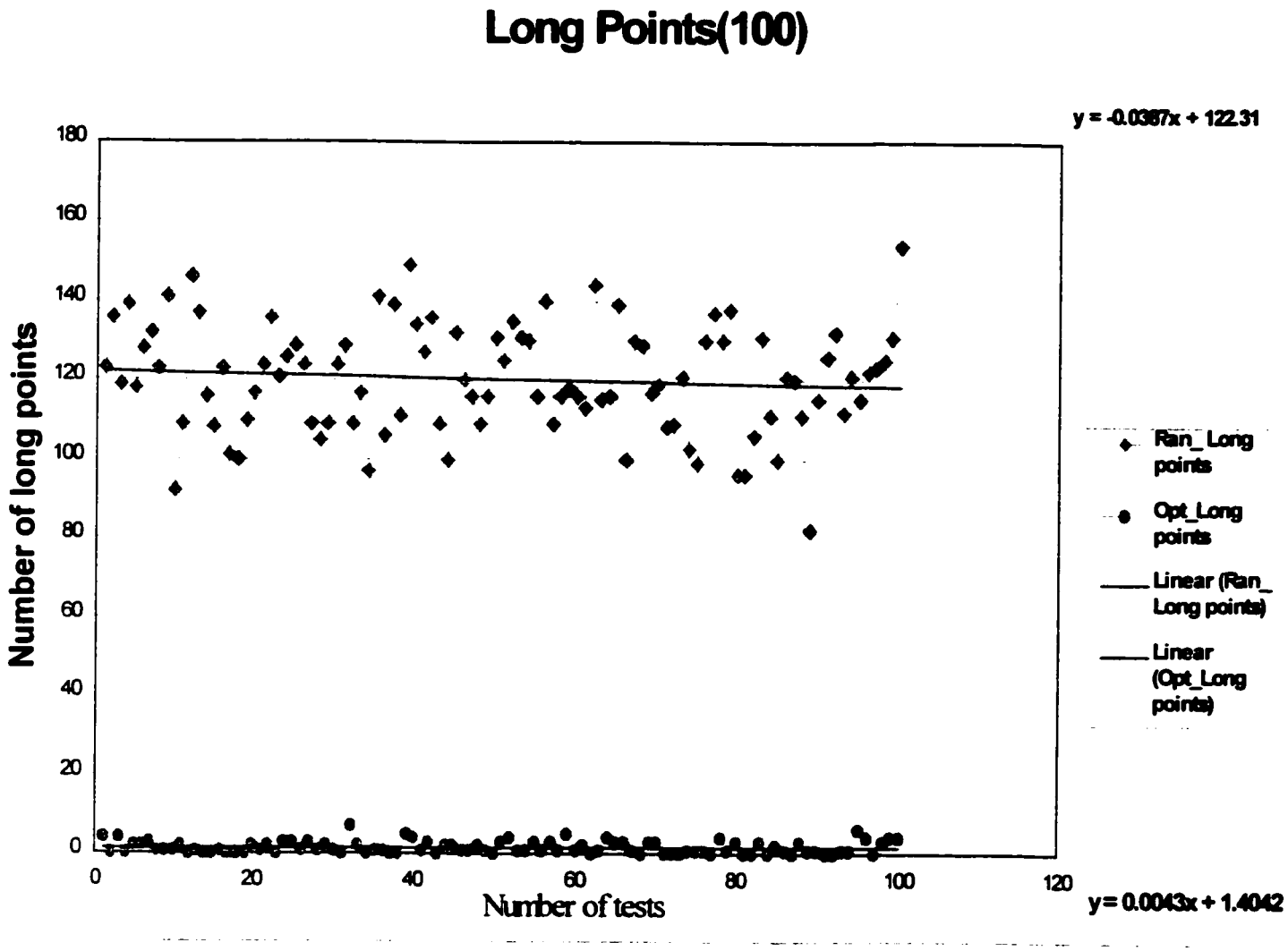


Fig. 26 Long points in Optimal sequence Vs Random sequence with maximum travel angle restriction (Optimal degree) 50°

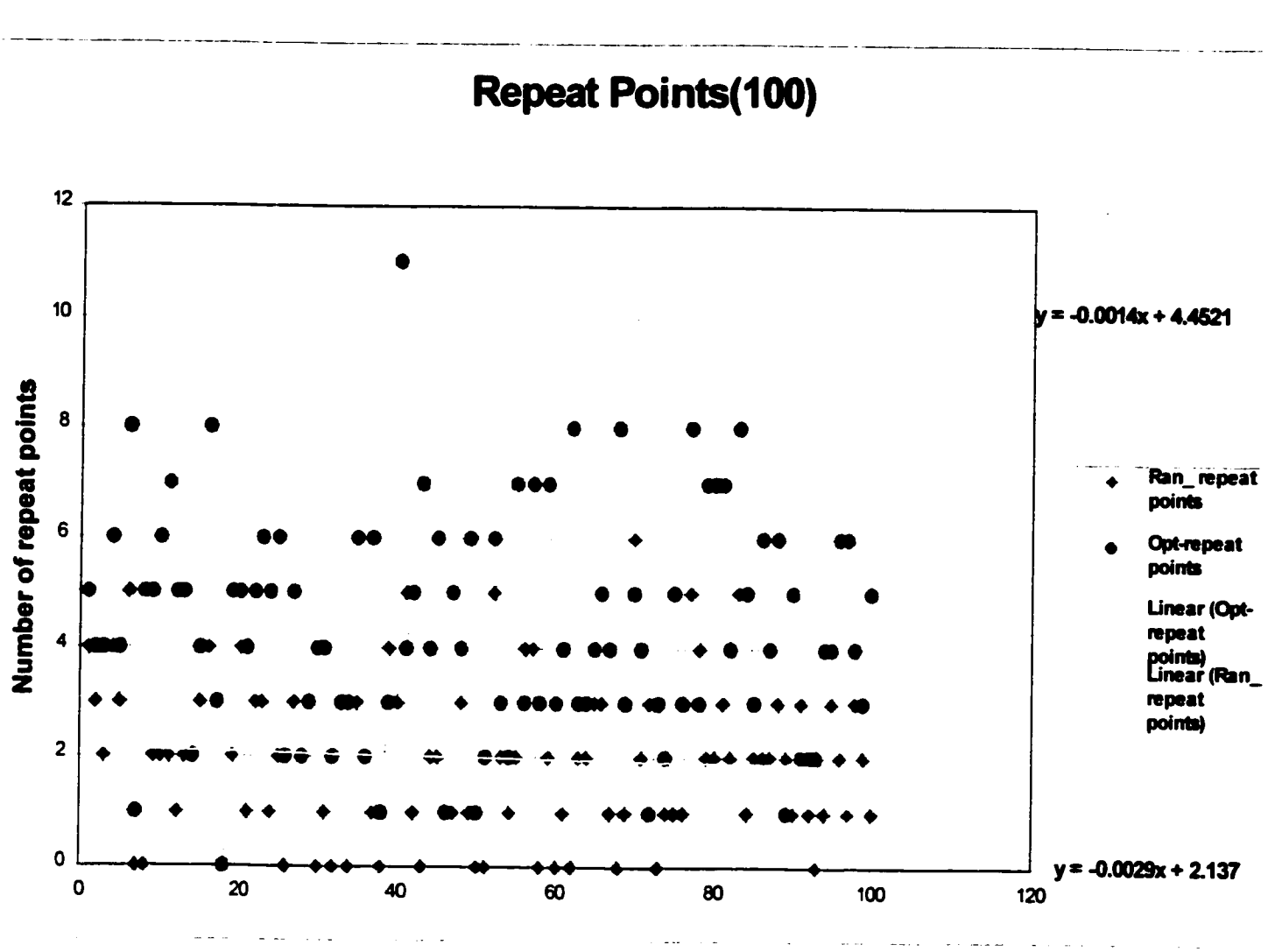


Fig. 27: repeat points in optimal sequence Vs in Random sequence with maximum travel angle restriction (Optimal degree) 50°

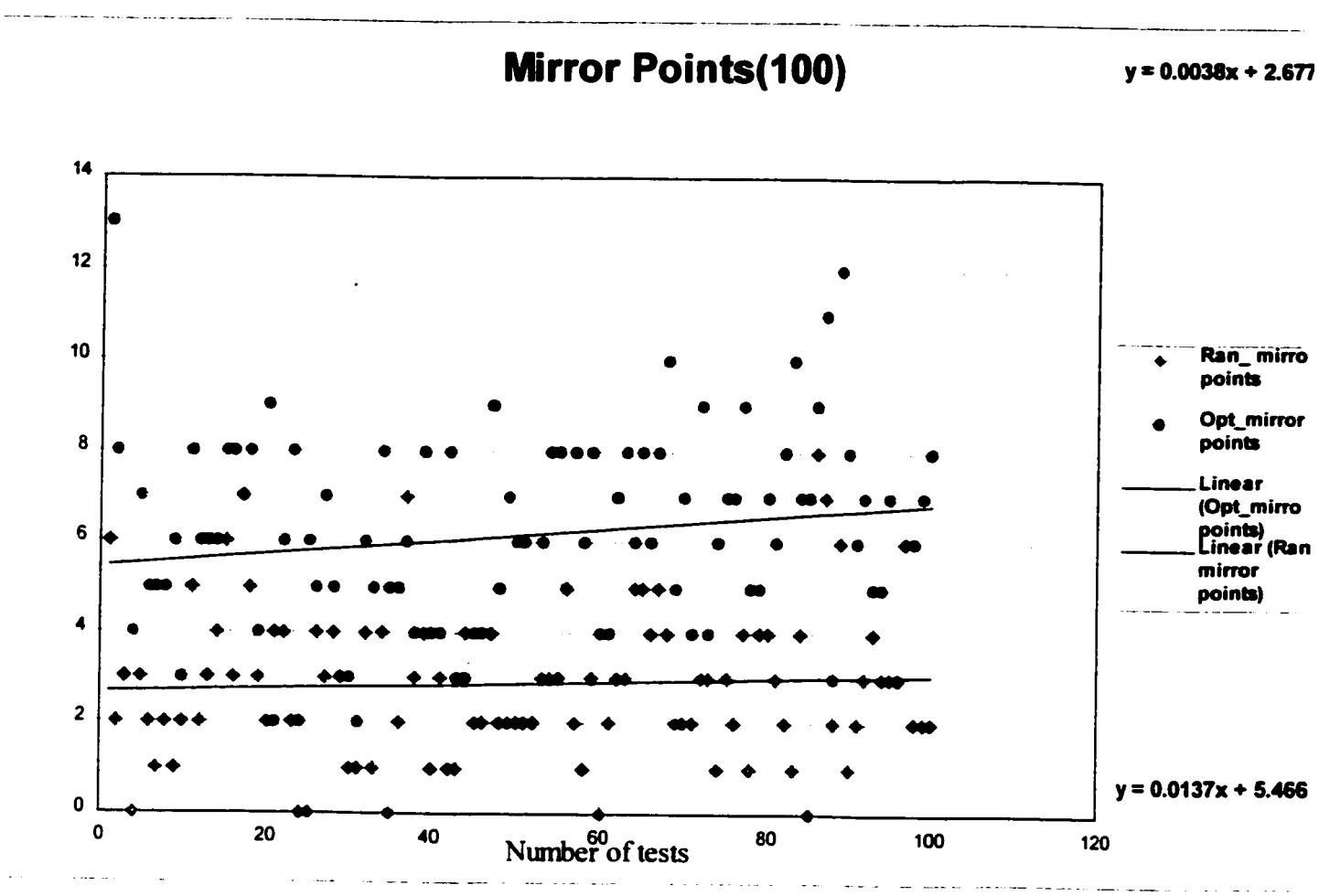


Fig. 28 mirror points in optimal sequence Vs in Random sequence with maximum travel angle restriction (Optimal degree) 50°

Table 10: The average variability (SD) of estimates of total test time, long points, mirror and repeat points in the 100-iteration optimal simulations

	Test Time	Long Points	Report Points	Mirror Points
Mean	541	1.62	4.38	6.16
SD	(±17.57)	(±1.31)	(±2.01)	(±2.16)

Table 11. Slopes for test time across iteration for each of the three 100-iteration simulation

	Test 1	Test 2	Test 3
Random	-0.1026	0.1123	0.1747
Optimal	0.1287	-0.1178	-0.1209

3. **Determining the most effective Maximum Travel Angle Restriction**

Figure 22 and Table 6 demonstrate that test time decreases continually and linearly from the minimum value of 150 degrees for the Maximum Travel Angle Restriction (1321 seconds) down to 149 seconds on average for a restriction of 10 degrees. However the values for the indicators of predictable or non-random test sequences (repeat points, mirror points see Figs 24, Tables 8 and 9) appear to increase exponentially, rising rapidly as the restriction is tightened to values smaller than 50 degrees. Inspection of these two sets of results together makes it clear that restrictions that limit travel angles to less than 50 degrees result in unacceptable increases in non-random test sequences. Thus the most effective Maximum Angle Restriction would appear to be 50 degrees, as this reduces test time the most without significantly increasing the frequency of non-random events.

4.3 Discussion

4.3.1 Test time

The results demonstrate clearly that the Optimized algorithm enhances the efficiency of the program by reducing overall test time and the number of test sequences requiring long travel times. The improvement in performance increases as the restriction a maximum travel time is tightened. For example, with the restriction of the maximum travel distance of 50°, the test time decreased to less than half that generated by the Random algorithm. When restriction of the maximum travel distance of 10°, the test time can be as short as one tenth that generated by Random algorithm. This is mainly achieved by limiting long travel time by eliminating the next test point where the travel distance exceeds maximum travel angle. In the case where these locations remain unresolved at the end of the test session, they are treated as long points and presented separately.

4.3.2 Randomness Vs Optimization

The results demonstrate increased efficiency as the restrictions on the maximum length of travel time are tightened. However such restrictions also increase the number of non-random test sequences. I can see this in the results where the test time decreased and there were fewer long points, but more repeat and mirror points in each simulation run. This means that the next test points tend to have more chance of being presented in a predicable location. For example, if a patient determines that mirror points, where the fixation point moves to the identical location in the opposite quadrant, are more likely, they can anticipate the movement, enhancing their response if they are right, but interfering with their performance if they are wrong. Similarly, if repeat points are more frequent, the patient may come to anticipate the next point, reducing uncertainly and providing an artificial performance advantage if they are correct and a performance disadvantage if they are wrong. An increase in mirror points and repeat points reduces the randomness of the test sequence and eventually result in an unreliable visual field.

However within a certain range of restrictions, my results show very good performance in test time and still maintain a low number of the repeat points and mirror points, that preserving randomness of the test sequence. Figure 22, 23, 24 shows around the 50° maximum travel angle restriction (Optimal degree), the test time is saved as half long as the total random sequence but the number of repeat points and the mirror points have not been significantly increased. This result demonstrates that it is possible to reduce test time by a factor of at least two while not significantly increasing the frequency of non-random test sequences.

Chapter 5- Summary and Conclusion

I have developed an optimal algorithm for moving the fixation arm in the Flicker Perimeter in order to achieve a more rapid test and thus a more reliable and accurate visual field assessment for early detection of glaucoma. Glaucoma is a leading cause of irreversible blindness. Once blindness from glaucoma has occurred, no treatment will restore the lost vision. However, early detection and proper treatment make the prevention of the blindness from glaucoma possible. Such assessments are carried out using a perimeter. Perimetry is a method of assessing visual field loss by testing sensitivity at various locations of visual field of the each eye of the patient. Previous research indicates the perception of flickering targets is reduced early on in the process of glaucomatous damage. To take advantage of this finding, the Flicker Perimeter and testing method has been designed. It tests the ability to detect whether or not a test target is flickering or steady, as the modulation amplitude of a sinusoidal flickering light is varied to determine patient sensitivity. Because this is a perimetric test, patient sensitivity to flickering lights is assessed in a large number of locations throughout the visual field.

The current algorithms controlling the order of presenting the test targets at different locations in the peripheral visual field is quite inefficient and makes the overall test time quite long. Long tests create patient fatigue and result in unacceptable variability of the test results. Unfortunately due to the design of the Flicker Perimeter the travel time among the test points in this prototype is significantly slower than other perimeters. Thus, reducing test time is critical to further development of this method.

To optimize the function of the prototype Flicker Perimeter, it is important to reduce test time by reducing the amount of time required to change test target location in the visual field. On the other hand it is important to maintain the apparent randomness of the test sequence to ensure accurate and reliable results. The purpose of this project is to develop and test a software algorithm that achieves these two goals simultaneously.

I have developed, implemented and tested an algorithm that is designed to produce a pseudo-random determination of test location sequence, while at the same time maximizing the efficiency of travel in the motors governing target in the fixation arm. This is achieved by using a travel angle restriction to reduce the amount of travel around the last test point, but maintain a pseudo-random sequence over a limited time frame.

I hypothesized that this approach will speed up the time between test trials as well as the entire sequence. The algorithm was developed within the contest of a software model and simulation of the Flicker Perimeter and test process. I also developed a simulation of the original random algorithm and compared the results of both methods in terms of test time; the number of long points and the number of non-random events as defined by the number of repeat and mirror points in the test sequence. I demonstrate that the simulation algorithm reflects the behavioral relationship presented in real flicker system and that it is possible to obtain a reliable estimate of the performance of the algorithm with a simulation of 100 iterations. Using this simulation procedure I compared the Optimal algorithm to the original Random algorithm and demonstrated that, as hypothesized, the Optimal algorithm does enhance the efficiency of travel time and speeds up the whole test. I also looked at the impact of the degree of restriction of travel time and found that restricting the travel time to less than 50 degrees resulted in unacceptable levels of non-random events. The number of these events increases with increased restrictions. At 50° maximum angle travel restriction, the savings in test time is substantial (half as long as the travel time generated by the Random algorithm, with out a significant increase in the number of mirror or repeat points as compared to the Random algorithm. This result indicates that it will be possible to modify the software of the Flicker Perimeter to improve the efficiency of testing, making this new test a better diagnostic tool for glaucoma. In addition the Optimal algorithm can be applied to other perimetric systems to improve their test times as well; providing a wider application of this research.

References

[1] Shields, B., Glaucoma, Williams & Wilkins, 1992

[2] Casson, E. J., Johnson, C. A., Shapiro, L. R., Longitudinal analysis of temporal modulation perimetry in early glaucoma and comparison with white-on-white and blue-on-yellow perimetry, *Inves. Ophthal. and Vis. Sci.* (1992), **33** (suppl.), 1384.

[3] Casson, E. J., Flicker Perimetry, Optimal Parameters for Detection of Glaucoma, International Perimetry Society, Würzburg, Germany, June, 1996.

[4] Casson, E. J., Damji, K. F., Zackon, D. H., Rock, W. J. Low Luminance, mid-frequency flicker perimetry in glaucoma: Is it better? Annual meeting of the Canadian Ophthalmologic Society, Quebec, June 1997.

[5] P. L'Ecuyer, Random Numbers for Simulation, Communications of the ACM 33, 10 (1990) 85-97.

[6] P. L'Ecuyer: Uniform Random Number Generation (tutorials), 1997

[7] G. Marsaglia, A Current View of Random Number Generation, Computer Science and Statistics, Proceedings of the Sixteenth Symposium on the Interface, Elsevier Science Publ. (North-Holland) (1985) 3-10

[8] F. James, A review of pseudorandom number generators, Computer Physics Communications, 60 (1990) 329-344.

[9] E. J. Dudewicz and T. G. Ralley, The Handbook of Random Number Generation and Testing with TESTRAND Computer Code, American Sciences Press, Columbus, Ohio (1981).

[10] D. E. Knuth, The Art of Computer Programming: Semi-numerical Algorithms, vol. 2,second edition. Addison-Wesley, 1981.

[11] P. L'Ecuyer, Efficient and Portable Combined Random Number Generators. Communications of the ACM 31, 6 (1988) 742-749 and 774. See also the correspondence in the same journal, 32, 8 (1989) 1019-1024.

[12] P. L'Ecuyer, Testing Random Number Generators, Proceedings of the 1992 Winter Simulation Conference, IEEE Press (1992), 305-313.

[13] Wichmann B. A. and Hill, I. D., Algorithm AS 183 Appl. Statist. (1982) vol.31 Pages 188-190, correction in vol 33, pg 123

[14] Haley, M. J., The Field Analyzer Primer, Allergan Humphrey, 1997

[15] B. P. Zeigler and H. Praehofer, Theory of Modelling and Simulation Academic Press, 2000

Appendix 1 Time Matrix

77X77 Time Matrix

---	90	7	8	11	18	19	23	31	36	45	54	59	67	71	72	79	82	83	97	98	101	108	109	113	121	126	135	144	149	157	161	162	169	172	173
90	0	-83	-82	-79	-72	-71	-67	-59	-54	-45	-36	-31	-23	-19	-18	-11	-8	-7	7	8	11	18	19	23	31	36	45	54	59	67	71	72	79	82	83
7	83	0	1	4	11	12	16	24	29	38	47	52	60	64	65	72	75	76	90	91	94	101	102	106	114	119	128	137	142	150	154	155	162	165	166
7	83	0	1	4	11	12	16	24	29	38	47	52	60	64	65	72	75	76	90	91	94	101	102	106	114	119	128	137	142	150	154	155	162	165	166
8	82	-1	0	3	10	11	15	23	28	37	46	51	59	63	64	71	74	75	89	90	93	100	101	105	113	118	127	136	141	149	153	154	161	164	165
8	82	-1	0	3	10	11	15	23	28	37	46	51	59	63	64	71	74	75	89	90	93	100	101	105	113	118	127	136	141	149	153	154	161	164	165
11	79	-4	-3	0	7	8	12	20	25	34	43	48	56	60	61	68	71	72	86	87	90	97	98	102	110	115	124	133	138	146	150	151	158	161	162
11	79	-4	-3	0	7	8	12	20	25	34	43	48	56	60	61	68	71	72	86	87	90	97	98	102	110	115	124	133	138	146	150	151	158	161	162
18	72	-11	-10	-7	0	1	5	13	18	27	36	41	49	53	54	61	64	65	79	80	83	90	91	95	103	108	117	126	131	139	143	144	151	154	155
18	72	-11	-10	-7	0	1	5	13	18	27	36	41	49	53	54	61	64	65	79	80	83	90	91	95	103	108	117	126	131	139	143	144	151	154	155
19	71	-12	-11	-8	-1	0	4	12	17	26	35	40	48	52	53	60	63	64	78	79	82	89	90	94	102	107	116	125	130	138	142	143	150	153	154
19	71	-12	-11	-8	-1	0	4	12	17	26	35	40	48	52	53	60	63	64	78	79	82	89	90	94	102	107	116	125	130	138	142	143	150	153	154
23	67	-16	-15	-12	-5	-4	0	8	13	22	31	36	44	48	49	56	59	60	74	75	78	85	86	90	98	103	112	121	126	134	138	139	146	149	150
23	67	-16	-15	-12	-5	-4	0	8	13	22	31	36	44	48	49	56	59	60	74	75	78	85	86	90	98	103	112	121	126	134	138	139	146	149	150
31	59	-24	-23	-20	-13	-12	-8	0	5	14	23	28	36	40	41	48	51	52	66	67	70	77	78	82	90	95	104	113	118	126	130	131	138	141	142
36	54	-29	-28	-25	-18	-17	-13	-5	0	9	18	23	31	35	36	43	46	47	61	62	65	72	73	77	85	90	99	108	113	121	125	126	133	136	137
36	54	-29	-28	-25	-18	-17	-13	-5	0	9	18	23	31	35	36	43	46	47	61	62	65	72	73	77	85	90	99	108	113	121	125	126	133	136	137
45	45	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38	52	53	56	63	64	68	76	81	90	99	104	112	116	117	124	127	128
45	45	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38	52	53	56	63	64	68	76	81	90	99	104	112	116	117	124	127	128
45	45	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38	52	53	56	63	64	68	76	81	90	99	104	112	116	117	124	127	128
45	45	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38	52	53	56	63	64	68	76	81	90	99	104	112	116	117	124	127	128
45	45	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38	52	53	56	63	64	68	76	81	90	99	104	112	116	117	124	127	128
54	36	-47	-46	-43	-36	-35	-31	-23	-18	-9	0	5	13	17	18	25	28	29	43	44	47	54	55	59	67	72	81	90	95	103	107	108	115	118	119
59	31	-52	-51	-48	-41	-40	-36	-28	-23	-14	-5	0	8	12	13	20	23	24	38	39	42	49	50	54	62	67	76	85	90	98	102	103	110	113	114
59	31	-52	-51	-48	-41	-40	-36	-28	-23	-14	-5	0	8	12	13	20	23	24	38	39	42	49	50	54	62	67	76	85	90	98	102	103	110	113	114
67	23	-60	-59	-56	-49	-48	-44	-36	-31	-22	-13	-8	0	4	5	12	15	16	30	31	34	41	42	46	54	59	68	77	82	90	94	95	102	105	106
67	23	-60	-59	-56	-49	-48	-44	-36	-31	-22	-13	-8	0	4	5	12	15	16	30	31	34	41	42	46	54	59	68	77	82	90	94	95	102	105	106
71	19	-64	-63	-60	-53	-52	-48	-40	-35	-26	-17	-12	-4	0	1	8	11	12	26	27	30	37	38	42	50	55	64	73	78	86	90	91	98	101	102

77X77 Time Matrix

135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
144	-54	-137	-136	-133	-126	-125	-121	-113	-108	-99	-90	-85	-77	-73	-72	-65	-62	-61	-47	-46	-43	-36	-35	-31	-23	-18	-9	0	5	13	17	18	25	28	29
144	-54	-137	-136	-133	-126	-125	-121	-113	-108	-99	-90	-85	-77	-73	-72	-65	-62	-61	-47	-46	-43	-36	-35	-31	-23	-18	-9	0	5	13	17	18	25	28	29
149	-59	-142	-141	-138	-131	-130	-126	-118	-113	-104	-95	-90	-82	-78	-77	-70	-67	-66	-52	-51	-48	-41	-40	-36	-28	-23	-14	-5	0	8	12	13	20	23	24
149	-59	-142	-141	-138	-131	-130	-126	-118	-113	-104	-95	-90	-82	-78	-77	-70	-67	-66	-52	-51	-48	-41	-40	-36	-28	-23	-14	-5	0	8	12	13	20	23	24
157	-67	-150	-149	-146	-139	-138	-134	-126	-121	-112	-103	-98	-90	-86	-85	-78	-75	-74	-60	-59	-56	-49	-48	-44	-36	-31	-22	-13	-8	0	4	5	12	15	16
157	-67	-150	-149	-146	-139	-138	-134	-126	-121	-112	-103	-98	-90	-86	-85	-78	-75	-74	-60	-59	-56	-49	-48	-44	-36	-31	-22	-13	-8	0	4	5	12	15	16
161	-71	-154	-153	-150	-143	-142	-138	-130	-125	-116	-107	-102	-94	-90	-89	-82	-79	-78	-64	-63	-60	-53	-52	-48	-40	-35	-26	-17	-12	-4	0	1	8	11	12
161	-71	-154	-153	-150	-143	-142	-138	-130	-125	-116	-107	-102	-94	-90	-89	-82	-79	-78	-64	-63	-60	-53	-52	-48	-40	-35	-26	-17	-12	-4	0	1	8	11	12
162	-72	-155	-154	-151	-144	-143	-139	-131	-126	-117	-108	-103	-95	-91	-90	-83	-80	-79	-65	-64	-61	-54	-53	-49	-41	-36	-27	-18	-13	-5	-1	0	7	10	11
162	-72	-155	-154	-151	-144	-143	-139	-131	-126	-117	-108	-103	-95	-91	-90	-83	-80	-79	-65	-64	-61	-54	-53	-49	-41	-36	-27	-18	-13	-5	-1	0	7	10	11
169	-79	-162	-161	-158	-151	-150	-146	-138	-133	-124	-115	-110	-102	-98	-97	-90	-87	-86	-72	-71	-68	-61	-60	-56	-48	-43	-34	-25	-20	-12	-8	-7	0	3	4
169	-79	-162	-161	-158	-151	-150	-146	-138	-133	-124	-115	-110	-102	-98	-97	-90	-87	-86	-72	-71	-68	-61	-60	-56	-48	-43	-34	-25	-20	-12	-8	-7	0	3	4
172	-82	-165	-164	-161	-154	-153	-149	-141	-136	-127	-118	-113	-105	-101	-100	-93	-90	-89	-75	-74	-71	-64	-63	-59	-51	-46	-37	-28	-23	-15	-11	-10	-3	0	1
172	-82	-165	-164	-161	-154	-153	-149	-141	-136	-127	-118	-113	-105	-101	-100	-93	-90	-89	-75	-74	-71	-64	-63	-59	-51	-46	-37	-28	-23	-15	-11	-10	-3	0	1
173	-83	-166	-165	-162	-155	-154	-150	-142	-137	-128	-119	-114	-106	-102	-101	-94	-91	-90	-76	-75	-72	-65	-64	-60	-52	-47	-38	-29	-24	-16	-12	-11	-4	-1	0
173	-83	-166	-165	-162	-155	-154	-150	-142	-137	-128	-119	-114	-106	-102	-101	-94	-91	-90	-76	-75	-72	-65	-64	-60	-52	-47	-38	-29	-24	-16	-12	-11	-4	-1	0

77X77 Time Matrix

135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
135	-45	-128	-127	-124	-117	-116	-112	-104	-99	-90	-81	-76	-68	-64	-63	-56	-53	-52	-38	-37	-34	-27	-26	-22	-14	-9	0	9	14	22	26	27	34	37	38
144	-54	-137	-136	-133	-126	-125	-121	-113	-108	-99	-90	-85	-77	-73	-72	-65	-62	-61	-47	-46	-43	-36	-35	-31	-23	-18	-9	0	5	13	17	18	25	28	29
144	-54	-137	-136	-133	-126	-125	-121	-113	-108	-99	-90	-85	-77	-73	-72	-65	-62	-61	-47	-46	-43	-36	-35	-31	-23	-18	-9	0	5	13	17	18	25	28	29
149	-59	-142	-141	-138	-131	-130	-126	-118	-113	-104	-95	-90	-82	-78	-77	-70	-67	-66	-52	-51	-48	-41	-40	-36	-28	-23	-14	-5	0	8	12	13	20	23	24
149	-59	-142	-141	-138	-131	-130	-126	-118	-113	-104	-95	-90	-82	-78	-77	-70	-67	-66	-52	-51	-48	-41	-40	-36	-28	-23	-14	-5	0	8	12	13	20	23	24
157	-67	-150	-149	-146	-139	-138	-134	-126	-121	-112	-103	-98	-90	-86	-85	-78	-75	-74	-60	-59	-56	-49	-48	-44	-36	-31	-22	-13	-8	0	4	5	12	15	16
157	-67	-150	-149	-146	-139	-138	-134	-126	-121	-112	-103	-98	-90	-86	-85	-78	-75	-74	-60	-59	-56	-49	-48	-44	-36	-31	-22	-13	-8	0	4	5	12	15	16
161	-71	-154	-153	-150	-143	-142	-138	-130	-125	-116	-107	-102	-94	-90	-89	-82	-79	-78	-64	-63	-60	-53	-52	-48	-40	-35	-26	-17	-12	-4	0	1	8	11	12
161	-71	-154	-153	-150	-143	-142	-138	-130	-125	-116	-107	-102	-94	-90	-89	-82	-79	-78	-64	-63	-60	-53	-52	-48	-40	-35	-26	-17	-12	-4	0	1	8	11	12
162	-72	-155	-154	-151	-144	-143	-139	-131	-126	-117	-108	-103	-95	-91	-90	-83	-80	-79	-65	-64	-61	-54	-53	-49	-41	-36	-27	-18	-13	-5	-1	0	7	10	11
162	-72	-155	-154	-151	-144	-143	-139	-131	-126	-117	-108	-103	-95	-91	-90	-83	-80	-79	-65	-64	-61	-54	-53	-49	-41	-36	-27	-18	-13	-5	-1	0	7	10	11
169	-79	-162	-161	-158	-151	-150	-146	-138	-133	-124	-115	-110	-102	-98	-97	-90	-87	-86	-72	-71	-68	-61	-60	-56	-48	-43	-34	-25	-20	-12	-8	-7	0	3	4
169	-79	-162	-161	-158	-151	-150	-146	-138	-133	-124	-115	-110	-102	-98	-97	-90	-87	-86	-72	-71	-68	-61	-60	-56	-48	-43	-34	-25	-20	-12	-8	-7	0	3	4
172	-82	-165	-164	-161	-154	-153	-149	-141	-136	-127	-118	-113	-105	-101	-100	-93	-90	-89	-75	-74	-71	-64	-63	-59	-51	-46	-37	-28	-23	-15	-11	-10	-3	0	1
172	-82	-165	-164	-161	-154	-153	-149	-141	-136	-127	-118	-113	-105	-101	-100	-93	-90	-89	-75	-74	-71	-64	-63	-59	-51	-46	-37	-28	-23	-15	-11	-10	-3	0	1
173	-83	-166	-165	-162	-155	-154	-150	-142	-137	-128	-119	-114	-106	-102	-101	-94	-91	-90	-76	-75	-72	-65	-64	-60	-52	-47	-38	-29	-24	-16	-12	-11	-4	-1	0
173	-83	-166	-165	-162	-155	-154	-150	-142	-137	-128	-119	-114	-106	-102	-101	-94	-91	-90	-76	-75	-72	-65	-64	-60	-52	-47	-38	-29	-24	-16	-12	-11	-4	-1	0

Appendix 2 100 - Iteration Simulations Output

50	3x76	10	1365	124	211	2	2	3	15	25
51	3x76	10	1365	124	211	2	2	3	15	25
52	3x76	10	1445	144	208	3	2	6	23	30
53	3x76	10	1327	178	200	3	0	9	13	22
54	3x76	10	1327	178	200	3	0	9	13	22
55	3x76	10	1392	146	208	3	3	10	15	30
56	3x76	10	1460	149	209	1	2	8	19	31
57	3x76	10	1460	149	209	1	2	8	19	31
58	3x76	10	1327	121	210	3	3	7	19	30
59	3x76	10	1252	149	201	5	4	6	13	32
60	3x76	10	1336	151	211	3	3	10	19	29
61	3x76	10	1336	151	211	3	3	10	19	29
62	3x76	10	1355	145	215	1	2	6	18	36
63	3x76	10	1389	158	197	0	4	12	15	30
64	3x76	10	1423	167	209	2	2	5	18	33
65	3x76	10	1423	167	209	2	2	5	18	33
66	3x76	10	1419	145	191	1	9	12	16	33
67	3x76	10	1376	140	190	0	2	9	16	20
68	3x76	10	1376	140	190	0	2	9	16	20
69	3x76	10	1419	146	197	0	7	10	21	29
70	3x76	10	1194	146	196	3	4	11	16	25
71	3x76	10	1194	146	196	3	4	11	16	25
72	3x76	10	1361	132	213	5	2	4	14	33
73	3x76	10	1328	165	209	4	4	12	26	30
74	3x76	10	1328	165	209	4	4	12	26	30
75	3x76	10	1319	156	200	3	3	4	24	20
76	3x76	10	1286	159	188	0	6	9	20	33
77	3x76	10	1286	159	188	0	6	9	20	33
78	3x76	10	1465	160	213	1	3	8	18	20
79	3x76	10	1231	164	192	2	3	11	16	27
80	3x76	10	1304	163	194	3	3	10	14	26
81	3x76	10	1304	163	194	3	3	10	14	26
82	3x76	10	1433	152	208	3	2	12	18	26
83	3x76	10	1439	130	203	4	2	7	20	23
84	3x76	10	1395	173	205	1	4	11	16	17
85	3x76	10	1395	173	205	1	4	11	16	17
86	3x76	10	1395	175	196	3	1	3	14	26
87	3x76	10	1246	161	197	4	4	13	20	24
88	3x76	10	1246	161	197	4	4	13	20	24
89	3x76	10	1465	161	205	2	3	9	17	27
90	3x76	10	1324	149	193	3	2	1	22	29
91	3x76	10	1433	154	198	1	4	8	9	29
92	3x76	10	1433	154	198	1	4	8	9	29
93	3x76	10	1365	153	199	2	4	5	22	29
94	3x76	10	1355	145	208	1	5	9	19	28
95	3x76	10	1467	141	191	1	2	19	19	28
96	3x76	10	1371	134	215	0	0	7	21	29
97	3x76	10	1371	134	215	0	0	7	21	29
98	3x76	10	1297	144	199	3	4	6	26	22
99	3x76	10	1326	182	204	1	1	9	15	26
100	3x76	10	1326	182	204	1	1	9	15	26

100 – iteration simulations with 10 degree

Test	Size	O_degree	R-time	O-time	R_Long	R_Repeat	R_Mirror	O_Long	O_Repeat	O_Mirror
1	3x76	10	1304	127	185	3	3	2	23	29
2	3x76	10	1419	130	211	0	5	12	13	27
3	3x76	10	1391	151	195	3	4	10	14	28
4	3x76	10	1391	151	195	3	4	10	14	28
5	3x76	10	1358	139	197	1	1	3	11	30
6	3x76	10	1400	125	215	1	0	10	21	29
7	3x76	10	1278	150	215	3	2	11	16	27
8	3x76	10	1278	150	215	3	2	11	16	27
9	3x76	10	1382	137	211	1	4	4	20	27
10	3x76	10	1347	151	188	3	3	7	21	25
11	3x76	10	1347	151	188	3	3	7	21	25
12	3x76	10	1340	141	194	0	3	10	11	30
13	3x76	10	1426	129	203	0	3	7	12	31
14	3x76	10	1426	129	203	0	3	7	12	31
15	3x76	10	1383	147	215	4	2	8	16	25
16	3x76	10	1332	176	194	2	4	19	29	15
17	3x76	10	1332	176	194	2	4	19	29	15
18	3x76	10	1327	171	194	5	6	12	22	26
19	3x76	10	1386	159	208	2	1	4	16	18
20	3x76	10	1330	145	201	2	2	4	24	23
21	3x76	10	1330	145	201	2	2	4	24	23
22	3x76	10	1335	150	210	2	3	9	27	30
23	3x76	10	1316	164	217	2	2	4	22	18
24	3x76	10	1348	133	213	2	3	6	14	17
25	3x76	10	1348	133	213	2	3	6	14	17
26	3x76	10	1242	141	198	1	6	6	16	38
27	3x76	10	1465	164	219	5	0	5	25	21
28	3x76	10	1465	164	219	5	0	5	25	21
29	3x76	10	1420	157	207	1	3	7	14	24
30	3x76	10	1330	140	202	1	2	9	22	32
31	3x76	10	1330	140	202	1	2	9	22	32
32	3x76	10	1412	141	211	1	2	4	31	21
33	3x76	10	1386	149	198	2	4	11	20	31
34	3x76	10	1390	152	192	1	5	12	19	28
35	3x76	10	1390	152	192	1	5	12	19	28
36	3x76	10	1365	148	191	3	4	5	16	31
37	3x76	10	1388	144	215	5	1	12	17	23
38	3x76	10	1388	144	215	5	1	12	17	23
39	3x76	10	1335	154	212	3	0	7	22	24
40	3x76	10	1385	135	215	1	0	4	16	27
41	3x76	10	1385	135	215	1	0	4	16	27
42	3x76	10	1493	163	210	0	2	8	15	22
43	3x76	10	1329	125	198	4	5	12	16	35
44	3x76	10	1329	125	198	4	5	12	16	35
45	3x76	10	1333	126	200	2	2	5	12	31
46	3x76	10	1339	142	180	1	2	9	21	17
47	3x76	10	1205	130	190	5	5	8	21	26
48	3x76	10	1205	130	190	5	5	8	21	26
49	3x76	10	1353	147	207	1	5	6	15	36

100 – iteration simulations with 30 degree

Test	Size	O_degree	R-time	O-time	R_Long	R_Repeat	R_Mirror	O_Long	O_Repeat	O_Mirror
1	3x76	30	1356	346	159	3	2	3	7	11
2	3x76	30	1315	361	153	0	1	3	5	19
3	3x76	30	1230	337	148	3	5	2	7	7
4	3x76	30	1359	354	140	1	2	1	9	8
5	3x76	30	1287	361	143	1	3	1	3	11
6	3x76	30	1391	365	161	4	4	3	3	11
7	3x76	30	1381	348	160	0	4	3	7	9
8	3x76	30	1365	349	173	3	3	2	8	12
9	3x76	30	1453	357	167	3	0	6	5	5
10	3x76	30	1296	340	160	1	4	8	5	13
11	3x76	30	1360	356	150	3	3	5	6	11
12	3x76	30	1415	395	173	2	4	11	5	13
13	3x76	30	1365	362	156	2	3	2	5	7
14	3x76	30	1259	344	125	1	4	3	4	10
15	3x76	30	1358	378	147	2	5	2	4	9
16	3x76	30	1330	389	140	3	2	2	7	6
17	3x76	30	1288	360	141	5	6	2	5	8
18	3x76	30	1274	373	154	2	1	7	3	6
19	3x76	30	1293	338	153	3	5	0	11	9
20	3x76	30	1300	373	162	1	4	3	5	11
21	3x76	30	1359	356	149	0	3	2	8	8
22	3x76	30	1345	355	154	2	4	1	9	3
23	3x76	30	1313	344	146	6	4	5	8	6
24	3x76	30	1295	367	147	3	7	0	6	14
25	3x76	30	1279	359	136	0	2	0	3	6
26	3x76	30	1392	346	145	0	5	5	3	11
27	3x76	30	1416	365	168	1	5	5	3	11
28	3x76	30	1338	370	158	0	3	4	6	12
29	3x76	30	1325	364	161	1	1	2	4	11
30	3x76	30	1305	371	161	0	1	1	5	7
31	3x76	30	1359	343	153	2	5	1	9	11
32	3x76	30	1265	382	140	1	4	0	3	12
33	3x76	30	1480	344	186	1	2	6	7	9
34	3x76	30	1335	363	152	2	4	2	2	5
35	3x76	30	1484	365	176	2	3	0	6	7
36	3x76	30	1392	359	163	2	0	6	5	11
37	3x76	30	1417	343	166	2	2	2	8	12
38	3x76	30	1358	389	183	5	2	3	7	5
39	3x76	30	1425	378	155	1	2	3	7	9
40	3x76	30	1366	394	169	2	2	4	3	7
41	3x76	30	1415	363	175	3	2	5	5	8
42	3x76	30	1437	350	164	1	2	1	7	8
43	3x76	30	1239	367	150	5	3	3	7	11
44	3x76	30	1317	359	163	4	2	1	12	6
45	3x76	30	1357	349	159	1	1	1	5	11
46	3x76	30	1412	380	158	1	0	1	4	7
47	3x76	30	1353	355	154	1	2	6	6	13
48	3x76	30	1432	379	161	0	3	1	3	11
49	3x76	30	1352	395	145	2	2	3	3	10

100 – iteration simulations with 30 degree

Test	Size	O_degree	R-time	O-time	R_Long	R_Repeat	R_Mirror	O_Long	O_Repeat	O_Mirror
1	3x76	30	1356	346	159	3	2	3	7	11
2	3x76	30	1315	361	153	0	1	3	5	19
3	3x76	30	1230	337	148	3	5	2	7	7
4	3x76	30	1359	354	140	1	2	1	9	8
5	3x76	30	1287	361	143	1	3	1	3	11
6	3x76	30	1391	365	161	4	4	3	3	11
7	3x76	30	1381	348	160	0	4	3	7	9
8	3x76	30	1365	349	173	3	3	2	8	12
9	3x76	30	1453	357	167	3	0	6	5	5
10	3x76	30	1296	340	160	1	4	8	5	13
11	3x76	30	1360	356	150	3	3	5	6	11
12	3x76	30	1415	395	173	2	4	11	5	13
13	3x76	30	1365	362	156	2	3	2	5	7
14	3x76	30	1259	344	125	1	4	3	4	10
15	3x76	30	1358	378	147	2	5	2	4	9
16	3x76	30	1330	389	140	3	2	2	7	6
17	3x76	30	1288	360	141	5	6	2	5	8
18	3x76	30	1274	373	154	2	1	7	3	6
19	3x76	30	1293	338	153	3	5	0	11	9
20	3x76	30	1300	373	162	1	4	3	5	11
21	3x76	30	1359	356	149	0	3	2	8	8
22	3x76	30	1345	355	154	2	4	1	9	3
23	3x76	30	1313	344	146	6	4	5	8	6
24	3x76	30	1295	367	147	3	7	0	6	14
25	3x76	30	1279	359	136	0	2	0	3	6
26	3x76	30	1392	346	145	0	5	5	3	11
27	3x76	30	1416	365	168	1	5	5	3	11
28	3x76	30	1338	370	158	0	3	4	6	12
29	3x76	30	1325	364	161	1	1	2	4	11
30	3x76	30	1305	371	161	0	1	1	5	7
31	3x76	30	1359	343	153	2	5	1	9	11
32	3x76	30	1265	382	140	1	4	0	3	12
33	3x76	30	1480	344	186	1	2	6	7	9
34	3x76	30	1335	363	152	2	4	2	2	5
35	3x76	30	1484	365	176	2	3	0	6	7
36	3x76	30	1392	359	163	2	0	6	5	11
37	3x76	30	1417	343	166	2	2	2	8	12
38	3x76	30	1358	389	183	5	2	3	7	5
39	3x76	30	1425	378	155	1	2	3	7	9
40	3x76	30	1366	394	169	2	2	4	3	7
41	3x76	30	1415	363	175	3	2	5	5	8
42	3x76	30	1437	350	164	1	2	1	7	8
43	3x76	30	1239	367	150	5	3	3	7	11
44	3x76	30	1317	359	163	4	2	1	12	6
45	3x76	30	1357	349	159	1	1	1	5	11
46	3x76	30	1412	380	158	1	0	1	4	7
47	3x76	30	1353	355	154	1	2	6	6	13
48	3x76	30	1432	379	161	0	3	1	3	11
49	3x76	30	1352	395	145	2	2	3	3	10

100 – iteration simulations with 50 degree

Test	Size	O_degre	R-time	O-time	R_Long	R_Repea	R_Mirror	O_Long	O_Repea	O_Mirror
1	3x76	50	1413	540	120	3	3	3	5	5
2	3x76	50	1369	564	116	2	0	2	6	2
3	3x76	50	1349	532	131	5	2	0	9	4
4	3x76	50	1388	567	117	0	1	2	2	5
5	3x76	50	1357	560	133	0	2	0	2	5
6	3x76	50	1370	556	124	3	2	6	8	6
7	3x76	50	1437	534	122	2	2	0	0	8
8	3x76	50	1379	544	106	0	4	2	2	8
9	3x76	50	1287	511	108	4	9	0	8	10
10	3x76	50	1206	509	102	4	5	3	4	7
11	3x76	50	1320	566	122	0	1	3	6	0
12	3x76	50	1375	561	124	0	3	4	3	5
13	3x76	50	1366	545	109	4	1	2	3	5
14	3x76	50	1463	578	146	1	0	0	2	1
15	3x76	50	1371	542	149	3	6	2	2	12
16	3x76	50	1264	551	91	5	1	7	8	9
17	3x76	50	1443	571	123	1	2	1	3	3
18	3x76	50	1233	533	108	2	2	2	5	5
19	3x76	50	1314	520	103	2	3	9	9	5
20	3x76	50	1295	547	110	1	3	2	2	10
21	3x76	50	1395	559	112	1	1	1	1	6
22	3x76	50	1400	537	112	3	5	1	3	8
23	3x76	50	1375	524	119	5	3	5	6	4
24	3x76	50	1355	538	117	4	2	2	3	12
25	3x76	50	1413	563	128	2	1	6	5	4
26	3x76	50	1357	560	124	2	1	3	3	4
27	3x76	50	1307	540	99	1	2	2	4	6
28	3x76	50	1263	555	120	2	0	2	3	6
29	3x76	50	1352	512	96	2	4	0	4	7
30	3x76	50	1402	581	121	2	1	1	5	4
31	3x76	50	1416	537	129	1	1	4	2	7
32	3x76	50	1400	526	119	0	2	1	0	7
33	3x76	50	1341	552	97	1	4	2	1	6
34	3x76	50	1419	516	119	2	1	0	4	12
35	3x76	50	1328	560	99	4	0	1	4	1
36	3x76	50	1325	551	131	4	1	0	6	4
37	3x76	50	1437	552	125	0	0	0	4	6
38	3x76	50	1496	538	132	4	0	0	6	9
39	3x76	50	1363	572	111	0	4	1	5	8
40	3x76	50	1351	545	117	1	3	3	5	3
41	3x76	50	1297	535	108	1	2	2	3	5
42	3x76	50	1414	561	113	0	4	2	3	5
43	3x76	50	1257	552	114	1	3	3	3	4
44	3x76	50	1341	532	97	2	3	0	5	5
45	3x76	50	1407	551	139	3	3	4	5	7
46	3x76	50	1355	563	127	1	4	3	3	7
47	3x76	50	1325	542	99	2	5	1	2	5
48	3x76	50	1423	498	122	2	6	2	5	10
49	3x76	50	1262	535	127	2	3	3	4	8
50	3x76	50	1281	542	123	2	2	4	6	3
51	3x76	50	1372	528	106	2	3	2	2	11
52	3x76	50	1267	553	110	4	4	1	4	8

100 – iteration simulations with 50 degree

Test	Size	O_degre	R-time	O-time	R_Long	R_Repea	R_Mirror	O_Long	O_Repea	O_Mirror
1	3x76	50	1413	540	120	3	3	3	5	5
2	3x76	50	1369	564	116	2	0	2	6	2
3	3x76	50	1349	532	131	5	2	0	9	4
4	3x76	50	1388	567	117	0	1	2	2	5
5	3x76	50	1357	560	133	0	2	0	2	5
6	3x76	50	1370	556	124	3	2	6	8	6
7	3x76	50	1437	534	122	2	2	0	0	8
8	3x76	50	1379	544	106	0	4	2	2	8
9	3x76	50	1287	511	108	4	9	0	8	10
10	3x76	50	1206	509	102	4	5	3	4	7
11	3x76	50	1320	566	122	0	1	3	6	0
12	3x76	50	1375	561	124	0	3	4	3	5
13	3x76	50	1366	545	109	4	1	2	3	5
14	3x76	50	1463	578	146	1	0	0	2	1
15	3x76	50	1371	542	149	3	6	2	2	12
16	3x76	50	1264	551	91	5	1	7	8	9
17	3x76	50	1443	571	123	1	2	1	3	3
18	3x76	50	1233	533	108	2	2	2	5	5
19	3x76	50	1314	520	103	2	3	9	9	5
20	3x76	50	1295	547	110	1	3	2	2	10
21	3x76	50	1395	559	112	1	1	1	1	6
22	3x76	50	1400	537	112	3	5	1	3	8
23	3x76	50	1375	524	119	5	3	5	6	4
24	3x76	50	1355	538	117	4	2	2	3	12
25	3x76	50	1413	563	128	2	1	6	5	4
26	3x76	50	1357	560	124	2	1	3	3	4
27	3x76	50	1307	540	99	1	2	2	4	6
28	3x76	50	1263	555	120	2	0	2	3	6
29	3x76	50	1352	512	96	2	4	0	4	7
30	3x76	50	1402	581	121	2	1	1	5	4
31	3x76	50	1416	537	129	1	1	4	2	7
32	3x76	50	1400	526	119	0	2	1	0	7
33	3x76	50	1341	552	97	1	4	2	1	6
34	3x76	50	1419	516	119	2	1	0	4	12
35	3x76	50	1328	560	99	4	0	1	4	1
36	3x76	50	1325	551	131	4	1	0	6	4
37	3x76	50	1437	552	125	0	0	0	4	6
38	3x76	50	1496	538	132	4	0	0	6	9
39	3x76	50	1363	572	111	0	4	1	5	8
40	3x76	50	1351	545	117	1	3	3	5	3
41	3x76	50	1297	535	108	1	2	2	3	5
42	3x76	50	1414	561	113	0	4	2	3	5
43	3x76	50	1257	552	114	1	3	3	3	4
44	3x76	50	1341	532	97	2	3	0	5	5
45	3x76	50	1407	551	139	3	3	4	5	7
46	3x76	50	1355	563	127	1	4	3	3	7
47	3x76	50	1325	542	99	2	5	1	2	5
48	3x76	50	1423	498	122	2	6	2	5	10
49	3x76	50	1262	535	127	2	3	3	4	8
50	3x76	50	1281	542	123	2	2	4	6	3
51	3x76	50	1372	528	106	2	3	2	2	11
52	3x76	50	1267	553	110	4	4	1	4	8

100 – iteration simulations with 90 degree

Test	Size	O_degree	R-time	O-time	R_Long	R_Repeat	R_Mirror	O_Long	O_Repeat	O_Mirror
1	3x76	90	1294	946	43	2	6	0	4	7
2	3x76	90	1329	994	52	2	3	1	2	4
3	3x76	90	1348	887	61	3	3	1	3	3
4	3x76	90	1337	915	35	1	2	1	2	1
5	3x76	90	1364	915	53	3	1	2	5	1
6	3x76	90	1457	896	83	2	2	0	0	3
7	3x76	90	1394	968	65	2	0	0	2	1
8	3x76	90	1357	970	58	0	2	2	1	3
9	3x76	90	1380	938	39	4	2	0	1	5
10	3x76	90	1348	920	43	3	0	2	3	0
11	3x76	90	1376	881	55	0	4	0	3	8
12	3x76	90	1421	915	63	2	5	2	2	6
13	3x76	90	1326	929	54	7	3	2	7	4
14	3x76	90	1298	933	58	3	6	6	3	6
15	3x76	90	1475	1006	70	1	1	0	2	4
16	3x76	90	1396	952	59	2	3	1	2	5
17	3x76	90	1350	962	51	0	6	0	0	7
18	3x76	90	1332	963	47	0	4	0	1	4
19	3x76	90	1389	885	55	3	4	0	3	7
20	3x76	90	1321	917	48	2	3	0	3	3
21	3x76	90	1379	998	44	0	5	2	5	5
22	3x76	90	1345	909	55	3	1	1	3	3
23	3x76	90	1339	847	69	0	4	0	0	6
24	3x76	90	1360	966	50	1	3	0	2	4
25	3x76	90	1403	990	51	1	2	1	2	2
26	3x76	90	1272	963	21	3	2	1	2	3
27	3x76	90	1282	880	47	3	3	0	4	4
28	3x76	90	1342	903	66	4	4	3	2	6
29	3x76	90	1308	910	55	4	4	7	3	2
30	3x76	90	1418	966	65	6	4	1	6	5
31	3x76	90	1320	880	54	2	4	1	0	5
32	3x76	90	1317	926	43	4	4	0	4	5
33	3x76	90	1360	924	47	0	3	0	0	7
34	3x76	90	1455	991	54	2	1	3	4	3
35	3x76	90	1300	884	51	4	5	0	3	8
36	3x76	90	1432	967	51	0	4	1	0	6
37	3x76	90	1339	901	57	2	1	0	5	1
38	3x76	90	1391	968	59	3	3	1	4	3
39	3x76	90	1299	934	54	1	8	2	3	5
40	3x76	90	1309	946	50	3	4	1	3	2
41	3x76	90	1322	906	73	4	4	4	4	3
42	3x76	90	1406	947	69	3	2	1	4	3
43	3x76	90	1388	933	43	4	3	0	3	3
44	3x76	90	1373	1040	62	2	4	0	2	4
45	3x76	90	1347	906	63	4	5	1	5	6
46	3x76	90	1381	943	44	3	2	3	4	3
47	3x76	90	1399	922	51	2	4	2	4	3
48	3x76	90	1364	1036	55	2	3	2	2	3
49	3x76	90	1360	901	55	0	0	0	1	1
50	3x76	90	1365	888	66	2	4	3	4	2
51	3x76	90	1373	959	43	0	4	2	3	5

100 – iteration simulations with 90 degree

Test	Size	O_degree	R-time	O-time	R_Long	R_Repeat	R_Mirror	O_Long	O_Repeat	O_Mirror
1	3x76	90	1294	946	43	2	6	0	4	7
2	3x76	90	1329	994	52	2	3	1	2	4
3	3x76	90	1348	887	61	3	3	1	3	3
4	3x76	90	1337	915	35	1	2	1	2	1
5	3x76	90	1364	915	53	3	1	2	5	1
6	3x76	90	1457	896	83	2	2	0	0	3
7	3x76	90	1394	968	65	2	0	0	2	1
8	3x76	90	1357	970	58	0	2	2	1	3
9	3x76	90	1380	938	39	4	2	0	1	5
10	3x76	90	1348	920	43	3	0	2	3	0
11	3x76	90	1376	881	55	0	4	0	3	8
12	3x76	90	1421	915	63	2	5	2	2	6
13	3x76	90	1326	929	54	7	3	2	7	4
14	3x76	90	1298	933	58	3	6	6	3	6
15	3x76	90	1475	1006	70	1	1	0	2	4
16	3x76	90	1396	952	59	2	3	1	2	5
17	3x76	90	1350	962	51	0	6	0	0	7
18	3x76	90	1332	963	47	0	4	0	1	4
19	3x76	90	1389	885	55	3	4	0	3	7
20	3x76	90	1321	917	48	2	3	0	3	3
21	3x76	90	1379	998	44	0	5	2	5	5
22	3x76	90	1345	909	55	3	1	1	3	3
23	3x76	90	1339	847	69	0	4	0	0	6
24	3x76	90	1360	966	50	1	3	0	2	4
25	3x76	90	1403	990	51	1	2	1	2	2
26	3x76	90	1272	963	21	3	2	1	2	3
27	3x76	90	1282	880	47	3	3	0	4	4
28	3x76	90	1342	903	66	4	4	3	2	6
29	3x76	90	1308	910	55	4	4	7	3	2
30	3x76	90	1418	966	65	6	4	1	6	5
31	3x76	90	1320	880	54	2	4	1	0	5
32	3x76	90	1317	926	43	4	4	0	4	5
33	3x76	90	1360	924	47	0	3	0	0	7
34	3x76	90	1455	991	54	2	1	3	4	3
35	3x76	90	1300	884	51	4	5	0	3	8
36	3x76	90	1432	967	51	0	4	1	0	6
37	3x76	90	1339	901	57	2	1	0	5	1
38	3x76	90	1391	968	59	3	3	1	4	3
39	3x76	90	1299	934	54	1	8	2	3	5
40	3x76	90	1309	946	50	3	4	1	3	2
41	3x76	90	1322	906	73	4	4	4	4	3
42	3x76	90	1406	947	69	3	2	1	4	3
43	3x76	90	1388	933	43	4	3	0	3	3
44	3x76	90	1373	1040	62	2	4	0	2	4
45	3x76	90	1347	906	63	4	5	1	5	6
46	3x76	90	1381	943	44	3	2	3	4	3
47	3x76	90	1399	922	51	2	4	2	4	3
48	3x76	90	1364	1036	55	2	3	2	2	3
49	3x76	90	1360	901	55	0	0	0	1	1
50	3x76	90	1365	888	66	2	4	3	4	2
51	3x76	90	1373	959	43	0	4	2	3	5

100- iteration simulation with 150 degree

Test	Size	O_degree	R-time	O-time	R_Long	R_Repeat	R_Mirror	O_Long	O_Repeat	O_Mirror
1	3x76	150	1376	1275	13	2	3	2	2	3
2	3x76	150	1342	1327	6	2	2	0	2	3
3	3x76	150	1335	1279	9	3	5	0	3	5
4	3x76	150	1380	1305	7	3	2	0	2	2
5	3x76	150	1476	1422	8	1	0	4	1	1
6	3x76	150	1284	1254	1	2	0	1	2	0
7	3x76	150	1285	1272	6	3	4	2	3	3
8	3x76	150	1421	1385	6	2	3	0	2	3
9	3x76	150	1247	1219	4	3	5	1	3	5
10	3x76	150	1322	1236	4	3	4	0	4	4
11	3x76	150	1376	1273	11	3	5	2	3	5
12	3x76	150	1238	1218	0	2	7	0	2	7
13	3x76	150	1307	1262	12	4	2	0	4	1
14	3x76	150	1350	1291	13	4	2	8	5	2
15	3x76	150	1459	1453	1	3	1	0	3	1
16	3x76	150	1318	1311	5	3	2	3	3	2
17	3x76	150	1408	1388	4	3	5	0	3	4
18	3x76	150	1324	1277	6	1	3	2	1	3
19	3x76	150	1384	1336	13	3	5	0	3	5
20	3x76	150	1333	1269	2	3	3	0	3	3
21	3x76	150	1427	1406	13	3	1	3	3	1
22	3x76	150	1371	1341	5	2	3	0	2	3
23	3x76	150	1363	1347	6	1	1	0	1	1
24	3x76	150	1509	1456	9	1	1	0	1	1
25	3x76	150	1320	1280	8	1	6	3	1	5
26	3x76	150	1414	1355	12	3	4	1	3	4
27	3x76	150	1350	1340	12	2	3	4	2	3
28	3x76	150	1367	1324	15	4	2	1	4	2
29	3x76	150	1426	1411	7	0	1	0	0	1
30	3x76	150	1424	1360	9	3	0	0	4	0
31	3x76	150	1376	1363	14	1	5	5	1	5
32	3x76	150	1319	1277	6	1	3	0	1	3
33	3x76	150	1408	1337	11	1	1	0	1	2
34	3x76	150	1406	1342	7	0	1	0	0	1
35	3x76	150	1347	1323	7	2	6	0	2	6
36	3x76	150	1378	1332	2	2	2	0	2	2
37	3x76	150	1448	1342	12	1	1	3	1	1
38	3x76	150	1341	1315	2	1	5	2	1	5
39	3x76	150	1436	1402	10	0	3	2	0	3
40	3x76	150	1378	1325	7	1	5	1	1	5
41	3x76	150	1395	1371	3	2	5	0	2	5
42	3x76	150	1333	1276	3	3	7	1	4	7
43	3x76	150	1450	1389	10	1	1	1	1	1
44	3x76	150	1368	1320	13	3	7	0	4	8
45	3x76	150	1408	1357	6	2	0	3	2	0
46	3x76	150	1314	1300	3	3	0	0	3	0
47	3x76	150	1369	1348	7	0	4	0	0	4
48	3x76	150	1385	1330	12	5	1	2	5	2
49	3x76	150	1332	1313	15	5	3	0	5	3

100- iteration simulation with 150 degree

Test	Size	O_degree	R-time	O-time	R_Long	R_Repeat	R_Mirror	O_Long	O_Repeat	O_Mirror
1	3x76	150	1376	1275	13	2	3	2	2	3
2	3x76	150	1342	1327	6	2	2	0	2	3
3	3x76	150	1335	1279	9	3	5	0	3	5
4	3x76	150	1380	1305	7	3	2	0	2	2
5	3x76	150	1476	1422	8	1	0	4	1	1
6	3x76	150	1284	1254	1	2	0	1	2	0
7	3x76	150	1285	1272	6	3	4	2	3	3
8	3x76	150	1421	1385	6	2	3	0	2	3
9	3x76	150	1247	1219	4	3	5	1	3	5
10	3x76	150	1322	1236	4	3	4	0	4	4
11	3x76	150	1376	1273	11	3	5	2	3	5
12	3x76	150	1238	1218	0	2	7	0	2	7
13	3x76	150	1307	1262	12	4	2	0	4	1
14	3x76	150	1350	1291	13	4	2	8	5	2
15	3x76	150	1459	1453	1	3	1	0	3	1
16	3x76	150	1318	1311	5	3	2	3	3	2
17	3x76	150	1408	1388	4	3	5	0	3	4
18	3x76	150	1324	1277	6	1	3	2	1	3
19	3x76	150	1384	1336	13	3	5	0	3	5
20	3x76	150	1333	1269	2	3	3	0	3	3
21	3x76	150	1427	1406	13	3	1	3	3	1
22	3x76	150	1371	1341	5	2	3	0	2	3
23	3x76	150	1363	1347	6	1	1	0	1	1
24	3x76	150	1509	1456	9	1	1	0	1	1
25	3x76	150	1320	1280	8	1	6	3	1	5
26	3x76	150	1414	1355	12	3	4	1	3	4
27	3x76	150	1350	1340	12	2	3	4	2	3
28	3x76	150	1367	1324	15	4	2	1	4	2
29	3x76	150	1426	1411	7	0	1	0	0	1
30	3x76	150	1424	1360	9	3	0	0	4	0
31	3x76	150	1376	1363	14	1	5	5	1	5
32	3x76	150	1319	1277	6	1	3	0	1	3
33	3x76	150	1408	1337	11	1	1	0	1	2
34	3x76	150	1406	1342	7	0	1	0	0	1
35	3x76	150	1347	1323	7	2	6	0	2	6
36	3x76	150	1378	1332	2	2	2	0	2	2
37	3x76	150	1448	1342	12	1	1	3	1	1
38	3x76	150	1341	1315	2	1	5	2	1	5
39	3x76	150	1436	1402	10	0	3	2	0	3
40	3x76	150	1378	1325	7	1	5	1	1	5
41	3x76	150	1395	1371	3	2	5	0	2	5
42	3x76	150	1333	1276	3	3	7	1	4	7
43	3x76	150	1450	1389	10	1	1	1	1	1
44	3x76	150	1368	1320	13	3	7	0	4	8
45	3x76	150	1408	1357	6	2	0	3	2	0
46	3x76	150	1314	1300	3	3	0	0	3	0
47	3x76	150	1369	1348	7	0	4	0	0	4
48	3x76	150	1385	1330	12	5	1	2	5	2
49	3x76	150	1332	1313	15	5	3	0	5	3

Appendix 3: Wichmann Hill random number generator

```
program ...
var
   x, y, z: integer; ( global seeds )
   ...
function random: real;
   var
      temp: real;
   begin
   ( first generator )
   x := 171 * (x mod 177) - 2 * (x div 177);
   if x < 0 then
      x := x + 30269;
   ( second generator )
   y := 172 * (y mod 176) - 35* (y div 176);
   if y < 0 then
      y := y + 30307;
   ( third generator )
   z := 170 * (z mod 178) -63* (z div 178);
   if z < 0 then
      z := z + 30323;
   ( combine to give function value )
   temp := x/30269.0 + y/30307.0 + z/30323.0;
   random := temp - trunc(temp)
   end;
   ...
begin
( initialize seeds.  For production runs, different values (between 1
and 30000) should be used each time, preferably by some automatic
method such as from date and time readings if available )
x := 1; y := 10000; z := 3000;
...
end
```

(following code generate integer between 1 to76, contribution by Wu, Jie)

```
double random;

random = random*100000/76 + 1;
```