



5-2007

Implementing Quantum Random Walks in Two-Dimensions with Application to Diffusion-Limited Aggregation

Colin Frederick Sanberg
Butler University

Follow this and additional works at: <https://digitalcommons.butler.edu/ugtheses>



Part of the [Physics Commons](#)

Recommended Citation

Sanberg, Colin Frederick, "Implementing Quantum Random Walks in Two-Dimensions with Application to Diffusion-Limited Aggregation" (2007). *Undergraduate Honors Thesis Collection*. 14.
<https://digitalcommons.butler.edu/ugtheses/14>

This Thesis is brought to you for free and open access by the Undergraduate Scholarship at Digital Commons @ Butler University. It has been accepted for inclusion in Undergraduate Honors Thesis Collection by an authorized administrator of Digital Commons @ Butler University. For more information, please contact digitalscholarship@butler.edu.

BUTLER UNIVERSITY HONORS PROGRAM

Honors Thesis Certification

Please type all information in this section:

Applicant Colin Frederick Sanberg
(Name as it is to appear on diploma)

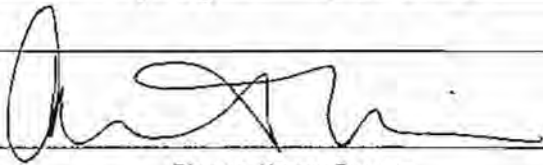
Thesis title Implementing Quantum Random Walks in Two-Dimensions
with Application to Diffusion-Limited Aggregation

Intended date of commencement May 12, 2007

Read, approved, and signed by:

Thesis adviser(s)  4/20/07
Date

Reader(s) Jan W. Kasik 4/20/07
Date

Certified by  5/23/07
Date
Director, Honors Program

For Honors Program use:

Level of Honors conferred: University Cum Laude

Departmental _____

LB
701
.883-6
-258
-17

**Implementing Quantum Random Walks in
Two-Dimensions with Application to
Diffusion-Limited Aggregation**

A Thesis

Presented to the Department of Physics & Astronomy

College of Liberal Arts and Sciences

and

The Honors Program

of

Butler University

In Partial Fulfillment

of the Requirements for Graduation Honors

Colin Frederick Sanberg

April 20, 2007

Table of Contents

Abstract.....	3
Introduction.....	4
Methodology.....	7
Results.....	14
Conclusions.....	20
Appendix of Tables & Figures.....	22
First Quantum Model Program Code ⁺	26
First Classical Model Program Code.....	33
Second Quantum Model Program Code.....	40
Second Classical Model Program Code.....	47
References.....	52

⁺ Code copied from FORTRAN

Abstract

This study simulates random movement and aggregation of particles in two-dimensional space based upon both quantum and classical mechanics. Using an original computer program to perform the calculations, the objective is to compare how quantum effects influence the random movement of a particle in comparison to the classical random movement. These effects are further studied by analyzing how the amassing of particles around a "seed" is affected by the differences in the random movement. Using the classical models that were generated as the basis of comparison, the initial results show that the quantum model aggregate grows at a slower rate than the classical case. Also, the quantum model grows in a more amorphous manner than the clear branching of the classical example. In an effort to more accurately simulate the behavior of the probability function as it encounters other particles, both the quantum and classical models were adjusted. This yielded a quantum aggregation that developed more similarly than the classical model. The primary difference in the quantum model was a noticeable lack of symmetry as the particles amassed around the seeded particles. It is possible that this iteration of the quantum model develops more rapidly than the classical model, though more simulations are needed to further test this. The effect other particles have on the development of the probability function also needs to be further examined to ensure that it is being modeled as accurately as possible.

Introduction

Since its discovery and subsequent development, quantum mechanics has profoundly changed how many scientists view the natural world, particularly at the microscopic level. One of the central tenants of quantum mechanics is the superposition of states, which in essence acts as a superposition of realities for individual particles to exist in. To further explain how this is possible, a tiny particle such as an electron exhibits behavior similar to both a particle and a wave. Pioneered by scientists such as Niels Bohr and Louis Victor de Broglie, this wave-particle duality states that subatomic bodies exist in both states at the same time, and can therefore have the probability of existing in multiple places at the same instance. Standing waves provide the best model for this seeming contradiction. A standing wave has the greatest amplitude at the central location between the two fixed nodes, which corresponds to the location where the particle has the greatest probability of being located when measured, but other amplitudes of lesser value exist between the wave nodes. The Heisenberg Uncertainty Principle states that the accuracy of measuring a particle's state, such as its instantaneous position or momentum, is limited so that the product of both values' standard deviations is equal to a constant, $\frac{h}{4\pi}$, where h is Planck's constant[†]. Taking these two principles in combination, a particle has the probability of being in multiple locations at any particular moment in time until it is physically measured, the value of which is accurate only in relation to the measurement of the momentum.

[†] $h = 6.626 \times 10^{-34} \text{ [J}\cdot\text{s]}$

The movement of a particle based upon its two-dimensional probability wave establishes one of the main focuses of this study. Given a particle's initial position, it has a certain probability to move to another location within a certain distance in a wave-like manner. As the wave moves further from the initial location, it spreads out. Therefore the particle has a higher probability of moving to a nearby location, but larger steps are not discounted, just more improbable. These probability values become factors during the random walk process.

Random walks, as the name implies, deal with the movement of an object wherein the location of each step is determined at random through some means. The simplest example of this is the movement of a particle along a line so that it can only move either to the left or to the right with each step. The flip of a coin randomly determines which way the particle moves. The two-dimensional analog of this idea is sometimes referred to as a "drunken walk" because each small step is taken in a random direction, with one coin dictating forward or backward movement and another choosing left or right. This is a prime example of a classical random walk because the movement is based upon Newtonian mechanics, which is demonstrated in the fact that the state of the particle does not affect how its movement progresses. A quantum random walk (QRW) would be similar in nature; however the internal state of the particle would influence the outcome of the coin flip as well as the probability of the location for each step. Moreover, the particle would occupy different locations at the same time in a wave-like manner. Currently, quantum random walks are a relatively new field of study, as stated by Julia Kempe (2003). Existing research in this area focuses primarily in modeling quantum

random walks in one-dimensional space, with higher dimensions limited to mostly theoretical description, which is seen in work done by Mackay et al (2002). Random walks equate to a random exploration, which can be used, for example, to sample or explore large data structures. They also form the basis for many computer algorithms, so that a significant difference between QRWs and classical random walks may lead to the development of more powerful algorithms.

Diffusion-limited aggregation (DLA) is a process in which randomly moving particles accumulate when they come in contact with a seed particle and, in time, with other amassed particles. This simple rule of motion leads to the formation of complex branching structures similar in nature to those found in the formation of snowflakes, trees, and coral reefs. These structures have self-similar patterns that repeat themselves at smaller and smaller scales, and are called fractals. Fractals are geometric figures that are said to have infinite detail, because as a fractal is divided into parts, each of the components has a shape and structure similar to the original image. The formation of these shapes via DLA is the other primary focus of this study. It aims to observe how a QRW affects fractal growth in comparison to a classical random walk. This visualization will help understand how random movement of a particle based upon quantum mechanics differs from classical movement.

Methodology

The primary basis for the calculation of two-dimensional QRWs was put forth in the paper by Mackay et al (2003). A QRW is obtained by attributing an internal state to the particle that will undergo the random movement. Looking first at a one-dimensional walk for this study, the particle will have a spin-1/2 system with an internal Hilbert space $H_{int} = H_2$, with basis state $|\epsilon_1\rangle = |\pm\rangle$. For two-dimensional space, the Hilbert space gets extrapolated to $H_{int} = H_2 \otimes H_2$, with basis states $|\epsilon_1 \epsilon_2\rangle = |\epsilon_1\rangle \otimes |\epsilon_2\rangle$, where $|\epsilon_i\rangle = |\pm\rangle$. This means that the particle can have either a positive or negative spin to it in either the first or second quantum bit (or qubit). The positive or negative value in ϵ_1 or ϵ_2 will determine if the particle will move positively or negatively in its respective dimension, meaning four resultant directions of motion. The spatial state of a two-dimensional lattice is denoted by $H_{spatial}$ such that the Hilbert space is defined by the basis states $|ij\rangle = |i\rangle \otimes |j\rangle$, where i and j are both integers that define the location of the particle in two-dimensional discrete space. The total state of the particle is therefore described by a state

$$H_T = H_{spatial} \otimes H_{int} \dots (1)$$

To simulate the coin flip that determines particle movement, two separate unitary operators are used to first transform the internal state into a superposition of multiple states and then select the movement of the particle based upon the internal state. The Hadamard transformation is the first operator. For a one-dimensional QRW, the Hadamard transformation is of the form

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \dots (2)$$

Extrapolating this for two-dimensional QRWs yields

$$\mathbf{H}_2 = \mathbf{H} \otimes \mathbf{H} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \dots (3)$$

This transformation acts solely on the internal state of the particle. The internal states created by a two-dimensional system and their resulting superpositions are detailed below in **Table 1**.

Internal State	Shorthand Label	Superposition
$ ++\rangle$	1	$\frac{1}{2}(++\rangle + +-\rangle + --\rangle + -+\rangle)$
$ +-\rangle$	2	$\frac{1}{2}(++\rangle - +-\rangle + --\rangle - -+\rangle)$
$ --\rangle$	3	$\frac{1}{2}(++\rangle + +-\rangle - --\rangle - -+\rangle)$
$ -+\rangle$	4	$\frac{1}{2}(++\rangle - +-\rangle - --\rangle + -+\rangle)$

Table 1: Internal States and Their Post-Hadamard Superpositions

After applying the Hadamard transformation to the particle, a unitary operator, \mathbf{F} , is then utilized to move the position of the particle based upon its internal state. The \mathbf{F} is defined as

$$\begin{aligned} \mathbf{F}(|i, j\rangle \otimes |++\rangle) &= |i, j+1\rangle \otimes |++\rangle \\ \mathbf{F}(|i, j\rangle \otimes |+-\rangle) &= |i+1, j\rangle \otimes |+-\rangle \\ \mathbf{F}(|i, j\rangle \otimes |--\rangle) &= |i, j-1\rangle \otimes |--\rangle \\ \mathbf{F}(|i, j\rangle \otimes |-+\rangle) &= |i-1, j\rangle \otimes |-+\rangle \end{aligned} \dots (4)$$

Thus, for internal state $|++\rangle$ the particle moves one space up, for $|+-\rangle$ it moves to the right, down for $|--\rangle$, and to the left for internal state $| - + \rangle$. It should be noted that these movements are assigned for each of the four possible internal state combinations rather than simply using one qubit value for movement in the x-direction and one for movement in the y-direction. This was done to have the particle move either vertically or horizontally each time rather than always in one of four diagonal directions. In addition, the F operator does not alter the internal state of the particle, but rather transforms the superposition state into a superposition state of a particle that has moved in one of the four cardinal directions. The two operators are used in this alternating manner each iteration, causing the spatial and internal degrees of freedom to become entangled.

These calculations were performed using original FORTRAN code that was run on a UNIX-based computer. The development of the code occurred in stages. After writing an initial program to simulate the QRW, it was modified to simulate a classical random walk. This allowed for generation of a basis to compare the QRW to. Upon fine-tuning the classical walk program, the quantum model program was further amended.

The particle was simulated by having a value of one at a location in an empty three-dimensional matrix, the first two coordinates defining the location and the third location defining the internal state. For example, a value of one at location (9, 13, 3) meant that a particle located at x-coordinate 9 and y-coordinate 13 had an internal state of $|--\rangle$. The starting location of the particle is selected using a random number generator. The generator selects a number between zero and one, which is then multiplied by the

maximum location value of the matrix (in a 99×99 matrix, the random number is multiplied by 99) and then rounded to the nearest integer. The particles always begin with an internal state of $|++\rangle$. After the initial location is selected, the Hadamard and F operators are performed in an alternating manner on the location matrix. As the operators progress through the individual matrix elements, the changes they enact are reflected in a secondary location matrix so as to not influence the calculations being performed on the initial matrix. Once the operator has acted upon the whole matrix, it is set equal to the secondary matrix, which is then reset to zero. This is necessary because the operators are dependent upon the values in the locations at the same time they are changing the values of adjacent locations so as to simulate the propagation of the probability wave. The changes are made in the secondary matrix so as to not affect the current iteration of the operations.

To calculate the probability of a particle selecting a location, the square of the values in each state are summed for a given location. After n iterations of alternately applying the two operators, the particle is in an entangled state $|\Psi_n\rangle \in H_T$, so the probability that it is found in location (i, j) is given by the equation

$$P_i = \left| \langle i, j | \otimes \langle ++ | \Psi_n \rangle \right|^2 + \left| \langle i, j | \otimes \langle +- | \Psi_n \rangle \right|^2 + \left| \langle i, j | \otimes \langle - - | \Psi_n \rangle \right|^2 + \left| \langle i, j | \otimes \langle - + | \Psi_n \rangle \right|^2 \dots \quad (5)$$

Although the internal state of the particle influences how its probability of relocation develops, it is not important for the actual selection of the location. The probability values are then stored in another matrix with values corresponding to their respective positions in the location matrix. Summing all the values in the probability matrix checks

that the total probability of the system remains at one, ensuring that the particle must go *somewhere*. Also, as the probability wave reaches the edges of the matrix, it wraps itself around to the other side, creating a sort of miniature globe that it walks over. A particle in the upper most location that will have a probability of moving upwards despite no place to go has that probability transferred to the location at the bottom of that same column.

The random selection of the location is done by taking the probability matrix and making each element equal to the sum of the elements up to and including it. A random number is generated, and the first element encountered to be greater than the random number becomes the chosen location for the particle. Consider a 3×3 probability matrix with particle located in the center having given probabilities of moving to the surrounding locations:

$$[\mathbf{P}] = \begin{bmatrix} 0.05 & 0.20 & 0.05 \\ 0.20 & 0.00 & 0.20 \\ 0.05 & 0.20 & 0.05 \end{bmatrix}$$

The probability matrix then becomes

$$[\mathbf{P}] = \begin{bmatrix} 0.05 & 0.25 & 0.30 \\ 0.50 & 0.50 & 0.70 \\ 0.75 & 0.95 & 1.00 \end{bmatrix}$$

Looking at the elements row by row, a randomly generated number of 0.67953 would first encounter an element greater than itself at $\mathbf{P}(3,2)$. The coordinate (3,2) would then become the location selected for the particles random movement.

Since applying the QRW to diffusion-limited aggregation (DLA) is the ultimate goal, the random selection of locations is taken a step further. DLA looks at how groups form as randomly moving particles cluster together. The program emphasizes this by initializing the clustering with a “seed” in the matrix that maintains the final selected locations. A group of locations are selected at the beginning of the program a 6×6 square located just off the center of the selected location matrix. If the location randomly selected does not lay adjacent to a location that is already occupied, then the particle is discarded and a new particle begins the process all over again. Initially, these particles will be encountering only the seeded locations, but over time the amassed particles take on their own shape.

For the classical case, this system all remains the same, with the exception that the **H** and **F** operators are only applied once to the particle before choosing a new location, giving it a 0.25 probability of moving to each of the four surrounding locations: up, right, down, and left. The primary difference between the models is that the particle is allowed to take a designated number of steps so that it has the opportunity to aggregate with the seeded particles. After so many steps have been taken, in this case 1,000, without coming in contact with other particles, it is discarded and another particle begins the process again. As with the probability calculations, the matrix space wraps in on itself. Therefore, if a particle is at the edge of the matrix and wants to move to a location not defined in the matrix, it finds itself on the other side.

After the initial results, both the quantum and the classical programs were modified to change how the probability function behaves when it encounters other particles, as well

as slightly increase the chances of particles encountering each the initial seed. Instead of randomly beginning anywhere within the matrix, the particles randomly appear on a circular boundary that is defined around the perimeter of the matrix. This is done to make the initial distance from the aggregate the same for all particles, only varying their position as a function an angle and not a radial distance.

In the previous model for both the classical and quantum cases, as the probability function developed it reflected off of any of the aggregate that it encountered. This caused the probability to cluster together in particular areas in between arms of the amassed particles. In an effort to counteract this, the second iteration of the model would allow for the selection of a location once the probability function encounters another particle. If the randomly selected location was not immediately next to that particle, the probability for the particle being in that specific location was set to zero and the other location probabilities were adjusted accordingly. This equates to taking a measurement and definitively finding that a particle was *not* located in the spot being examined, so the possibility of it being located in the other locations is increased just a little bit. Once this check is performed, the probability function is allowed to grow as before until it encounters another particle.

Results

The initial iterations of the quantum and classical models yield visibly different trends in the aggregate growth. The particle clusters generated by the classical model display the beginnings of common fractal growth. The particles are extending from their respective seeds in a branching fashion.

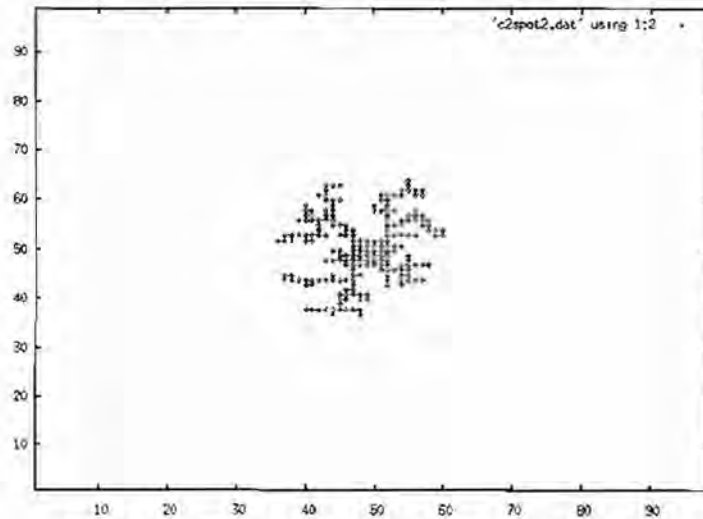


Figure 1: Classical DLA with 1,000 Particles

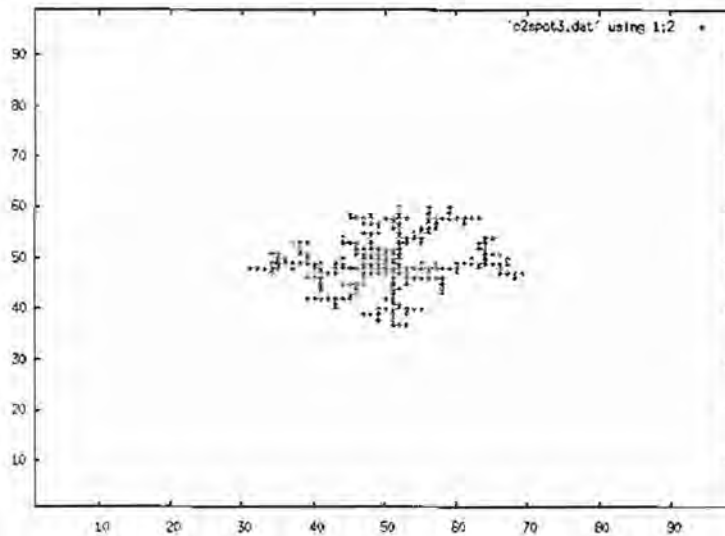


Figure 2: Classical DLA with 1,000 Particles

Both Figure 1 and Figure 2 are consistent with the shapes generated from classical random walk DLA. These shapes, commonly referred to as Brownian trees, have fairly

evenly distributed branching in all directions around the central seed, which is located at the center of each of the growths. This supports that the classical model is working as it should by producing results consistent with established findings.

The data gathered from the quantum simulations show a greatly different pattern of growth. Simulations that allow 1,000, 2,500 and 5,000 individual particles a chance to aggregate all show clusters that do not branch of as dramatically as the classical case.

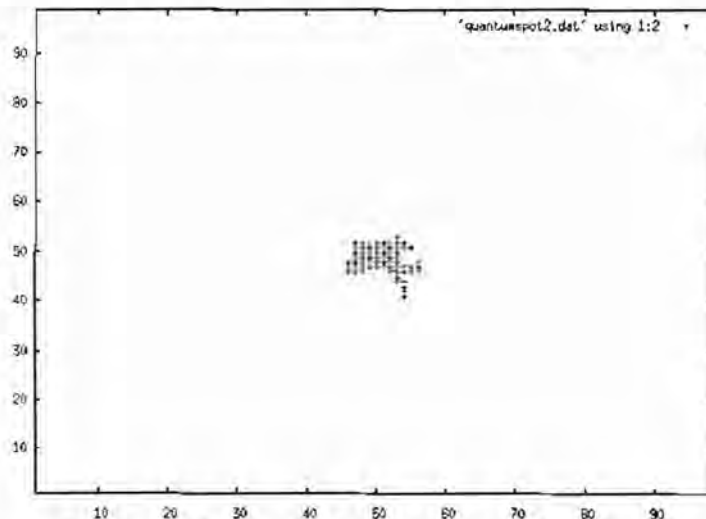


Figure 3: Quantum DLA with 1,000 Particles

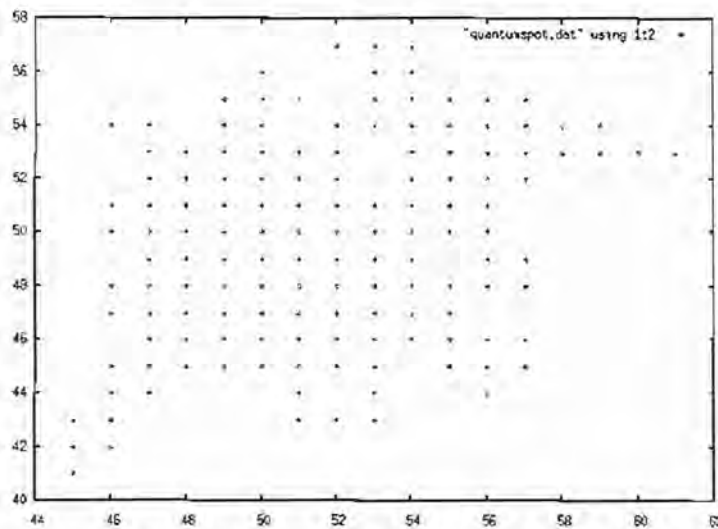


Figure 4: Quantum DLA with 2,500 Particles

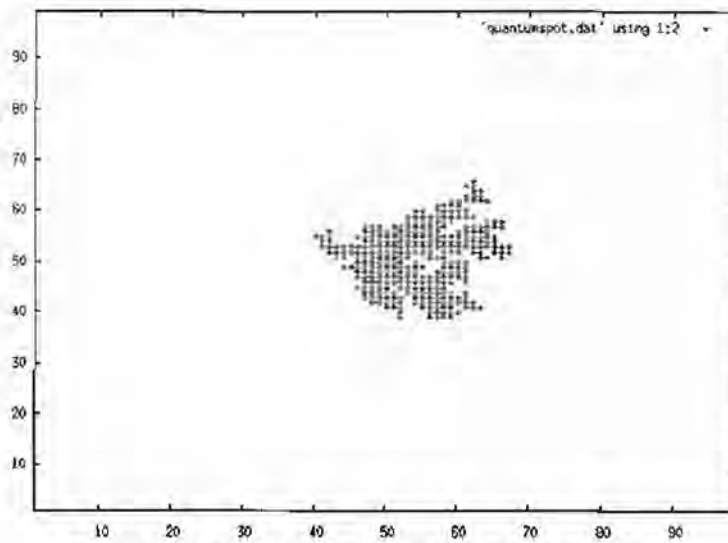


Figure 5: Quantum DLA with 5,000 Particles

Figures 3-5 each illustrate a model that grows in a more amorphous manner rather than classical simulation. This is promising, because it supports the fact that the quantum effects of the particle have a prevalent impact on how the particles move around and amass together. Also, the simulation that only runs 1,000 particles, the same amount as in the two classical simulations, shows that the quantum aggregation does not appear to happen as quickly as the in the classical case. However, it is possible that the different pattern of growth occurs because the probability function is getting confined in areas as it reflects of aggregated particles. This would cause a disproportionately high probability in certain areas which may not be consistent with how the behavior would naturally occur.

The second version of both models attempts to take this into account as described above and has yielded distinctly different results for the quantum model, though the classical model remains similar in nature to the expected out come.

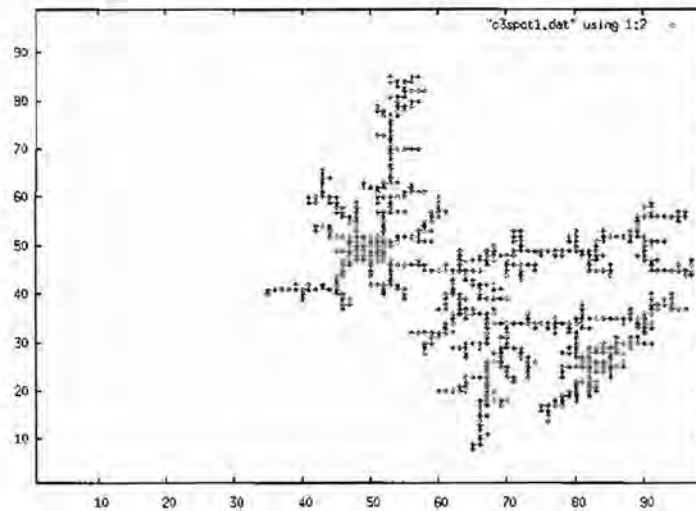


Figure 6: Classic DLA with 4,000 Particles

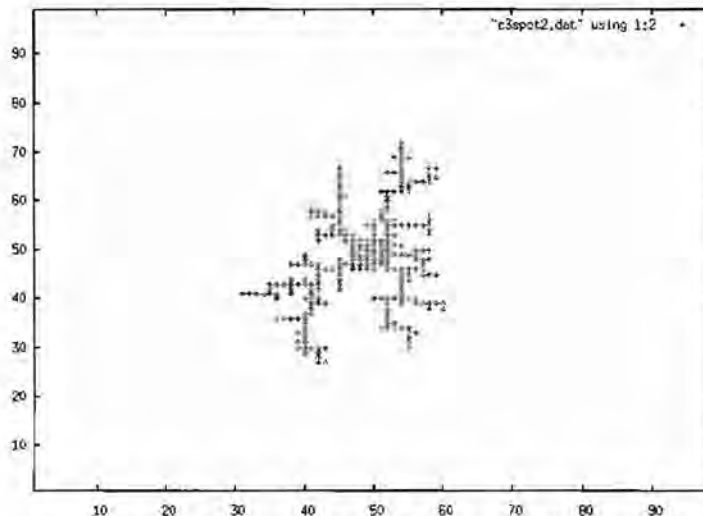


Figure 7: Classic DLA with 4,000 Particles

The above results still display development similar to the expected Brownian trees, though the aggregate in Figure 6 does not have very symmetric growth. The aggregate has spread out so far that the circular perimeter where the particles begin is identifiable. As the amassed particles approach closer to the starting location, the particles have a greater chance of attaching to the aggregate. Therefore, Figure 6 may not display the most favorable classical conditions. Figure 7, on the other hand, display results more consistent with those expected.

The quantum model results from the second iteration of the programs display development more similar to the classical model than in the previous model.

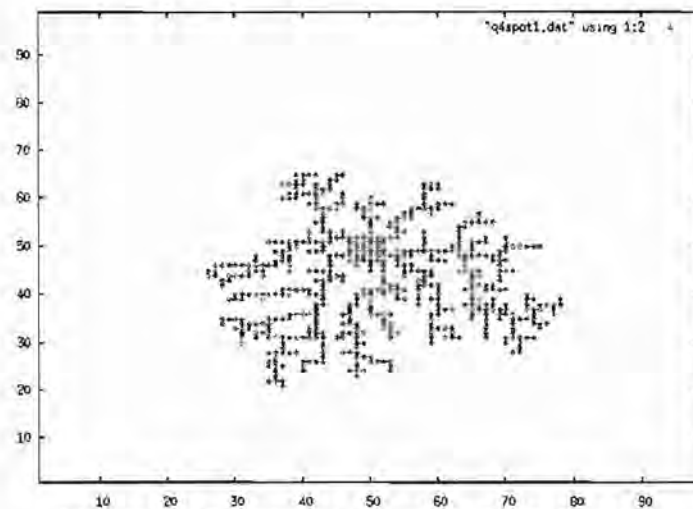


Figure 8: Quantum DLA with 4,000 Particles

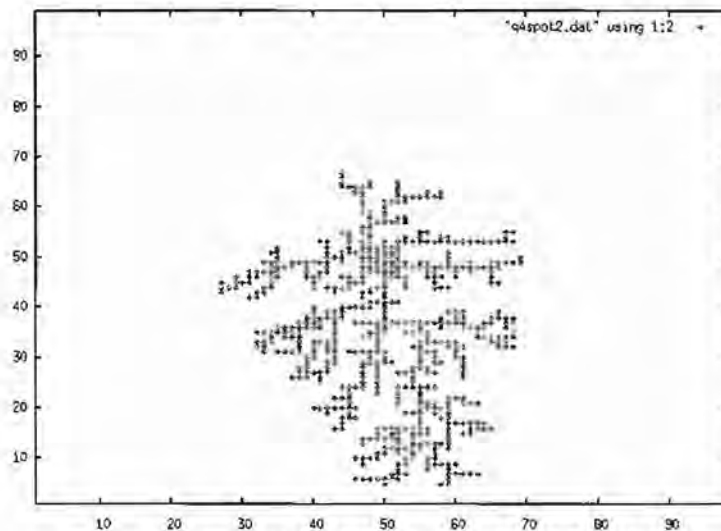


Figure 9: Quantum DLA with 4,000 Particles

Figure 8 and Figure 9 show aggregates that are drastically more similar to the classical Brownian tree model. The thin branching that occurs rather than the amorphous growing that occurred in the previous quantum model suggests that the clustering of the probability function may have been a factor in the previous difference in the growth pattern. This may occur because as specific locations are discounted from having

particles located there, the probability function approaches a more classical model given enough steps develop. These quantum models also appear to grow at a rate comparable to the classical models in this iteration of the programs. All simulations simulated a total of 4,000 particles and yielded similar sized structures. However, if the first classical simulation (Figure 6) is to be considered somewhat of an anomaly for approaching the boundary, it could be considered that the quantum model grew more rapidly in comparison to the results seen in Figure 7. Another trend to note in these results is that both of the quantum models in the second round of simulations tend to grow downward rather than symmetrically around the seed. This is most likely due to the fact that as the probability function develops for a particle initially in the $|++\rangle$ state, it does not grow symmetrically about the central starting location. One way to counteract this in future study would be to randomly select the initial state of the particle as well as its location.

Conclusions

Judging from the data obtained thus far, it can be seen that the quantum effects of the particle have the ability to play a significant role in influencing how the particles move randomly about in two-dimensional space. The degree of these effects is dependent upon the accuracy of the modeling of the quantum effects and the development of the probability function. In the preliminary batch of simulations, this difference in random movement in turn affects how the particles aggregate around the seed. The quantum models depict more centralized gathering of the particles. The amassed particles seen here do not appear to form in a traditional fractal-like manner, though this may be because it takes longer for the pattern to develop or emerge. It may also be due to the probability function growing disproportionately in certain areas as it reflects off the amassed particles. The simulation that run 5,000 particles already displays some branching, though they tend to favor the one side of the aggregate rather than developing symmetrically like the classical model. This is most likely due to the fact that the quantum locations are being selected via a probability wave that is spreading out from a randomly selected location and is effectively only being allowed to take one big step. The classical model allows the particle to take numerous small steps, in these simulations as many as 1000, to come in contact with the seed and other particles. Also, the first set of quantum model results appear to grow at a slower rate than the classical models. This might imply that the random movement based on quantum mechanics is not as conducive to amassing of particles in this manner.

The second set of models display more similar trends, though more simulations will need to be run before any more concrete conclusions can be extrapolated. Both the classical

and quantum models develop at comparable rates, though if the first classical model is not indicative of the regular behavior, it is possible that the quantum model actually develops more quickly in this instance. Also, the quantum model is displaying a tendency to develop favoring one side of the seed rather than symmetrically about it. This is most likely do to the asymmetrical probability function which should be taken into account by randomly selecting the initial state of the particles. The bunching of the particles into amorphous growths appears to have been offset by the changes to the program, in turn displaying development much more similar to the classical model.

The differences between the classical and quantum models are clearly observed in these simulations, although those differences are dissimilar between the two models. For future study, the behavior of the probability function as it develops and comes into contact with other particles needs to be examined more closely to ensure that the model is simulated the actual behavior as accurately as possible. As this is better understood, the programs can be modified to represent the systems faithfully and provide more decisive conclusions on the difference between the quantum and classical models.

Appendix of Tables & Figures

Internal State	Shorthand Label	Superposition
$ ++\rangle$	1	$\frac{1}{2}(++\rangle + +-\rangle + --\rangle + -+\rangle)$
$ +-\rangle$	2	$\frac{1}{2}(++\rangle - +-\rangle + --\rangle - -+\rangle)$
$ --\rangle$	3	$\frac{1}{2}(++\rangle + +-\rangle - --\rangle - -+\rangle)$
$ -+\rangle$	4	$\frac{1}{2}(++\rangle - +-\rangle - --\rangle + -+\rangle)$

Table A1: Internal States and Their Post-Hadamard Superpositions

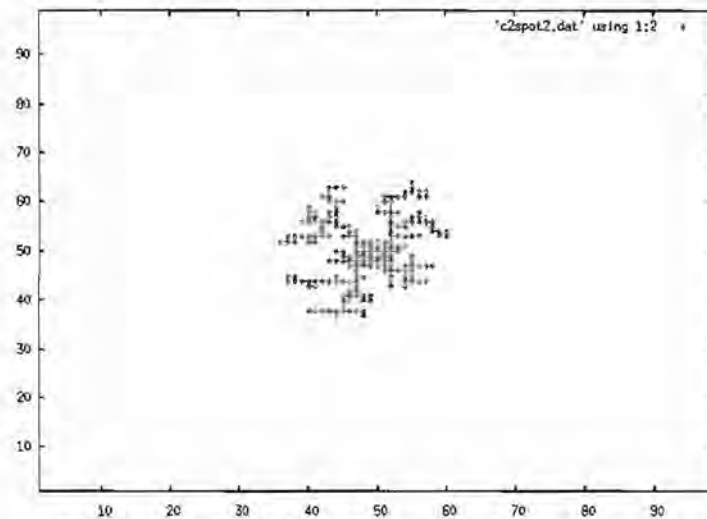


Figure A1: Classical DLA with 1,000 Particles

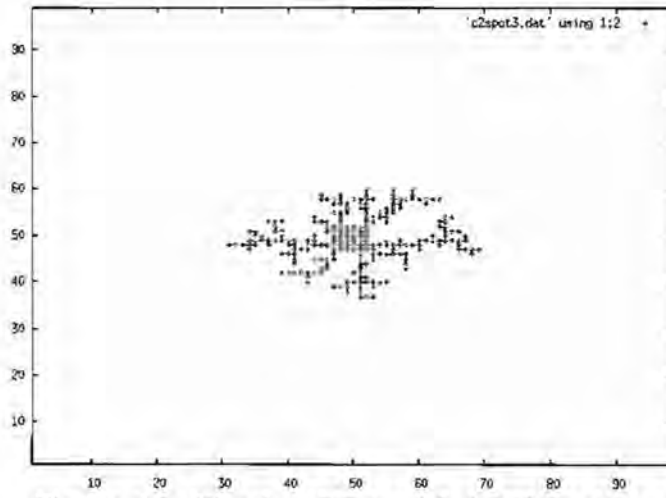


Figure A2: Classical DLA with 1,000 Particles

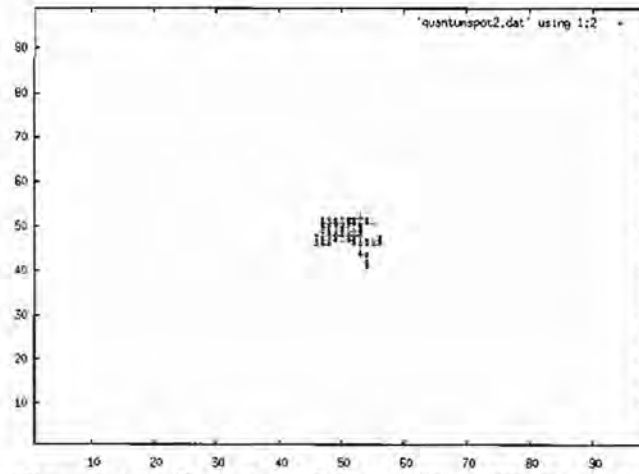


Figure A3: Quantum DLA with 1,000 Particles

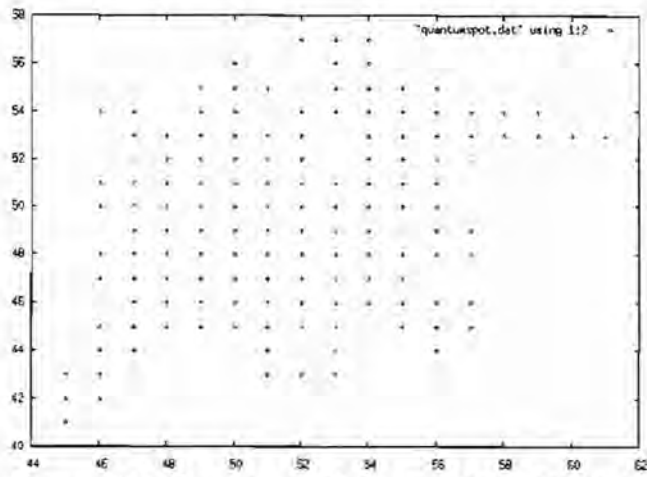


Figure A4: Quantum DLA with 2,500 Particles

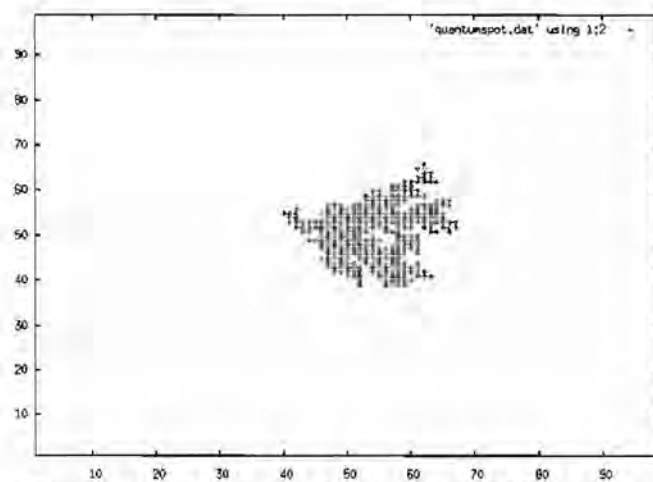


Figure A5: Quantum DLA with 5,000 Particles

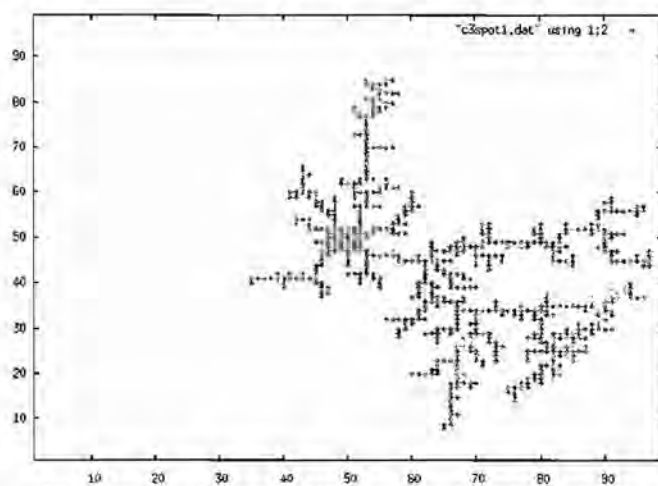


Figure A6: Classic DLA with 4,000 Particles

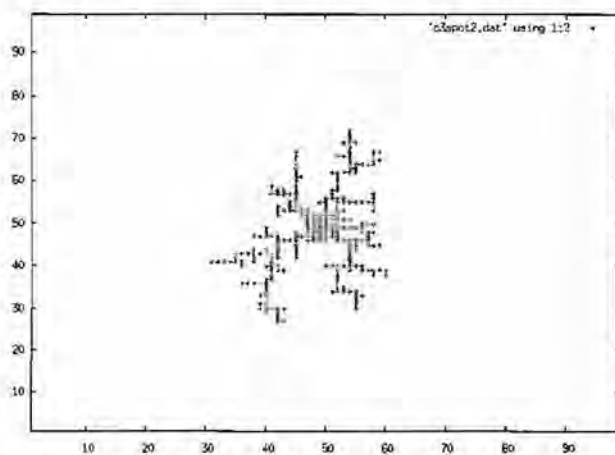


Figure A7: Classic DLA with 4,000 Particles

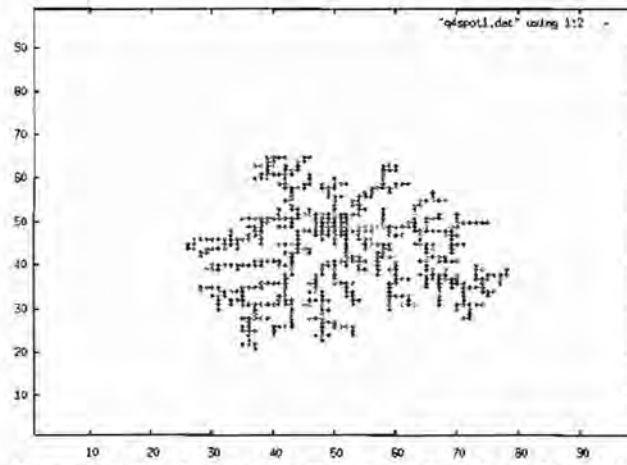


Figure A8: Quantum DLA with 4,000 Particles

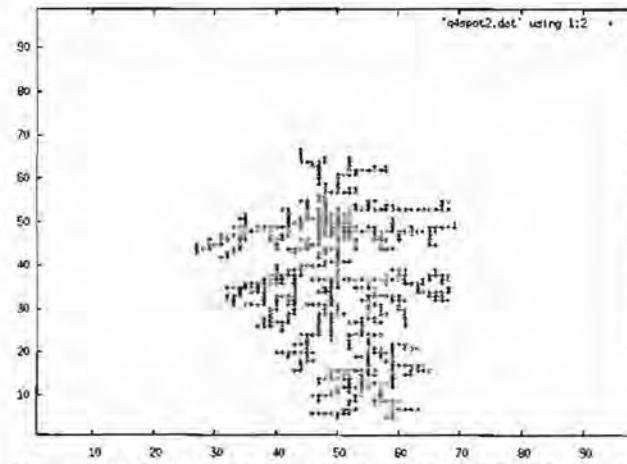


Figure A9: Quantum DLA with 4,000 Particles

First Quantum Model Program Code

```

PROGRAM quantum2

C-----C
C
C   PROJECTION
C
C   This program aims to simulate random movement
C   of a particle based upon quantum mechanics and
C   take into account the internal states of the particles.
C-----C
C
C   PARAMETERS
C
C   LOC = LOCATION MATRIX
C   STP = STEP NUMBER
C   BIGSTEP = NUMBER OF PARTICLES
C   STPMAX = MAXIMUM NUMBER OF OPERATOR ITERATIONS
C   SEED2 = SEED LOCATIONS
C   PROB = PROBABILITY MATRIX
C   CHX = CHOSEN LOCATION
C   SUM = CHECK TO ENXURE THAT TOTAL PROBABILITY IS 1
C   SUMT = USED IN CONJUNCTION WITH 'SUM'
C   NN = DEFINES MATRIX SIZE
C   STRT = DEFINES CENTER OF MATRIX
C   DLA = MATRIX OF CHOSEN LOCATIONS
C   NEIGHBORS = KEEPS TRACK OF LOACTIONS WITH NEIGHBORS
C   FLAG = CHECKS FOR NEIGHBOR PARTICLES
C-----C

IMPLICIT REAL *8 (A-H,O-Z)
INTEGER STP,SEED2,NN,DLA(99,99),BIGSTEP,II,JJ
INTEGER NEIGHBORS (99,99)
INTEGER SEED,IIp,IIm,JJp,JJm,FLAG,STPMAX
REAL *8 PROB(99,99),CHX,SUM,LOC1(99,99,4),LOC2(99,99,4)
REAL *8 SUMT
OPEN(4,FILE='quantumspot2.dat',STATUS='UNKNOWN')

C-----C
C   SET DEFAULT PARAMETERS
C-----C

ZERO = 0.0D0
ONE = 1.0D0
SUM = ZERO
NN = 99
SEED2 = 47
STP = 0
BIGSTEP = 0
STPMAX=30

C-----C
C   INITIALIZE RANDOM NUMBER GENERATOR
C-----C

SEED=TIME()
CALL=RAND(SEED)

```

```

C-----C
C   INTERNAL STATE DEFINITIONS   C
C
C   1 = ++   MOVE UP ONE STEP   C
C   2 = +-   MOVE RIGHT ONE STEP C
C   3 = --   MOVE ONE STEP DOWN  C
C   4 = -+   MOVE ONE STEP LEFT  C
C-----C

C-----C
C   SET LOCATION MATRICES       C
C-----C

      DO 10 I=1,NN,+1
      DO 11 J=1,NN,+1
      DO 12 K=1,4,+1
          LOC1(I,J,K)=ZERO
          LOC2(I,J,K)=ZERO
12      CONTINUE
          DLA(I,J)=0
11      CONTINUE
10      CONTINUE

      LOC1(INT(RAND(0)*NN),INT(RAND(0)*NN),1) = ONE

      DO I=0,5
      DO J=0,5
          DLA(SEED2+I, SEED2+J)=1
      END DO
      END DO

      DO WHILE (BIGSTEP.LT.1000)
C-----C
C   Update Neighbors matrix     C
C-----C
          DO 807 I=1,NN,1
          DO 808 J=1,NN,1
              NEIGHBORS(I,J)=0
808      CONTINUE
807      CONTINUE

          DO 707 I=1,NN,1
          DO 708 J=1,NN,1
              IF (DLA(I,J).NE.0) THEN
                  IF (I.LT.NN) THEN
                      IIp=I+1
                  ELSE
                      IIp=1
                  ENDIF
                  IF (J.LT.NN) THEN
                      JJp=J+1
                  ELSE
                      JJp=1
                  ENDIF
                  IF (I.GT.1) THEN
                      IIm=I-1
                  ELSE
                      IIm=99
                  ENDIF
                  IF (J.GT.1) THEN
                      JJm=J-1

```

```

        ELSE
          JJm=99
        ENDIF
        NEIGHBORS(IIp,J) = 1
        NEIGHBORS(IIm,J) = 1
        NEIGHBORS(I,JJp) = 1
        NEIGHBORS(I,JJm) = 1
      ENDIF
708    CONTINUE
707    CONTINUE

      DO WHILE (STP.LT.STPMAX)

```

```

C-----C
C      Evolve wave function      C
C-----C
C      HADAMARD TRANSFORMATION  C
C-----C

```

```

      DO 607 I=1,NN,1
      DO 608 J=1,NN,1
      DO 609 K=1,4
        LOC2(I,J,K)=0
609    CONTINUE
608    CONTINUE
607    CONTINUE

      DO 20 I=1,NN,+1
      DO 21 J=1,NN,+1
      DO 22 K=1,4,+1
        IF (LOC1(I,J,K).NE.0.0D0) THEN
          IF (K.EQ.1.0D0) THEN
            LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
            LOC2(I,J,2)=LOC2(I,J,2)+LOC1(I,J,K)
            LOC2(I,J,3)=LOC2(I,J,3)+LOC1(I,J,K)
            LOC2(I,J,4)=LOC2(I,J,4)+LOC1(I,J,K)
          ELSEIF (K.EQ.2.0D0) THEN
            LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
            LOC2(I,J,2)=LOC2(I,J,2)-LOC1(I,J,K)
            LOC2(I,J,3)=LOC2(I,J,3)+LOC1(I,J,K)
            LOC2(I,J,4)=LOC2(I,J,4)-LOC1(I,J,K)
          ELSEIF (K.EQ.3.0D0) THEN
            LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
            LOC2(I,J,2)=LOC2(I,J,2)+LOC1(I,J,K)
            LOC2(I,J,3)=LOC2(I,J,3)-LOC1(I,J,K)
            LOC2(I,J,4)=LOC2(I,J,4)-LOC1(I,J,K)
          ELSE
            LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
            LOC2(I,J,2)=LOC2(I,J,2)-LOC1(I,J,K)
            LOC2(I,J,3)=LOC2(I,J,3)-LOC1(I,J,K)
            LOC2(I,J,4)=LOC2(I,J,4)+LOC1(I,J,K)
          ENDIF
        ENDIF
      ENDIF
22    CONTINUE
21    CONTINUE
20    CONTINUE

      DO 23 I=1,NN,+1
      DO 24 J=1,NN,+1
      DO 25 K=1,4,+1

```

```

        LOC1(I,J,K)=LOC2(I,J,K)/(2.0D0)
        LOC2(I,J,K)=ZERO
25      CONTINUE
24      CONTINUE
23      CONTINUE

C-----C
C      F OPERATOR      C
C-----C

DO 30 I=1,NN,+1
DO 31 J=1,NN,+1
  IF (I.LT.NN) THEN
    IIp=I+1
  ELSE
    IIp=1
  ENDIF
  IF (J.LT.NN) THEN
    JJp=J+1
  ELSE
    JJp=1
  ENDIF
  IF (I.GT.1) THEN
    IIm=I-1
  ELSE
    IIm=99
  ENDIF
  IF (J.GT.1) THEN
    JJm=J-1
  ELSE
    JJm=99
  ENDIF

DO 32 K=1,4,+1
  IF (LOC1(I,J,K).NE.0.0D0) THEN
    IF (K.EQ.1) THEN
      IF (DLA(I,JJp).EQ.0) THEN
        LOC2(I,J,K)=ZERO
        LOC2(I,JJp,K)=LOC2(I,JJp,K)+LOC1(I,J,K)
      ELSE
        LOC2(I,J,K)=LOC1(I,J,K)
      ENDIF
    ENDIF

    IF (K.EQ.2) THEN
      IF (DLA(IIp,J).EQ.0) THEN
        LOC2(I,J,K)=ZERO
        LOC2(IIp,J,K)=LOC2(IIp,J,K)+LOC1(I,J,K)
      ELSE
        LOC2(I,J,K)=LOC1(I,J,K)
      ENDIF
    ENDIF

    IF (K.EQ.3) THEN
      IF (DLA(I,JJm).EQ.0) THEN
        LOC2(I,J,K)=ZERO
        LOC2(I,JJm,K)=LOC2(I,JJm,K)+LOC1(I,J,K)
      ELSE
        LOC2(I,J,K)=LOC1(I,J,K)
      ENDIF
    ENDIF

    IF (K.EQ.4) THEN

```

```

        IF (DLA(IIm,J).EQ.0) THEN
            LOC2(I,J,K)=ZERO
            LOC2(IIm,J,K)=LOC2(IIm,J,K)+LOC1(I,J,K)
        ELSE
            LOC2(I,J,K)=LOC1(I,J,K)
        ENDIF
    ENDIF
ENDIF
32    CONTINUE
31    CONTINUE
30    CONTINUE

    DO 33 I=1,NN,+1
    DO 34 J=1,NN,+1
    DO 35 K=1,4,+1
        LOC1(I,J,K)=LOC2(I,J,K)
35    CONTINUE
34    CONTINUE
33    CONTINUE

C-----C
C    CALCULATE MATRIX FOR PROBABILITY IN EACH LOCATION    C
C-----C

    DO 40 I=1,NN,+1
    DO 41 J=1,NN,+1
        PROB(I,J)=ABS(LOC1(I,J,1))**2+ABS(LOC1(I,J,2))**2+
1        ABS(LOC1(I,J,3))**2+ABS(LOC1(I,J,4))**2
        PROB(I,J)=PROB(I,J)*NEIGHBORS(I,J)*(1-DLA(I,J))
41    CONTINUE
40    CONTINUE

C    SUMT=ZERO

C    DO 50 I=1,NN,+1
C    DO 52 J=1,NN,+1
C        SUMT=SUM+PROB(I,J)
C 52    CONTINUE
C 50    CONTINUE

C-----C
C    SET PROB MATRIX FOR CHOOSING LOACTION    C
C-----C
C    Each location has a probability of the particle    C
C    choosing that spot. The probability of each    C
C    location will now be changed to a value so that    C
C    the probability is represented by the range    C
C    between the location and the location before it.    C
C    The total range is from 0 to 1, and the range    C
C    will be assigned row by row.    C
C-----C

    SUM=ZERO

    DO 60 I=1,NN,+1
    DO 61 J=1,NN,+1
        IF (PROB(I,J).NE.ZERO) THEN
            SUM=SUM+PROB(I,J)
            PROB(I,J)=SUM
        ENDIF
61    CONTINUE
60    CONTINUE

```



```

C-----C
C   CHOOSE LOCATION                               C
C-----C
C   Use random number generator to get a random value.      C
C   The location chosen by the particle will be decided      C
C   by the ranges now defined in the matrix PROB. If the    C
C   number is less than the value in a location, but        C
C   greater than the number in the previous location, it     C
C   comes to rest in the current location.                  C
C-----C

```

```

      CHX=RAND(0)
      FLAG = 0

      DO 70 I=1,NN,+1
      DO 71 J=1,NN,+1
        IF (CHX.LT.PROB(I,J)) THEN
          II=I
          JJ=J
          FLAG=1
          GOTO 888
        ENDIF
      71 CONTINUE
      70 CONTINUE

      888 CONTINUE

```

```

C-----C
C   This section dictates the location selection            C
C   based upon previously selected spots. If a              C
C   spot is already full, the surrounding spots are         C
C   randomly selected based upon whether they are           C
C   already occupied or not.                                C
C-----C

```

```

      IF (FLAG.EQ.1) THEN
        print *, II,JJ, "FOUND"
        DLA (II,JJ) = 1
        STP=STPMAX

```

```

C-----C
C   Particle keeps moving if it does                        C
C   not meet an occupied location.                          C
C-----C

```

```

      ELSE
        STP=STP+1
      ENDIF
    ENDDO
    STP=0
    print *, BIGSTEP
    BIGSTEP=BIGSTEP+1

    DO 1080 I=1,NN,+1
    DO 1081 J=1,NN,+1
    DO 1082 K=1,4,+1
      LOC1(I,J,K)=ZERO
    1082 CONTINUE
    1081 CONTINUE
    1080 CONTINUE

```

```
LOC1(INT(RAND(0)*NN),INT(RAND(0)*NN),1) = ONE
```

```
ENDDO
```

```
C-----C
C  WRITE IN DATA FILE  C
C-----C
```

```
DO 90 I=1,NN,+1
DO 91 J=1,NN,+1
  IF (DLA(I,J).NE.0) THEN
    WRITE(4,5) I,J
  ENDIF
91  CONTINUE
90  CONTINUE
```

```
CLOSE(4)
```

```
STOP
1  FORMAT(' ',1X,F12.4,' ',1X,I5,' ',1X,F9.6)
2  FORMAT(' ',6X,'X',' ',6X,'Y',' ',4X,'VALUE')
3  FORMAT(' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
1    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
2    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
3    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
4    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
5    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
6    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
7    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
8    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6)
4  FORMAT(' ',I5,' ',I5,' ',I5,' ',I5,' ',I5,
1    ' ',I5,' ',I5,' ',I5,' ',I5,' ',I5,
2    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
3    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
4    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
5    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
6    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5)
5  FORMAT(' ',1X,I9,' ',1X,I9)
END
```

First Classical Model Program Code

```

PROGRAM classic2

C-----C
C
C PROJECTION
C
C This program aims to simulate random movement
C of a particle based upon quantum mechanics and
C take into account the internal states of the particles.
C Using the quantum model, it aims to simulate a classical
C random walk by selecting a location after iteration.
C-----C
C
C PARAMETERS
C
C LOC = LOCATION MATRIX
C BIGSTEP = NUMBER OF PARTICLES
C STEPMAX = NUMBER OF STEPS PARTICLE CAN TAKE
C STP = STEP NUMBER
C PROB = PROBABILITY MATRIX
C CHX = CHOSEN LOCATION
C SUM = CHECK TO ENSURE THAT TOTAL PROBABILITY IS 1
C NN = DEFINES MATRIX SIZE
C SEED2 = SEED LOCATION
C DLA = MATRIX OF CHOSEN LOCATIONS
C FLAG = CHECKS FOR NEIGHBORING PARTICLES
C-----C

IMPLICIT REAL *8 (A-H,O-Z)
INTEGER STP,SEED2,NN,,DLA(99,99),BIGSTEP,II,JJ
INTEGER SEED,STEPSMAX,IIP,IIM,JJP,JJM,FLAG
REAL *8 PROB(99,99),CHX,SUM,LOC1(99,99,4),LOC2(99,99,4)
OPEN(1,FILE='c2.dat',STATUS='UNKNOWN')
OPEN(2,FILE='c2prob.dat',STATUS='UNKNOWN')
OPEN(3,FILE='c2spot.dat',STATUS='UNKNOWN')
OPEN(4,FILE='c2spot3.dat',STATUS='UNKNOWN')

C-----C
C SET DEFAULT PARAMETERS C
C-----C

ZERO = 0.0D0
ONE = 1.0D0
SUM = ZERO
NN = 99
STRTY = (NN+1)/2
STRTX = (NN+1)/2
SEED2 = 47
STP = 0
BIGSTEP = 0
STEPSMAX = 1000

C-----C
C INITIALIZE RANDOM NUMBER GENERATOR C
C-----C

SEED=TIME()
CALL=RAND(SEED)

```

```

C-----C
C   INTERNAL STATE DEFINITIONS   C
C   C
C   1 = ++   MOVE UP ONE STEP   C
C   2 = +-   MOVE RIGHT ONE STEP C
C   3 = --   MOVE ONE STEP DOWN C
C   4 = -+   MOVE ONE STEP LEFT C
C   C
C-----C

C-----C
C   SET LOCATION MATRICES       C
C-----C

      DO 10 I=1,NN,+1
      DO 11 J=1,NN,+1
      DO 12 K=1,4,+1
        LOC1(I,J,K)=ZERO
        LOC2(I,J,K)=ZERO
        DLA(I,J)=0
12    CONTINUE
11    CONTINUE
10    CONTINUE

      LOC1(INT(RAND(0)*NN),INT(RAND(0)*NN),1) = ONE

      DO 14 I=0,5,+1
      DO 15 J=0,5,+1
        DLA(SEED2+I,SEED2+J)=1
15    CONTINUE
14    CONTINUE

C-----C
C   HADAMARD TRANSFORMATION     C
C-----C

      DO WHILE (BIGSTEP.LT.1000)
      DO WHILE (STP.LT.STEPMAX)

        SUM=ZERO

        DO 20 I=1,NN,+1
        DO 21 J=1,NN,+1
        DO 22 K=1,4,+1
          IF (LOC1(I,J,K).NE.0.0D0) THEN
            IF (K.EQ.1.0D0) THEN
              LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
              LOC2(I,J,2)=LOC2(I,J,2)+LOC1(I,J,K)
              LOC2(I,J,3)=LOC2(I,J,3)+LOC1(I,J,K)
              LOC2(I,J,4)=LOC2(I,J,4)+LOC1(I,J,K)
            ELSEIF (K.EQ.2.0D0) THEN
              LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
              LOC2(I,J,2)=LOC2(I,J,2)-LOC1(I,J,K)
              LOC2(I,J,3)=LOC2(I,J,3)+LOC1(I,J,K)
              LOC2(I,J,4)=LOC2(I,J,4)-LOC1(I,J,K)
            ELSEIF (K.EQ.3.0D0) THEN
              LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
              LOC2(I,J,2)=LOC2(I,J,2)+LOC1(I,J,K)
              LOC2(I,J,3)=LOC2(I,J,3)-LOC1(I,J,K)
              LOC2(I,J,4)=LOC2(I,J,4)-LOC1(I,J,K)
            ELSE

```

```

        LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
        LOC2(I,J,2)=LOC2(I,J,2)-LOC1(I,J,K)
        LOC2(I,J,3)=LOC2(I,J,3)-LOC1(I,J,K)
        LOC2(I,J,4)=LOC2(I,J,4)+LOC1(I,J,K)
    ENDIF
ENDIF
22    CONTINUE
21    CONTINUE
20    CONTINUE

    DO 23 I=1,NN,+1
    DO 24 J=1,NN,+1
    DO 25 K=1,4,+1
        LOC1(I,J,K)=LOC2(I,J,K)/(2.0D0)
        LOC2(I,J,K)=ZERO
        PROB(I,J)=ZERO
25    CONTINUE
24    CONTINUE
23    CONTINUE

C-----C
C   F OPERATOR   C
C-----C

    DO 30 I=1,NN,+1
    DO 31 J=1,NN,+1
    DO 32 K=1,4,+1
        IF (LOC1(I,J,K).NE.0.0D0) THEN
            LOC2(I,J,K)=ZERO
            IF (K.EQ.1.0D0) THEN
                IF (J+1.LE.NN) THEN
                    LOC2(I,J+1,K)=LOC2(I,J+1,K)+LOC1(I,J,K)
                ELSE
                    LOC2(I,1,K)=LOC2(I,1,K)+LOC1(I,J,K)
                ENDIF
            ELSEIF (K.EQ.2.0D0) THEN
                IF (I+1.LE.NN) THEN
                    LOC2(I+1,J,K)=LOC2(I+1,J,K)+LOC1(I,J,K)
                ELSE
                    LOC2(1,J,K)=LOC2(1,J,K)+LOC1(I,J,K)
                ENDIF
            ELSEIF (K.EQ.3.0D0) THEN
                IF (J-1.GE.0) THEN
                    LOC2(I,J-1,K)=LOC2(I,J-1,K)+LOC1(I,J,K)
                ELSE
                    LOC2(I,NN,K)=LOC2(I,NN,K)+LOC1(I,J,K)
                ENDIF
            ELSE
                IF (I-1.GE.0) THEN
                    LOC2(I-1,J,K)=LOC2(I-1,J,K)+LOC1(I,J,K)
                ELSE
                    LOC2(NN,J,K)=LOC2(NN,J,K)+LOC1(I,J,K)
                ENDIF
            ENDIF
        ENDIF
    ENDIF
32    CONTINUE
31    CONTINUE
30    CONTINUE

    DO 33 I=1,NN,+1

```

```

      DO 34 J=1,NN,+1
      DO 35 K=1,4,+1
        LOC1(I,J,K)=LOC2(I,J,K)
35      CONTINUE
34      CONTINUE
33      CONTINUE

C-----C
C      CALCULATE MATRIX FOR PROBABILITY IN EACH LOCATION      C
C-----C

      DO 40 I=1,NN,+1
      DO 41 J=1,NN,+1
        PROB(I,J)=ABS(LOC1(I,J,1))**2+ABS(LOC1(I,J,2))**2+
1        ABS(LOC1(I,J,3))**2+ABS(LOC1(I,J,4))**2
41      CONTINUE
40      CONTINUE

      DO 50 I=1,NN,+1
      DO 52 J=1,NN,+1
        SUM=SUM+PROB(I,J)
52      CONTINUE
50      CONTINUE

c      DO 90 I=1,NN,+1
c      DO 91 J=1,NN,+1
c        WRITE(4,5) I,J,PROB(I,J)
c 91      CONTINUE
c 90      CONTINUE

C-----C
C      SET PROB MATRIX FOR CHOOSING LOACTION      C
C-----C
C      Each location has a probability of the particle      C
C      choosing that spot. The probability of each          C
C      location will now be changed to a value so that      C
C      the probability is represented by the range          C
C      between the location and the location before it.     C
C      The total range is from 0 to 1, and the range        C
C      will be assigned row by row.                          C
C-----C

      SUM=ZERO

      DO 60 I=1,NN,+1
      DO 61 J=1,NN,+1
        IF (PROB(I,J).NE.ZERO) THEN
          SUM=SUM+PROB(I,J)
          PROB(I,J)=SUM
        ENDIF
61      CONTINUE
60      CONTINUE

```

```

C-----C
C      CHOOSE LOCATION                                C
C-----C
C      Use random number generator to get a random value.      C
C      The location chosen by the particle will be decided      C
C      by the ranges now defined in the matrix PROB. If the    C
C      number is less than the value in a location, but        C
C      greater than the number in the previous location, it    C
C      comes to rest in the current location.                  C
C-----C

```

```
CHX=RAND(0)
```

```
DO 70 I=1,NN,+1
```

```
DO 71 J=1,NN,+1
```

```
IF (CHX.LT.PROB(I,J)) THEN
```

```
II=I
```

```
JJ=J
```

```
GOTO 888
```

```
ENDIF
```

```
71 CONTINUE
```

```
70 CONTINUE
```

```
888 CONTINUE
```

```

C-----C
C      This section dictates the location selection            C
C      based upon previously selected spots. If a              C
C      spot is already full, the surrounding spots are         C
C      randomly selected based upon whether they are           C
C      already occupied or not.                                 C
C-----C

```

```
IF (II.LT.NN) THEN
```

```
IIp=II+1
```

```
ELSE
```

```
IIp=1
```

```
ENDIF
```

```
IF (JJ.LT.NN) THEN
```

```
JJp=JJ+1
```

```
ELSE
```

```
JJp=1
```

```
ENDIF
```

```
IF (II.GT.1) THEN
```

```
IIm=II-1
```

```
ELSE
```

```
IIm=NN
```

```
ENDIF
```

```
IF (JJ.GT.1) THEN
```

```
JJm=JJ-1
```

```
ELSE
```

```
JJm=NN
```

```
ENDIF
```

```
FLAG = 0
```

```
IF (DLA(IIp,JJ).NE.0) THEN
```

```
FLAG=1
```

```
ENDIF
```

```
IF (DLA(IIm,JJ).NE.0) THEN
```

```

        FLAG=1
    ENDIF
    IF (DLA(II, JJp) .NE. 0) THEN
        FLAG=1
    ENDIF
    IF (DLA(II, JJm) .NE. 0) THEN
        FLAG=1
    ENDIF

    IF (FLAG, EQ. 1) THEN
        DLA(II, JJ)=1
        print *, II, JJ, "found"
        STP=STEPMAX
    
```

C-----C
 C Particle keeps moving if it does C
 C not meet an occupied location. C
 C-----C

```

        ELSE
            DO 80 I=1, NN, +1
            DO 81 J=1, NN, +1
            DO 82 K=1, 4, +1
                LOC1(I, J, K)=ZERO
                LOC2(I, J, K)=ZERO
82          CONTINUE
81          CONTINUE
80          CONTINUE

            LOC1(II, JJ, 1) = ONE
            STP=STP+1
        ENDIF

    ENDDO

    STP=0
    print *, bigstep
    BIGSTEP=BIGSTEP+1

        DO 100 I=1, NN, +1
        DO 101 J=1, NN, +1
        DO 102 K=1, 4, +1
            LOC1(I, J, K)=ZERO
            LOC2(I, J, K)=ZERO
102        CONTINUE
101        CONTINUE
100        CONTINUE

        LOC1(INT(RAND(0)*NN), INT(RAND(0)*NN), 1) = ONE

    ENDDO

    C-----C
    C WRITE IN DATA FILE C
    C-----C

    WRITE(1, 1) SUM, (BIGSTEP-1), CHX

    DO 99 I=1, NN, +1
        WRITE(2, 3) (PROB(I, J), J=1, NN, +1)
        WRITE(3, 4) (DLA(I, J), J=1, NN, +1)
99    CONTINUE
  
```



```

DO 90 I=1,NN,+1
DO 91 J=1,NN,+1
  IF (DLA(I,J).NE.0) THEN
    WRITE(4,5) I,J
  ENDIF
91 CONTINUE
90 CONTINUE

CLOSE(1)
CLOSE(2)
CLOSE(3)
CLOSE(4)

STOP
1  FORMAT(' ',1X,F12.4,' ',1X,I5,' ',1X,F9.6)
2  FORMAT(' ',6X,'X',' ',6X,'Y',' ',4X,'VALUE')
3  FORMAT(' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
1    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
2    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
3    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
4    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
5    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
6    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
7    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
8    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6)
4  FORMAT(' ',I5,' ',I5,' ',I5,' ',I5,' ',I5,
1    ' ',I5,' ',I5,' ',I5,' ',I5,' ',I5,
2    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
3    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
4    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
5    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
6    ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5)
5  FORMAT(' ',1X,I9,' ',1X,I9)
END

```

Second Quantum Model Program Code

```

PROGRAM quantum4

C-----C
C
C   PROJECTION
C
C   This program aims to simulate random movement
C   of a particle based upon quantum mechanics and
C   take into account the internal states of the particles.
C   Using the quantum model, it aims to simulate a classical
C   random walk by selecting a location after iteration.
C-----C
C
C   PARAMETERS
C
C   LOC = LOCATION MATRIX
C   STP = STEP NUMBER
C   PROB = PROBABILITY MATRIX
C   CHX = CHOSEN LOCATION
C   SUM = CHECK TO ENSURE THAT TOTAL PROBABILITY IS 1
C   NN = DEFINES MATRIX SIZE
C   STRT = DEFINES CENTER OF MATRIX
C   DLA = MATRIX OF CHOSEN LOCATIONS
C   NEIGHBORS = CHECKS NEIGHBORING POINTS FOR PARTICLES
C-----C

IMPLICIT REAL *8 (A-H,O-Z)
INTEGER STP,SEED2,NN,STRTY,DLA(99,99),BIGSTEP,II,JJ
INTEGER NEIGHBORS (99,99),I,J, MAXSTEP
INTEGER STRTX,SEED,IIP,IIM,JJP,JJM,FLAG,STPMAX
REAL *8 PROB(99,99),CHX,SUM, PROB2(99,99)
REAL *8 PROB3(99,99), TPI, ANG, R0
REAL *8 LOC1(99,99,4), LOC2(99,99,4)
OPEN(4,FILE='q4spot2.dat',STATUS='UNKNOWN')

C-----C
C   SET DEFAULT PARAMETERS
C-----C

ZERO = 0.0D0
ONE = 1.0D0
SUM = ZERO
NN = 99
STRTY = (NN+1)/2
STRTX = (NN+1)/2
SEED2 = 47
STP = 0
BIGSTEP = 0
MAXSTEP=4000
STPMAX=500
TPI=2*3.1415926536
R0=30.0

C-----C
C   INITIALIZE RANDOM NUMBER GENERATOR
C-----C

```

```
SEED=TIME()
CALL=RAND(SEED)
```

```
C-----C
C   SET LOCATION MATRICES   C
C-----C
```

```
DO 10 I=1,NN,+1
DO 11 J=1,NN,+1
DO 12 K=1,4,+1
    LOC1(I,J,K)=ZERO
    LOC2(I,J,K)=ZERO
12 CONTINUE
    DLA(I,J)=0
11 CONTINUE
10 CONTINUE
```

```
DO I=0,5,+1
    DO J=0,5,+1
        DLA(SEED2+I, SEED2+J)=1
    END DO
END DO
```

```
FLAG = 0
DO WHILE (FLAG.EQ.0)
    ANG=RAND(0)*TPI
    I = INT(R0*COS(ANG)) + 50
    J = INT(R0*SIN(ANG)) + 50
    IF (DLA(I,J).EQ.0) THEN
        LOC1(I,J,1) = ONE
        FLAG =1
    ENDIF
END DO
```

```
C-----C
C   BEGIN CALCULATIONS   C
C-----C
```

```
DO WHILE (BIGSTEP.LT.MAXSTEP)
```

```
C-----C
C   UPDATE NEIGHBORS MATRIX   C
C-----C
```

```
DO 807 I=1,NN,+1
DO 808 J=1,NN,+1
    NEIGHBORS(I,J)=0
    IF (DLA(I,J).GT.1) THEN
        DLA(I,J)=0
    END IF
808 END DO
807 END DO
```

```
DO 707 I=1,NN,+1
DO 708 J=1,NN,+1
    IF (DLA(I,J).NE.0) THEN
        IF (I.LT.NN) THEN
            IIP=I+1
        ELSE
            IIP=1
        ENDIF
```

```

      IF (J.LT.NN) THEN
        JJp=J+1
      ELSE
        JJp=1
      ENDIF
      IF (I.GT.1) THEN
        IIm=I-1
      ELSE
        IIm=99
      ENDIF
      IF (J.GT.1) THEN
        JJm=J-1
      ELSE
        JJm=99
      ENDIF
      NEIGHBORS(IIp,J) = 1
      NEIGHBORS(IIm,J) = 1
      NEIGHBORS(I,JJp) = 1
      NEIGHBORS(I,JJm) = 1
    ENDIF
708      ENDDO
707      ENDDO

      DO WHILE (STP.LT.STPMAX)

C-----C
C   CALCULATE MATRIX FOR PROBABILITY IN EACH LOCATION   C
C-----C
      DO 40 I=1,NN,+1
      DO 41 J=1,NN,+1
        PROB(I,J)=ABS(LOC1(I,J,1))*2+ABS(LOC1(I,J,2))*2+
1          ABS(LOC1(I,J,3))*2+ABS(LOC1(I,J,4))*2
        PROB(I,J)=PROB(I,J)*NEIGHBORS(I,J)*(1-DLA(I,J))
41      CONTINUE
40      CONTINUE

C-----C
C   Localize particle if there is a nonzero             C
C   probability on site(s) neighboring structure       C
C-----C
      SUM=ZERO

      DO 60 I=1,NN,+1
      DO 61 J=1,NN,+1
        PROB3(I,J)=PROB(I,J)*NEIGHBORS(I,J)*(1-DLA(I,J))
        PROB2(I,J)=ZERO
        IF (PROB3(I,J).NE.ZERO) THEN
          SUM=SUM+PROB3(I,J)
          PROB2(I,J)=SUM
        ENDIF
61      CONTINUE
60      CONTINUE

C-----C
C   CHOOSE LOCATION                                     C
C-----C
C   Use random number generator to get a random value.  C
C   The location chosen by the particle will be decided  C
C   by the ranges now defined in the matrix PROB. If the C
C   number is less than the value in a location, but    C
C   greater than the number in the previous location, it C
C   comes to rest in the current location.              C
C-----C

```

```

CHX=RAND(0)
FLAG = 0

DO 70 I=1,NN,+1
DO 71 J=1,NN,+1
  IF (CHX.LT.PROB2(I,J)) THEN
    II=I
    JJ=J
    FLAG=1
    GOTO 888
  ENDIF
71 CONTINUE
70 CONTINUE

888 CONTINUE
  IF (FLAG.EQ.1) THEN
    print *, II,JJ, SUM, R0, "FOUND"
    DLA (II,JJ) = 1
    STP=STPMAX
    R0=30.0+REAL(BIGSTEP*15.0)/MAXSTEP
C-----C
C   Particle keeps moving if it does   C
C   not meet an occupied location.    C
C-----C

    ELSE
      STP=STP+1
C-----C
C   MOVE WAVE FUNCTION                C
C-----C

      DO 23 I=1,NN,+1
      DO 24 J=1,NN,+1
      DO 25 K=1,4,+1
        LOC2(I,J,K)=ZERO
25 CONTINUE
24 CONTINUE
23 CONTINUE

      DO 20 I=1,NN,+1
      DO 21 J=1,NN,+1
      DO 22 K=1,4,+1
        LOC1(I,J,K)=LOC1(I,J,K)/SQRT((ONE-SUM))
        IF (PROB3(I,J).NE.ZERO) THEN
          LOC1(I,J,K)=ZERO
        ENDIF
        IF (LOC1(I,J,K).NE.ZERO) THEN
          IF (K.EQ.1.0D0) THEN
            LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
            LOC2(I,J,2)=LOC2(I,J,2)+LOC1(I,J,K)
            LOC2(I,J,3)=LOC2(I,J,3)+LOC1(I,J,K)
            LOC2(I,J,4)=LOC2(I,J,4)+LOC1(I,J,K)
          ELSEIF (K.EQ.2.0D0) THEN
            LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
            LOC2(I,J,2)=LOC2(I,J,2)-LOC1(I,J,K)
            LOC2(I,J,3)=LOC2(I,J,3)+LOC1(I,J,K)
            LOC2(I,J,4)=LOC2(I,J,4)-LOC1(I,J,K)
          ELSEIF (K.EQ.3.0D0) THEN
            LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
            LOC2(I,J,2)=LOC2(I,J,2)+LOC1(I,J,K)
            LOC2(I,J,3)=LOC2(I,J,3)-LOC1(I,J,K)
            LOC2(I,J,4)=LOC2(I,J,4)-LOC1(I,J,K)

```

```

                ELSE
                    LOC2(I,J,1)=LOC2(I,J,1)+LOC1(I,J,K)
                    LOC2(I,J,2)=LOC2(I,J,2)-LOC1(I,J,K)
                    LOC2(I,J,3)=LOC2(I,J,3)-LOC1(I,J,K)
                    LOC2(I,J,4)=LOC2(I,J,4)+LOC1(I,J,K)
                ENDIF
            ENDIF
22      CONTINUE
21      CONTINUE
20      CONTINUE

DO 93 I=1,NN,+1
DO 94 J=1,NN,+1
DO 95 K=1,4,+1
    LOC1(I,J,K)=LOC2(I,J,K)/(2.0D0)
    LOC2(I,J,K)=ZERO
95      CONTINUE
94      CONTINUE
93      CONTINUE

C-----C
C   F OPERATOR   C
C-----C

DO 30 I=1,NN,+1
DO 31 J=1,NN,+1
    IF (I.LT.NN) THEN
        IIp=I+1
    ELSE
        IIp=1
    ENDIF
    IF (J.LT.NN) THEN
        JJp=J+1
    ELSE
        JJp=1
    ENDIF
    IF (I.GT.1) THEN
        IIm=I-1
    ELSE
        IIm=99
    ENDIF
    IF (J.GT.1) THEN
        JJm=J-1
    ELSE
        JJm=99
    ENDIF
DO 32 K=1,4,+1
    IF (LOC1(I,J,K).NE.0.0D0) THEN
        LOC2(I,J,K)=ZERO
        IF (K.EQ.1) THEN
            LOC2(I,JJp,K)=LOC2(I,JJp,K)+LOC1(I,J,K)
        ENDIF
        IF (K.EQ.2) THEN
            LOC2(IIp,J,K)=LOC2(IIp,J,K)+LOC1(I,J,K)
        ENDIF
        IF (K.EQ.3) THEN
            LOC2(I,JJm,K)=LOC2(I,JJm,K)+LOC1(I,J,K)
        ENDIF
        IF (K.EQ.4) THEN
            LOC2(IIm,J,K)=LOC2(IIm,J,K)+LOC1(I,J,K)
        ENDIF
    ENDIF
ENDIF
32      CONTINUE
31      CONTINUE
30      CONTINUE

```

```

31          CONTINUE
30          CONTINUE

          DO 33 I=1,NN,+1
          DO 34 J=1,NN,+1
          DO 35 K=1,4,+1
              LOC1(I,J,K)=LOC2(I,J,K)
35          CONTINUE
34          CONTINUE
33          CONTINUE

          ENDIF
          ENDDO

          STP=0

          print *, BIGSTEP
          BIGSTEP=BIGSTEP+1

          DO 1080 I=1,NN,+1
          DO 1081 J=1,NN,+1
          DO 1082 K=1,4,+1
              LOC1(I,J,K)=ZERO
1082          CONTINUE
1081          CONTINUE
1080          CONTINUE

          FLAG = 0

          DO WHILE (FLAG.EQ.0)
              ANG=RAND(0)*TPI
              I = INT(R0*COS(ANG)) + 50
              J = INT(R0*SIN(ANG)) + 50
              IF (DLA(I,J).EQ.0) THEN
                  LOC1(I,J,1) = ONE
                  FLAG =1
              ENDIF
          ENDDO
          ENDDO

C-----C
C  WRITE IN DATA FILE  C
C-----C

          DO 90 I=1,NN,+1
          DO 91 J=1,NN,+1
              IF (DLA(I,J).NE.0) THEN
                  WRITE(4,5) I,J
              ENDIF
91          CONTINUE
90          CONTINUE

          CLOSE(4)

          STOP

1  FORMAT(' ',1X,F12.4,' ',1X,I5,' ',1X,F9.6)
2  FORMAT(' ',6X,'X',' ',6X,'Y',' ',4X,'VALUE')
3  FORMAT(' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
1    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
2    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
3    ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,

```

```

4      ' ',1x,F9.6,' ',1x,F9.6,' ',1x,F9.6,' ',1x,F9.6,
5      ' ',1x,F9.6,' ',1x,F9.6,' ',1x,F9.6,' ',1x,F9.6,
6      ' ',1x,F9.6,' ',1x,F9.6,' ',1x,F9.6,' ',1x,F9.6,
7      ' ',1x,F9.6,' ',1x,F9.6,' ',1x,F9.6,' ',1x,F9.6,
8      ' ',1x,F9.6,' ',1x,F9.6,' ',1x,F9.6)
4      FORMAT(' ',I5,' ',I5,' ',I5,' ',I5,' ',I5,
1      ' ',I5,' ',I5,' ',I5,' ',I5,' ',I5,
2      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
3      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
4      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
5      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
6      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5)
5      FORMAT(' ',1X,I9,' ',1X,I9)
      END

```


Second Classical Model Program Code

```

PROGRAM classic3

C-----C
C
C   PROJECTION
C
C   This program aims to simulate random movement
C   of a particle based upon quantum mechanics and
C   take into account the internal states of the particles.
C   Using the quantum model, it aims to simulate a classical
C   random walk by selecting a location after iteration.
C-----C
C
C   PARAMETERS
C
C   LOC = LOCATION MATRIX
C   STP = STEP NUMBER
C   PROB = PROBABILITY MATRIX
C   CHX = CHOSEN LOCATION
C   SUM = CHECK TO ENSURE THAT TOTAL PROBABILITY IS 1
C   NN = DEFINES MATRIX SIZE
C   STRT = DEFINES CENTER OF MATRIX
C   DLA = MATRIX OF CHOSEN LOCATIONS
C   NEIGHBORS = CHECKS NEIGHBORS LOCATIONS FOR PARTICLES
C-----C

IMPLICIT REAL *8 (A-H,O-Z)
INTEGER STP,SEED2,NN,STRTY,DLA(99,99),BIGSTEP,II,JJ
INTEGER NEIGHBORS (99,99),I,J, MAXBIG
INTEGER STRTX,SEED,Iip,IIm,JJp,JJm,FLAG,STPMAX
REAL *8 PROB(99,99),CHX,SUM,PROBOLD(99,99), PROB2(99,99)
REAL *8 PROB3(99,99), PI, AN, R0
OPEN(4,FILE='c3spot2.dat',STATUS='UNKNOWN')

C-----C
C   SET DEFAULT PARAMETERS
C-----C

ZERO = 0.0D0
ONE = 1.0D0
SUM = ZERO
NN = 99
STRTY = (NN+1)/2
STRTX = (NN+1)/2
SEED2 = 47
STP = 0
BIGSTEP = 0
STPMAX=500
MAXBIG=4000
TPI=2*3.1415926536
R0=30.0

C-----C
C   INITIALIZE RANDOM NUMBER GENERATOR
C-----C

SEED=TIME()
CALL=RAND(SEED)

```

```

C-----C
C   SET PROBABILITY MATRICE   C
C-----C

```

```

      DO 10 I=1,NN,+1
      DO 11 J=1,NN,+1
        DLA(I,J)=0
        PROB(I,J)=ZERO
11    CONTINUE
10    CONTINUE

```

```

      DO I=0,5,+1
      DO J=0,5,+1
        DLA(SEED2+I, SEED2+J)=1
      ENDDO
      ENDDO

```

```

      FLAG = 0

```

```

      DO WHILE (FLAG.EQ.0)
        ANG=Rand(0)*TPI
        I = INT(R0*COS(ANG)) + 50
        J = INT(R0*SIN(ANG)) + 50
        IF (DLA(I,J).EQ.0) THEN
          PROB(I,J) = ONE
          FLAG =1
        ENDIF
      ENDDO

```

```

C-----C
C   BEGIN CALCULATIONS       C
C-----C

```

```

      DO WHILE (BIGSTEP.LT.MAXBIG)

```

```

C-----C
C   Update Neighbors matrix   C
C-----C

```

```

      DO 807 I=1,NN,+1
      DO 808 J=1,NN,+1
        NEIGHBORS(I,J)=0
        IF (DLA(I,J).GT.1) THEN
          DLA(I,J)=0
        END IF
808    ENDDO
807    ENDDO

```

```

      DO 707 I=1,NN,+1
      DO 708 J=1,NN,+1
        IF (DLA(I,J).NE.0) THEN
          IF (I.LT.NN) THEN
            IIp=I+1
          ELSE
            IIp=1
          ENDIF
          IF (J.LT.NN) THEN
            JJp=J+1
          ELSE
            JJp=1
          ENDIF

```

```

        IF (I.GT.1) THEN
            IIm=I-1
        ELSE
            IIm=99
        ENDIF
        IF (J.GT.1) THEN
            JJm=J-1
        ELSE
            JJm=99
        ENDIF
        NEIGHBORS(IIp,J) = 1
        NEIGHBORS(IIm,J) = 1
        NEIGHBORS(I,JJp) = 1
        NEIGHBORS(I,JJm) = 1
    ENDIF
708     ENDDO
707     ENDDO

    DO WHILE (STP.LT.STPMAX)

        DO 23 I=1,NN,+1
        DO 24 J=1,NN,+1
            PROBOld(I,J)=PROB(I,J)
            PROB(I,J)=ZERO
24         CONTINUE
23         CONTINUE

C-----C
C    Localize particle is there is a nonzero      C
C    probability on site(s) neighboring structure  C
C-----C

        SUM=ZERO

        DO 60 I=1,NN,+1
        DO 61 J=1,NN,+1
            PROB3(I,J)=PROBOld(I,J)*NEIGHBORS(I,J)*(1-DLA(I,J))
            PROB2(I,J)=ZERO
            IF (PROB3(I,J).NE.ZERO) THEN
                SUM=SUM+PROB3(I,J)
                PROB2(I,J)=SUM
            ENDIF
61         CONTINUE
60         CONTINUE

C-----C
C    CHOOSE LOCATION                               C
C-----C
C    Use random number generator to get a random value.      C
C    The location chosen by the particle will be decided      C
C    by the ranges now defined in the matrix PROB. If the    C
C    number is less than the value in a location, but        C
C    greater than the number in the previous location, it     C
C    comes to rest in the current location.                   C
C-----C

        CHX=RAND(0)
        FLAG = 0

        DO 70 I=1,NN,+1
        DO 71 J=1,NN,+1
            IF (CHX.LT.PROB2(I,J)) THEN
                II=I

```

```

        JJ=J
        FLAG=1
        GOTO 888
    ENDIF
71      CONTINUE
70      CONTINUE

888     CONTINUE

    IF (FLAG.EQ.1) THEN
        print *, II,JJ, SUM, R0, "FOUND"
        DLA (II,JJ) = 1
        STP=STPMAX
        R0=30.+REAL(BIGSTEP*15.0)/MAXBIG

C-----C
C      Particle keeps moving if it does      C
C      not meet an occupied location.        C
C-----C

        ELSE
            STP=STP+1

C-----C
C      Probability bounces off structure      C
C-----C

        DO 30 I=1,NN,+1
        DO 31 J=1,NN,+1
            PROBOLD(I,J)=PROBOLD(I,J)/(ONE-SUM)
            IF (PROB3(I,J).NE.2ZERO) THEN
                PROBOLD(I,J)=2ZERO
            ENDIF
            IF (PROBOLD(I,J).NE.2ZERO) THEN
                IF (I.LT.NN) THEN
                    IIP=I+1
                ELSE
                    IIP=1
                ENDIF
                IF (J.LT.NN) THEN
                    JJP=J+1
                ELSE
                    JJP=1
                ENDIF
                IF (I.GT.1) THEN
                    IIM=I-1
                ELSE
                    IIM=99
                ENDIF
                IF (J.GT.1) THEN
                    JJM=J-1
                ELSE
                    JJM=99
                ENDIF
                PROB(I,JJP)=PROB(I,JJP)+PROBOLD(I,J)/4
                PROB(I,JJM)=PROB(I,JJM)+PROBOLD(I,J)/4
                PROB(IIP,J)=PROB(IIP,J)+PROBOLD(I,J)/4
                PROB(IIM,J)=PROB(IIM,J)+PROBOLD(I,J)/4
            ENDIF
31      CONTINUE
30      CONTINUE
        ENDIF
    ENDDO

```

```

      STP=0

      print *, BIGSTEP
      BIGSTEP=BIGSTEP+1

      DO 1080 I=1,NN,+1
      DO 1081 J=1,NN,+1
        PROB(I,J)=ZERO
1081    CONTINUE
1080    CONTINUE

      FLAG = 0
      DO WHILE (FLAG.EQ.0)
        ANG=Rand(0)*TPI
        I = INT(R0*COS(ANG)) + 50
        J = INT(R0*SIN(ANG)) + 50
        IF (DLA(I,J).EQ.0) THEN
          PROB(I,J) = ONE
          FLAG =1
        ENDIF
      ENDDO

      ENDDO

C-----C
C  WRITE IN DATA FILE  C
C-----C

      DO 90 I=1,NN,+1
      DO 91 J=1,NN,+1
        IF (DLA(I,J).NE.0) THEN
          WRITE(4,5) I,J
        ENDIF
91    CONTINUE
90    CONTINUE

      CLOSE(4)

      STOP

1    FORMAT(' ',1X,F12.4,' ',1X,I5,' ',1X,F9.6)
2    FORMAT(' ',6X,'X',' ',6X,'Y',' ',4X,'VALUE')
3    FORMAT(' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
1      ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
2      ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
3      ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
4      ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
5      ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
6      ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
7      ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,
8      ' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6,' ',1X,F9.6)
4    FORMAT(' ',I5,' ',I5,' ',I5,' ',I5,' ',I5,
1      ' ',I5,' ',I5,' ',I5,' ',I5,' ',I5,
2      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
3      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
4      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
5      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,
6      ' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5,' ',1X,I5)
5    FORMAT(' ',1X,I9,' ',1X,I9)

      END

```

References

- [1] Aharonov, Y, L Davidovich, and N Zagury. "Quantum Random Walks." Rphysical Review A 48 (1993): 1687-1690.
- [2] Blanchard, Ph, and M O. Hongler. "Quantum Random Walks and Piecewise Deterministic Evolutions." Physical Review Letters 92 (2004).
- [3] Brun, Todd A., H A. Carteret, and Andris Ambainis. "Quantum Random Walks with Decoherent Coins." Physical Review A 67 (2003).
- [4] Brun, Todd A., Hilary A. Carteret, and Andris Ambainis. "Quantum to Classical Transition for Random Walks." Physical Review Letters 91 (2003).
- [5] Du, Jiangfeng, Hui Li, Xiaodong Xu, Mingjun Shi, Jihui Wu, Xianyi Zhou, and Rongdian Han. "Experimental Implementation of the Quantum Random-Walk Algorithm." Physical Review A 67 (2003).
- [6] Jeong, H, M Paternostro, and M S. Kim. "Simulation of Quantum Random Walks Using Interference of a Classical Field." Physical Review A 69 (2004).
- [7] Kempe, J. "Quantum Random Walks: an Introductory Overview." Contemporary Physics 44 (2003): 307-327.
- [8] Liboff, Richard L. Introductory Quantum Mechanics. 4th ed. Patparganj, India: Pearson Education, Inc., 2003.
- [9] Mackay, T D., S D. Bartlett, L T. Stephenson, and B C. Sanders. "Quantum Walks in Higher Dimensions." Journal of Physics a: Mathematical and General 35 (2002): 2745-2753
- [10] Travaglione, B C., and G J. Milburn. "Implementing the Quantum Random Walk." Physical Review A 65 (2002).
- [11] Weickert, Brendan. "Infinite-Dimensional Complex Dynamics: a Quantum Random Walk." Discrete and Continuous Dynamical Systems 7 (2001): 517-524.
- [12] <http://www.gut-wirtz.de/dla/>
- [13] <http://astronomy.swin.edu.au/~pbourke/fractals/dla/>
- [14] <http://apricot.polyu.edu.hk/~lam/dla/dla.html>