THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Master of Science (Physics)

DEGREE

Physics	Physics
FIELD	**DEPARTMENT**

TITLE: Dynamical Simulation of Electron Hoping in Double Quantum Rings

NAME: Mr. Chonlathep Kitsinthopchai

THIS THESIS HAS BEEN ACCEPTED BY

THESIS ADVISOR

(Assistant Professor Sutee Boonchui, Ph.D.)

DEPARTMENT HEAD

(Mr. Wiwat Wongkokua, Ph.D.)

APPROVED BY THE GRADUATE SCHOOL ON _____

DEAN

(Associate Professor Gunjana Theeragool, D.Agr.)

THESIS

DYNAMICAL SIMULATION OF ELECTRON HOPING IN DOUBLE QUANTUM RINGS

CHONLATHEP KITSINTHOPCHAI

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Science (Physics)
Graduate School, Kasetsart University
2013

Chonlathep Kitsinthopchai 2013: Dynamical Simulation of Electron Hoping in Double Quantum Rings. Master of Science (Physics), Major Field: Physics, Department of Physics. Thesis Advisor: Assistant Professor Sutee Boonchui, Ph.D. 43 pages.

In this Thesis, we study effects of The wave packet of electron is hoping through quantum rings under the static magnetic field, can be solved by transformation to the canonical form of certain system of differential equation. The Lorentz force leads to electron asymmetry which enhances the electron passing through a quantum ring while the Aharanov-Bohm effect (AB effect) reduces the probability of transmission by phase shifted interference. For zero or similar magnetic field of both rings, the wave packet can pass both quantum rings to the exit quantum wire while different magnetic field of both rings prevent the second ring's injection of electron.

Student's signature Thesis Advisor's signature ___ / ___ / ___

ACKNOWLEDGEMENT

I wish to grateful thank and deeply indebted to my advisor Asst. Prof. Dr. Sutee Boonchui, who has provided me with continuous encouragement and support, as well as meticulous criticism of this work. Without him this thesis would not have been possible.

Also, I would like to thank the Graduate School, Kasetsart university and the Development and Promotion of the gifted in science and technology project (DPST) for partial supported.

Chonlathep Kitsinthopchai

May 2013

TABLE OF CONTENTS

Page

LIST OF FIGURES

DYNAMICAL SIMULATION OF ELECTRON HOPING IN DOUBLE QUANTUM RINGS

INTRODUCTION

The quantum dot is a portion of matter that confined electron in spatial three dimensions in 1-20 nanometer scale of diameter. Consequently, giving optical and electronic properties intermediate between semiconductors and discrete molecules. Sorting of many quantum dots give new material structure such as quantum wire, quantum rings. The quantum dot research and development are going to explore innovation of material property.

The quantum rings is produced by quantum dots in two dimensions plane circular pattern. The electrons are hopping through between quantum dots that generate the current in system. The wave function of an electron passing the quantum rings under magnetic field can be controlled the arms of passing or denied it by interference. In addition the Aharonov-Bohm effect give the explanation of occurring interference by phase shift of both arm. Moreover the effect of Lorentz force, which bend electron to aside, is involved. Transport properties of quantum wires, dots, and wells attract much attention because of rapid development of nanotechnology. In these structures the magnetic field can tune the phase of the electronic wave function.

The system of quantum rings can be solved by numerical method. Applied magnetic field strict the wave packet to arbitrary condition. The study will perform advantage of controlling electron by magnetic field such as transistor. For this reason the simulating of double quantum rings is remarkable.

In the reality, in experiments on very small rings of gold with circumference about 100-500 nm and width about 40 nm no strictly periodic behavior of any kind was observed. Basically magnetoresistance reveals the contribution of the aperiodic fluctuations. The detailed structure of the peak in the power spectrum is, as it was suggested by Stone in 1985, the result of mixing of the field scales corresponding to

the area of the hole in the ring and the area of the arms of the ring. Actually, with decreasing of the aspect ratio *d/L*, where *d* is a width of the arms and *L* is a size of the ring, the contribution of the aperiodic fluctuations increases. The first numerical analysis of quantum fluctuations of the magnetoresistence of the two-dimensional strip in the frame work of the hopping model with random site energies was performed by Stone. It was shown that the stationary fluctuations of the magnetotransport of electrons are a direct consequence of the microscopic quantum states in specific samples. These fluctuations enhance much if the states become localized (Webb *et al*, 1985).

In 1997, Pichugin and Sadreev solve the Schrodinger equation numerically in the framework of the two-dimensional tight-binding model to find the Aharanov-Bohm Oscillations of the conductance and detailed current flow distributions. However, they are restricted by the field region where one flux quantum per lattice plaquette is much less than unity, which makes the continual Schrodinger equation certainly applicable. The current flow patterns display rich vortex structures and show that the formation of the convectional vortex flow patterns is directly related to the complexity in flux dependence of the transmission. They show that the laminar flow of the electron takes place only in the very restricted case of the rings with small and moderate aspect ratios, with the single-channel transport, and with zero flux. Application of the flux gives rise to the current vortex near the entrance of the ring. This vortex mixes inner and outer paths of the electron transport in such a way that phase shifts induced by different fluxes enclosed by different trajectories of the electron in the ring become equaled. As a result they can observe the quasiregular Aharanov-Bohm Oscillations of the transmission in rings with small aspect ratios. With an increasing in the aspect ratio and the number of channels the Aharanov-Bohm Oscillations of the transmission become irregular and, correspondingly, the current flow patterns acquire a rather volatile form with complex distribution of vortices (Pichugin and Sadreev, 1997).

In 2000, Pedersen *et al* published the observation of quantum asymmetry in an Aharonov-Bohm ring. They have investigated the Aharonov-Bohm effect in a one-

dimensional GaAs/ $Ga_{0.7}Al_{0.3}$ as ring at low-magnetic fields. The oscillatory magnetoconductance of these systems is systematically studied as a function of density. They observe phase shifts in the magnetoconductance oscillations, and halving of the fundamental *h/e* period, as the density is varied. Theoretically we find agreement with the experiment, by introducing an asymmetry between the two arms of the ring (Pedersen, 2000).

In 2005, Szafran and Peeters published the simulations of electron transport through a quantum ring in the effect of the Lorentz force. The research describe the effect of the Lorentz-force-related deformation of the electron trajectories on the Aharonov-Bohm effect in a semiconductor quantum ring (Szafan and Peeters, 2005).

OBJECTIVES

1. To simulate flow characteristics of the electron wave packet through the double quantum rings.

2. To study effect of static magnetic field to the electron in the double quantum rings.

LITERATURE REVIEW

1. The Aharonov-Bohm Effect

When the beam of electrons is split in two and passed either side of a long solenoid before being recombined. The beams are kept well away from the solenoid itself, so they encounter only regions where $\mathbf{B} = 0$ but $\mathbf{A}$ is not zero. The two beams arrive with different phases (Aharonov and Bohm, 1959).

$$g = \frac{q}{\hbar}\int \mathbf{A}\cdot d\mathbf{r} = \frac{q\Phi}{2\pi\hbar}\int\left(\frac{1}{r}\hat{\phi}\right)\cdot\left(r\hat{\phi}d\phi\right) = \pm\frac{q\Phi}{2\hbar} \tag{1}$$

The plus sign applies to the electrons traveling in the same direction as $\mathbf{A}$. The beams arrive out of phase by an amount proportional to the magnetic flux their paths encircle.

$$\text{phase difference} = \frac{q\Phi}{\hbar} \tag{2}$$

This phase shift leads to measurable interference and has been confirmed experimentally (Chambers, 1960).

2. Observation Aharonov-Bohm ring

Electron wave packets circling a magnetic flux should exhibit the phase shift introduced by the magnetic vector potential (Aharonov and Bohm, 1959). In a metallic ring, small enough so that the electron states are not randomized by magnetic scattering during the traversal of the arm of the ring, an interference pattern should be present in the magnetoresistance of the device.

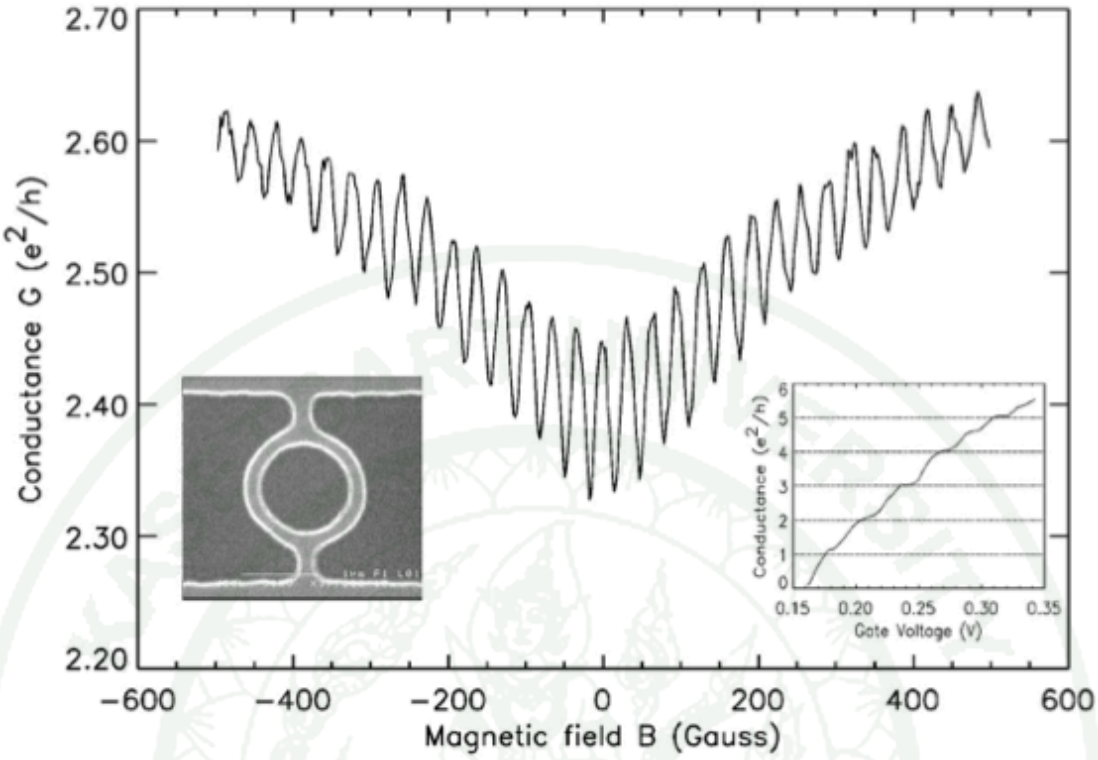

Figure 1 Measured magnetoconductance of the device

In Figure 1 presents a measurement of the magnetoconductance of the device displayed in the left insert (Pedersen *et al*, 2000). The magnetoconductance show large Aharonov-Bohm oscillations. Due to the long distance between the voltage probes, the measurement is an effective two-terminal measurement; hence the Aharonov-Bohm magnetoconductance is as observed forced to be symmetrical as a consequence of the Onsager relations. While the right inset in Figure 1 displays the conductance as function of gate voltage at T=4.2K.

3. Single quantum ring simulation

The time-dependent Schrodinger equation for an electron passing through a semiconductor quantum ring is solved in the presence of a perpendicular homogeneous magnetic field with the effects of the Lorentz force on the Aharonov-Bohm oscillations (Szafran and Peeters, 2005).

The electron confined in the (x,y) plane with perpendicular magnetic field. The Hamiltonian has the form

$$H = \frac{1}{2m}\left(-i\hbar\nabla + e\mathbf{A}\right)^2 \tag{3}$$

where m stands for the electron effective mass ($0.067m_0$) and $\mathbf{A}$ is the vector potential (Pichugin and Sadreev, 1997). The wave function is expanded in a basis of Gaussian functions centered on chosen point $\mathbf{R}_n = (X_n, Y_n)$

$$\Psi(x,y,t) = \sum_n c_n(t) f_n(x,y) \tag{4}$$

with

$$f_n(x,y) = \frac{1}{\lambda\sqrt{\pi}} \exp\left[\frac{(\vec{r} - \vec{R}_n)^2}{2\lambda^2} + \frac{ieB(x - X_n)(y + Y_n)}{2\hbar}\right] \tag{5}$$

The studied magnetic field range the increase of the electron localization is negligible. Substituting expansion Eq.(3) into the time-dependent Schrodinger equation obtain a system of linear equations for the time derivative of coefficients $c_n(t)$,

$$\mathbf{S}\dot{\mathbf{c}}(t) = \mathbf{H}\mathbf{c}(t) \tag{6}$$

which solved by $\mathbf{S}\mathbf{c}(t + dt) = \mathbf{S}\mathbf{c}(t - dt) - 2idt\mathbf{H}\mathbf{c}(t)/\hbar$, where the elements of overlap and Hamiltonian matrices are given by $\mathbf{S}_{kn} = \langle f_k | f_n \rangle$ and $\mathbf{H}_{kn} = \langle f_k | H | f_n \rangle$.

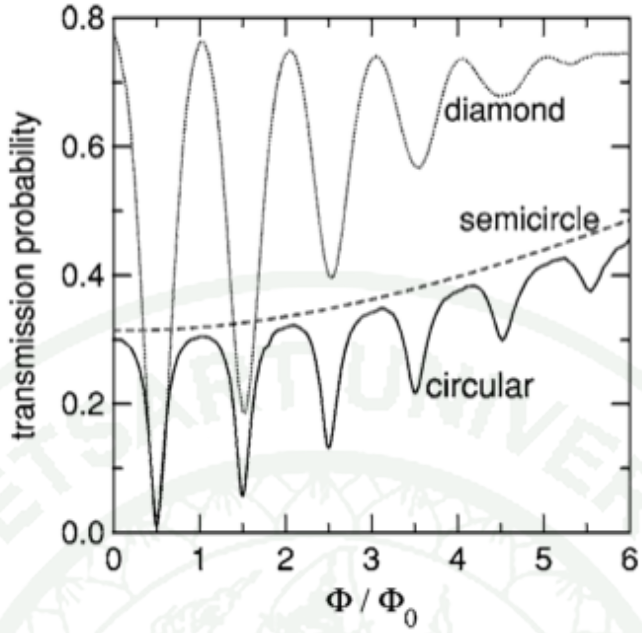

Figure 2 The transmission probability of the wave packet in differ pattern

In Figure 2, the solid line shows the transmission probability of the wave packet through the circular quantum ring (Szafran and Peeters, 2005). This quantity was obtained by integrating the probability density leaving the ring through the upper lead. The decreasing amplitude is due to the growing imbalance in the amount of charge transferred through the left and right arms of the ring, which prevents the interference from being completely destructive. The values of the transmission probability maxima and minima are increasing functions of the magnetic field, which is a consequence of the guiding behavior of the Lorentz force that eases the entrance and exit of the wave packet. The envelope of the maxima is well approximated by the packet transfer probability through a semicircular wire that is obtained when the right arm of the circular ring is removed, plotted with the dashed line in Figure 2.

MATERIALS AND METHODS

Materials

1. Computer (Intel Core 2 Duo, Ram 4GB)
2. MATLAB (Software)

Methods

1. Theory

1.1 Unitary Transformation

The solving Eq.(6) with iteration method had been taking for a long time. However there is another way for solve this problem by transform the system of equation to Canonical form which is faster than one. The Eq.(6) can be optimized by

$$\dot{\mathbf{c}}(t) = \mathbf{M}\mathbf{c}(t) \tag{7}$$

where $\mathbf{M} = \mathbf{S}^{-1}\mathbf{H}$, because of $\mathbf{S}$ and $\mathbf{H}$ are Hermitian matrix and the inverse of an invertible Hermitian matrix. Hence $\mathbf{M}$ is Hermitian matrix, which could be diagonalized by a unitary transformation,

$$\mathbf{D} = \mathbf{T}^{-1}\mathbf{M}\mathbf{T} \tag{8}$$

Here $\mathbf{D}$ is diagonal and $\mathbf{T}$ is transformation matrix which be found by solve eigenvalues and eigenvector of $\mathbf{M}$. Eq.(7) becomes

$$\mathbf{T}^{-1}\dot{\mathbf{c}}(t) = \mathbf{D}\mathbf{T}^{-1}\mathbf{c}(t) \tag{9}$$

Since is a time-independent matrix, we can rewrite Eq.(9) in the form

$$\dot{\mathbf{C}}(t) = \mathbf{D}\mathbf{C}(t) \tag{10}$$

The matrix Eq.(10) is a system of differential equation with the solution

$$\mathbf{C}(t) = \mathbf{C}(0)\exp(\mathbf{D}t) \tag{11}$$

Now the probability amplitude $\mathbf{c}(t)$ is given by the transformation

$$\mathbf{c}(t) = \mathbf{T}\mathbf{C}(t) \tag{12}$$

1.2 Element of S-Matrix

From $\mathbf{S}_{kn} = \langle f_k | f_n \rangle$ are the element of S-Matrix at row kth and column nth defined by

$$\langle f_k | f_n \rangle = \iint f_k^* f_n \, dx\, dy \tag{13}$$

Since Eq.(5), These elements are

$$\begin{aligned}\langle f_k | f_n \rangle = \frac{1}{\lambda^2 \pi} \iint \exp\left[\frac{(\vec{r} - \vec{R}_k)^2}{2\lambda^2} - \frac{ieB(x - X_k)(y + Y_k)}{2\hbar} \right] \\ \times \exp\left[\frac{(\vec{r} - \vec{R}_n)^2}{2\lambda^2} + \frac{ieB(x - X_n)(y + Y_n)}{2\hbar} \right] dxdy\end{aligned} \tag{14}$$

$$\begin{aligned}\langle f_k | f_n \rangle = \frac{1}{\lambda^2 \pi} \iint \exp\left[\frac{(x - X_n)^2 + (x - Y_n)^2 + (x - X_k)^2 + (x - Y_k)^2}{2\lambda^2} \right. \\ \left. + \frac{ieB\{(x - X_n)(y + Y_n) - (x - X_k)(y + Y_k)\}}{2\hbar} \right] dxdy\end{aligned} \tag{15}$$

1.3 Element of H-Matrix

From $\mathbf{H}_{kn} = \langle f_k | H | f_n \rangle$ are the element of H-Matrix at row k[th] and column n[th] defined by

$$\langle f_k | H | f_n \rangle = \frac{1}{2m} \iint f_k^* \left(-i\hbar\nabla + e\mathbf{A} \right)^2 f_n \, dx \, dy \tag{16}$$

where $\mathbf{A} = -By\hat{i}$ is the vector potential of magnetic field in z-direction. Then solve operator to

$$\left(-i\hbar\nabla + e\mathbf{A} \right)^2 = -\hbar^2\nabla^2 + 2i\hbar eBy\frac{\partial}{\partial x} + e^2B^2y^2 \tag{17}$$

Consider first term

$$-\hbar^2\nabla^2 f_n = \frac{-\hbar^2}{\lambda\sqrt{\pi}}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right)\exp\left[\frac{(\vec{r} - \vec{R}_n)^2}{2\lambda^2} + \frac{ieB(x - X_n)(y + Y_n)}{2\hbar} \right] \tag{18}$$

The 1[st] x-derivative of f_n

$$\frac{\partial}{\partial x} f_n = \frac{1}{\lambda\sqrt{\pi}}\left(-\frac{(x - X_n)}{\lambda^2} + \frac{ieB(y + Y_n)}{2\hbar} \right) f_n \tag{19}$$

The 2[nd] x-derivative of f_n

$$\begin{aligned} \frac{\partial^2}{\partial x^2} f_n &= \frac{1}{\lambda\sqrt{\pi}}\left(-\frac{1}{\lambda^2} f_n \right) + \left(-\frac{(x - X_n)}{\lambda^2} + \frac{ieB(y + Y_n)}{2\hbar} \right)\frac{\partial f_n}{\partial x} \\ &= \frac{1}{\lambda\sqrt{\pi}}\left\{ -\frac{1}{\lambda^2} + \left(-\frac{(x - X_n)}{\lambda^2} + \frac{ieB(y + Y_n)}{2\hbar} \right)^2 \right\} f_n \end{aligned} \tag{20}$$

The 1st y-derivative of f_n

$$\frac{\partial}{\partial y} f_n = \frac{1}{\lambda\sqrt{\pi}}\left(-\frac{y-Y_n}{\lambda^2} + \frac{ieB(x-X_n)}{2\hbar}\right) f_n \tag{21}$$

The 2nd y-derivative of f_n

$$\frac{\partial^2}{\partial y^2} = \frac{1}{\lambda\sqrt{\pi}}\left\{-\frac{1}{\lambda^2} + \left(-\frac{y-Y_n}{\lambda^2} + \frac{ieB(x-X_n)}{2\hbar}\right)^2\right\} f_n \tag{22}$$

Then Eq.(18) becomes

$$\frac{\hbar^2}{\lambda\sqrt{\pi}}\left\{\frac{2}{\lambda^2} + \left(\frac{(x-X_n)}{\lambda^2} - \frac{ieB(y+Y_n)}{2\hbar}\right)^2 + \left(\frac{(y-Y_n)}{\lambda^2} - \frac{ieB(x-X_n)}{2\hbar}\right)^2\right\} f_n \tag{23}$$

Hence the second term of Eq.(17) becomes

$$2i\hbar eBy\frac{\partial}{\partial x} f_n = \frac{2i\hbar eBy}{\lambda\sqrt{\pi}}\left(-\frac{(x-X_n)}{\lambda^2} + \frac{ieB(y+Y_n)}{2\hbar}\right) f_n \tag{24}$$

From Eq.(23) and Eq.(24), Then Eq. (17) becomes

$$\begin{aligned}\left(-i\hbar\nabla + e\mathbf{A}\right)^2 f_n = \frac{1}{\lambda\sqrt{\pi}}\Bigg\{&\frac{2\hbar^2}{\lambda^2} + e^2B^2y^2 \\ &+\hbar^2\left(\frac{(x-X_n)}{\lambda^2} - \frac{ieB(y+Y_n)}{2\hbar}\right)^2 \\ &+\hbar^2\left(\frac{(y-Y_n)}{\lambda^2} - \frac{ieB(x-X_n)}{2\hbar}\right)^2 \\ &+2i\hbar eBy\left(-\frac{(x-X_n)}{\lambda^2} + \frac{ieB(y+Y_n)}{2\hbar}\right)\Bigg\} f_n\end{aligned} \tag{25}$$

The elements of H-Matrix are

$$\begin{aligned}\langle f_k|H|f_n\rangle = \frac{1}{2m\lambda^2\pi}\iint\Bigg\{&\frac{2\hbar^2}{\lambda^2}+e^2B^2y^2\\&+\hbar^2\left(\frac{(x-X_n)}{\lambda^2}-\frac{ieB(y+Y_n)}{2\hbar}\right)^2\\&+\hbar^2\left(\frac{(y-Y_n)}{\lambda^2}-\frac{ieB(x-X_n)}{2\hbar}\right)^2\\&+2i\hbar eBy\left(-\frac{(x-X_n)}{\lambda^2}+\frac{ieB(y+Y_n)}{2\hbar}\right)\Bigg\}\\&\times\exp\Bigg[\frac{(x-X_n)^2+(x-Y_n)^2+(x-X_k)^2+(x-Y_k)^2}{2\lambda^2}\\&+\frac{ieB\{(x-X_n)(y+Y_n)-(x-X_k)(y+Y_k)\}}{2\hbar}\Bigg]dxdy\end{aligned} \tag{26}$$

1.4 Initial condition

The initial condition is the incident wave packet at nth dot defined by

$$\Psi(x,y,0)=f_n(x,y)e^{iqy} \tag{27}$$

where q is momentum 0.05/nm. The initial coefficient can be found by

$$\begin{aligned}|\Psi(0)\rangle &= e^{iqy}|f_n\rangle\\ \sum_k|f_k\rangle\langle f_k|\Psi(0)\rangle &= \sum_k|f_k\rangle\langle f_k|e^{iqy}|f_n\rangle\\ \sum_k\langle f_k|\Psi(0)\rangle|f_k\rangle &= \sum_k\langle f_k|e^{iqy}|f_n\rangle|f_k\rangle\\ \langle f_k|\Psi(0)\rangle &= \langle f_k|e^{iqy}|f_n\rangle\end{aligned} \tag{28}$$

Because $c_k(0)=\langle f_k|\Psi(0)\rangle=\langle f_k|e^{iqy}|f_n\rangle$ then the element kth of coefficient matrix are

$$c_k(0)=\iint f_k^* f_n e^{iqy}\,dx\,dy \tag{29}$$

2 Programming

2.1 Introduction

The simulation was developed on MATLAB software. In the case of 1,000 quantum dots use time about 1-15 minutes for calculation all matrices upon applied magnetic field. There are three main processes in the system “Setup parameters”, “Build matrices” and “Get result”. All source code are showed in the appendix.

Figure 3 Stated diagram of program

2.2 Setting parameters

All parameters, position of quantum dots and magnetic field can be setup in three following script files.

2.2.1 Script file “parameter.m”

All constant parameter and main magnet field are setup in this file. The main magnetic field is magnetic field of the system but still not present in the quantum dots because the system needs to control magnetic field in every quantum dots.

2.2.2 Script file “ position.m”

The properties of quantum dots are configure in this files such as position of all quantum dots in the system which collected in array named “positionQuantumDots”. The number of quantum dots and the momentum in the Eq.(27) are parameter named “initialQuantumDots” and “initialMomentum”, Consequently.

2.2.3 Script file “setMagneticField.m”

The magnetic field of every quantum dots, can be adjusted in this files, are collected in array which index of array is the number of quantum dots.

2.3 Preparation matrices

The coefficient matrices need to calculate from S-Matrix, H-Matrix and T-Matrix which is the longest time of this program. So after input all parameters the program can be start calculate by script file name “start.m”. This one will run all parameter script files and then start build all matrices.

S-Matrix is the first matrix which be calculated will take a time for 40% of all running time same as the secondary calculated H-matrix. Then T-matrix which is third calculation by a function of MATLAB called “eig” for solved eigenvalue and eigenvector of matrix. The final is the “initialCoefficientMatrix” the coefficient list at starting time.

2.4 Get results

This state is waiting state of the program for next order will have finished then show the results and the program become back to waiting state. The result are many type which upon a next step running script. This research present probability of the wave packet through the ring in many cases.

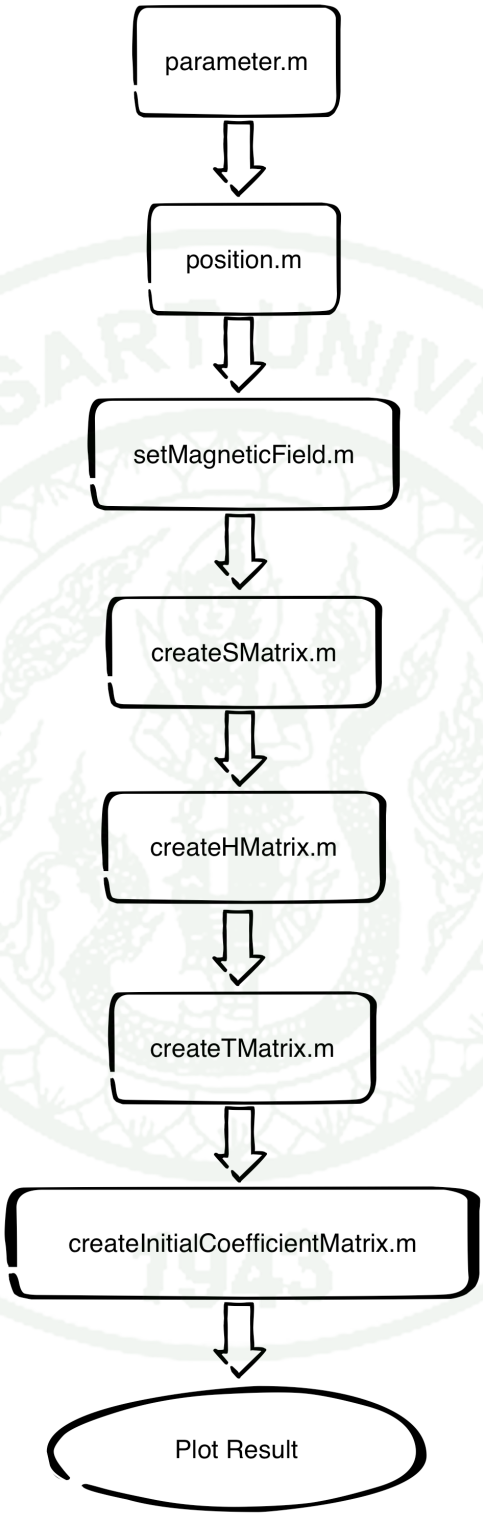

Figure 4 Sequence of script files

RESULTS AND DISCUSSION

Results

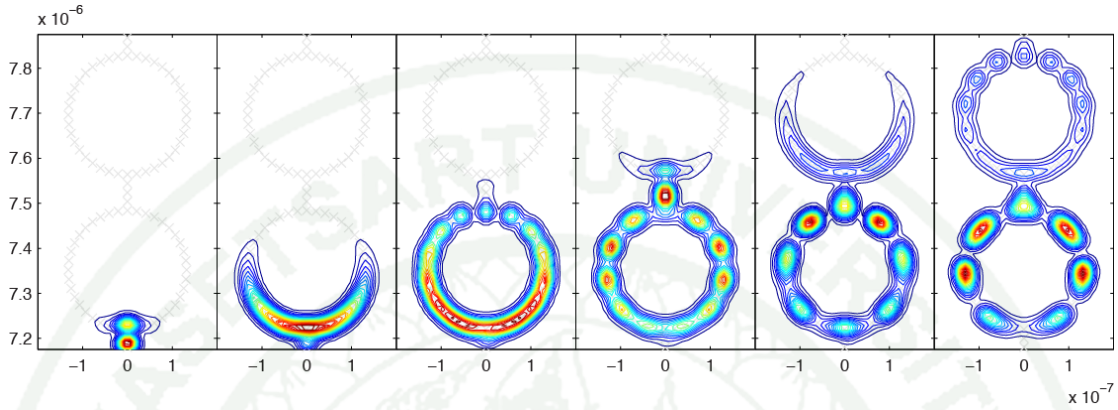

Figure 5 Probability for a Gaussian wave packet for zero magnetic field

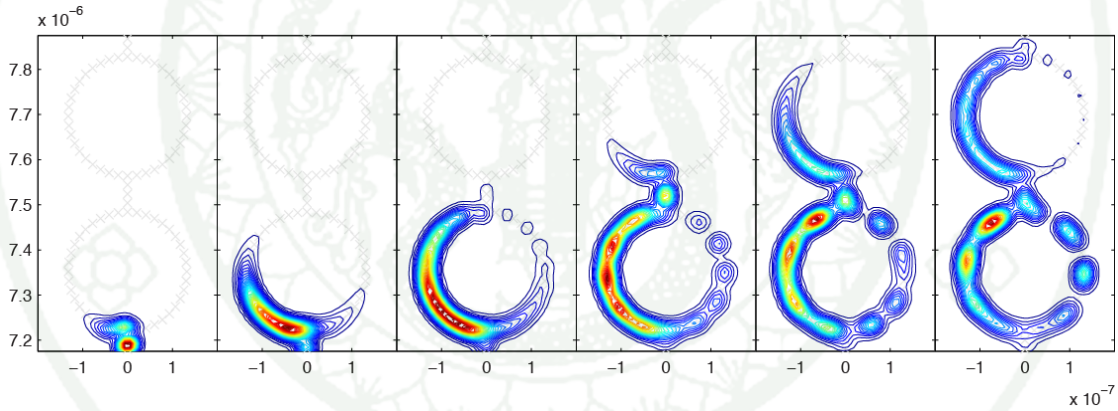

Figure 6 Probability for a Gaussian wave packet for magnetic field = 0.454T

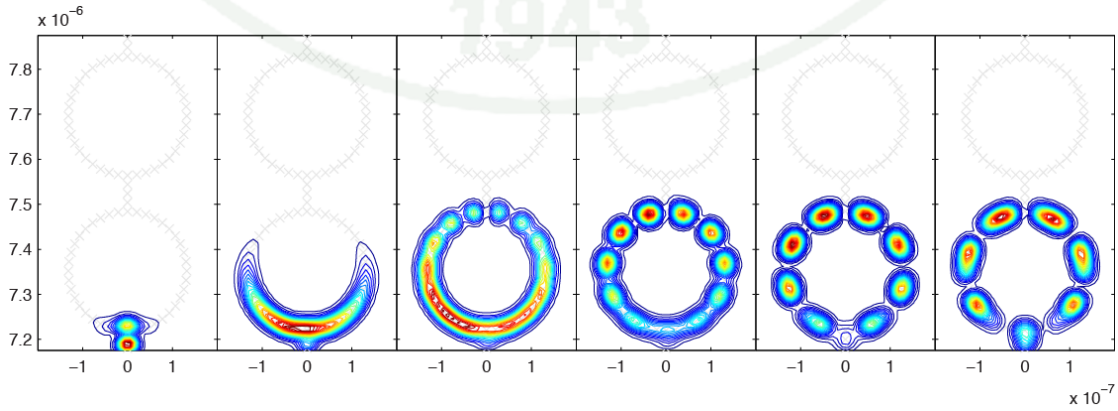

Figure 7 Probability for a Gaussian wave packet for magnetic field = 0.0378T

Figure 4-6 show the time evolution of wave at 2, 4, 6, 8, 10 and 12 ps with the same magnetic field in both rings. The contour plots show probability, which square the coefficients of wave function. For the zero magnetic field the wave packet being transferred through first ring and reduced in second ring. As the results apply the rings with magnetic field in +z directions the wave packet tend to left arm of the rings by the Lorentz force. However the packet can't pass the first ring in case flux of magnetic field $\Phi = 0.5\Phi_0$ due to detruction interference of the wave packet between the arms or Aharonov-Bohm effect.

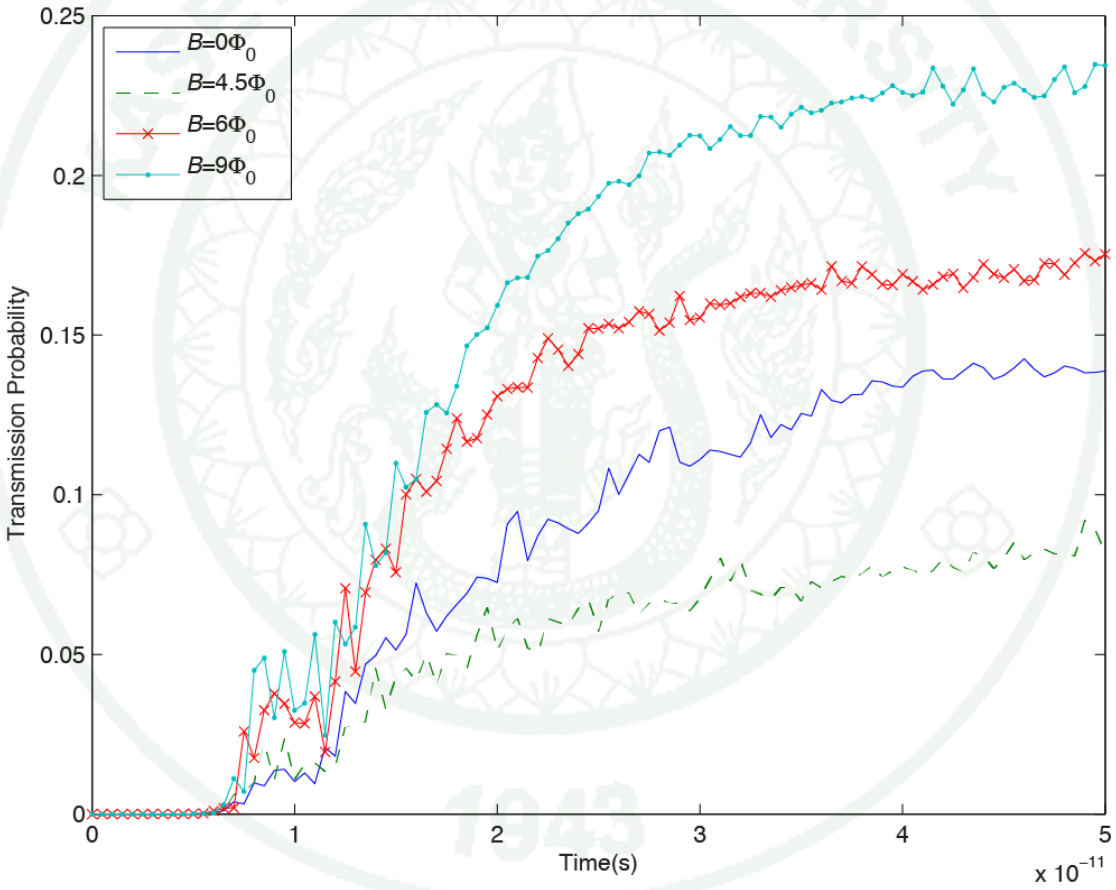

Figure 8 Transmission probability by time-evolution of the wave packet

The transmission probability which, defined by overall probability of the exit wire and is occasionally increasing. In Figure 7 the graph for $B = 4.5\Phi_0$ shows the inference not affect wave packet to completely disappear as $B = 0.5\Phi_0$. The tendency of transmission rate by same magnetic field in both rings is enlarged in low magnetic field.

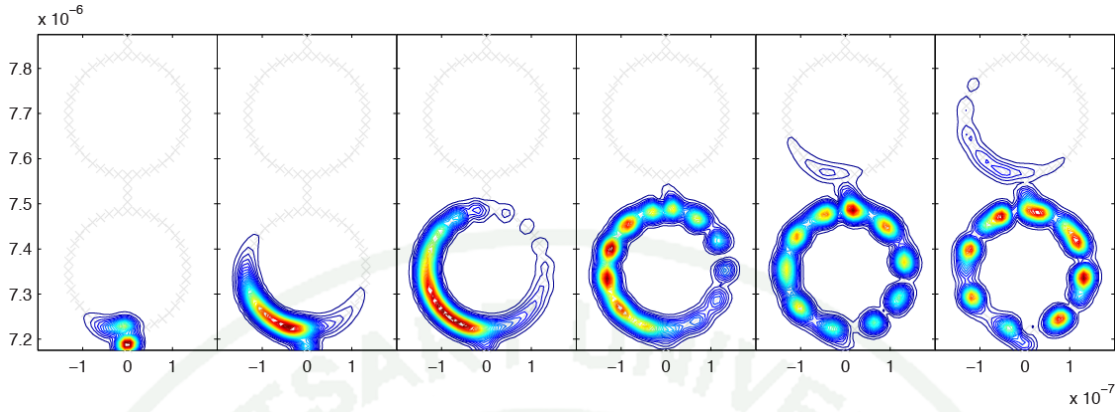

Figure 9 In the case of magnetic field of first ring = 0.454T and the second ring = 0.398T

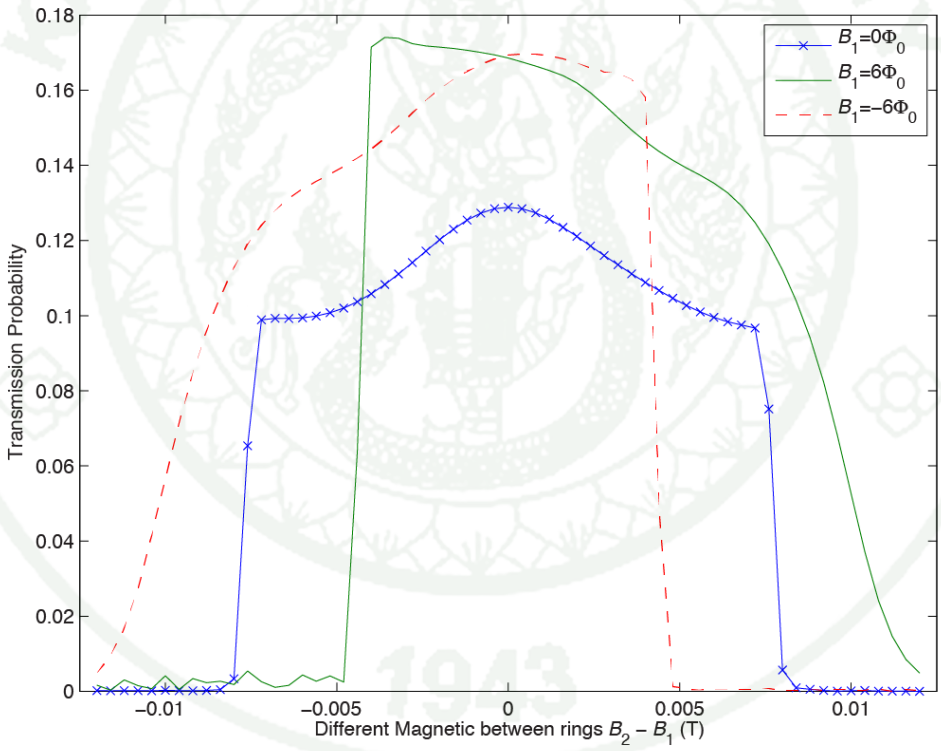

Figure 10 Transmission probability of the wave packet vary by different magnetic field between second and first rings with magnetic field of first ring = 0, 0.454T, -0.454T

The case of different magnetic field between both rings is shown by Figure 8. With $B_1 = 6\Phi_0$ and $B_2 = B_1 - 0.006$ the wave packet is nearly can't pass to second ring and circulate back to the entrance. Thus we plot the transmission probabilitiy at time $t = 50\,ps$ by strict magnetic field of first ring $B_1 = 0$, $6\Phi_0$, $-6\Phi_0$ and vary magnetic field of second ring . The result is shown in Figure 8, for zero magnetic field the probability is symmetry both side, but asymmetry for exist of magnetic field. The overlap of the magnet lines seems to be increased from the lower line.

Discussion

The purpose of this research is optimistic calculation to be better than calculate by iteration method and the result show in the same way. The transmission probability is very similar to another research but still have some noise in the result.

CONCLUSION

The simulation have solved the double quantum rings in static magnetic field with the system of differential equation by transformation to canonical form. The initial Guassian wave packet can pass through the double quantum rings in condition of applied magnetic field. The Aharonov-Bohm effect can reduce probability of transmission to a exit wire because the interference of phase shift between both arms. The different between magnetic field of both rings affect the transmission rate. If the difference is intensive enough, the wave packet can’t pass the rings.

LITERATURE CITED

Szafan, B. and F. M. Peeters. 2005. Time-dependent simulations of electron transport throught a quantum ring: Effect of the Lorentz force. **Phys. Rev. B.** 72: 165301-165309.

Aharonov, Y. and D. Bohm. 1959. Significance of Electromagnetic Potentials in the Quantum Theory. **Phys. Rev.** 115: 485-491.

Pedersen, S., A. E. Hansen, C. B. Sorensen and P. E. Lindelof. 2000. Observation of quantum asymmetry in an Aharonov-Borh ring. **Phys. Rev. B.** 61:5457-5460.

Webb, R. A., S. Washburn, C. P. Umbach and R. B. Laibowitz. 1985. Observation of h/e Aharonov-Bohm Oscillations in Normal-Metal Rings. **Phys. Rev. Lett.** 54:2696-2699.

Pichugin, K. N. and A. F. Sadreev. 1997. Aharanov-Bohm oscillations of conductance in two-dimensional rings. **Phys Rev. B** 56:9962-9673.

Chambers, R. G. 1960. Shift of an electron interference pattern by enclosed magnetic flux. **Phys. Rev. Lett.** 5:3-5.

APPENDIX

Appendix Source Code

1. Programming code of "start.m"

```
parameter
position
setMagneticField

pathNextScript = fullfile('file','createSMatrix');
run(pathNextScript)
```

2. Programming code of "parameter.m"

```
%% Defined System Properties
magneticField       = 0.756;

%% Defined Physics Constant
%Planck's Constant Divide by 2*PI
planckConstantReduced = 1.05457148e-34;

%% Defined Electron Properties
%Electron Effective Mass
massElectron        = 0.067*9.10938188e-31;
%Parameter Lambda in Function
lambdaElectron      = 19.8e-9;
%Electron Charge
chargeElectron      = 1.60217646e-19;
```

3. Programming code of "position.m"

```
%% Defined Quantum Dot Properties
magnitude = 1e-9;



%Radius of Quantum Dots
radiusQuantumDots            = 10*magnitude;
%Space between two closing Quantum Dots
```

```
spacingBetweenQuantumDots   = 2*radiusQuantumDots;
%Effective Range of Quantum Dots
effectiveRangeQuantumDots   = 14*2*radiusQuantumDots;

%% Create Position of All Quantum Dots
numberQuantumDots   = 1000;
positionQuantumDots = zeros(numberQuantumDots, 2);
initialQuantumDots  = 452;
initialMomentum     = 0.053e9;

%Create Incoming Wire
for j = 1:460
    positionQuantumDots(j,1) = 0;
    positionQuantumDots(j,2) = spacingBetweenQuantumDots*j -2000e-9;
end

%Prepare position set of ring
radiusRing = 132e-9;
radiusX = radiusRing*cos(-pi/2:pi/20:3*pi/2);    %Create X Position
Set of ring
radiusY = radiusRing*sin(-pi/2:pi/20:3*pi/2);    %Create Y Position
Set of ring
radiusX(1) = 0;

%Create First Ring
for j = 461:500
    positionQuantumDots(j,1) = positionQuantumDots(460,1)+radiusX(j-
460);
    positionQuantumDots(j,2) =
positionQuantumDots(460,2)+spacingBetweenQuantumDots +
radiusRing+radiusY(j-460);
end

numMidLine = 3; %Number of dots between rings(not zero)

%Create Dots Between Rings
for j = 500+1:500+numMidLine
    positionQuantumDots(j,1) = 0;
```

```
    positionQuantumDots(j,2) =
positionQuantumDots(481,2)+spacingBetweenQuantumDots*(j-500);
end

%Create Second Ring
for j = 500+numMidLine+1:500+39+numMidLine+1
    positionQuantumDots(j,1) =
positionQuantumDots(500+numMidLine,1)+radiusX(j-(500+numMidLine));
    positionQuantumDots(j,2) =
positionQuantumDots(500+numMidLine,2)+spacingBetweenQuantumDots +
radiusRing+radiusY(j-(500+numMidLine));
end

%Create Outgoing Wire
for j = 500+39+numMidLine+2:numberQuantumDots
    positionQuantumDots(j,1) = 0; %Position x
    positionQuantumDots(j,2) =
positionQuantumDots(521+numMidLine,2)+spacingBetweenQuantumDots*(j-
(500+39+numMidLine+1));
end
```

4. Programming code of "setMagneticField.m"

```
%%Create Matrix for collect magnetic field each dots
dotMagnet = zeros(numberQuantumDots, 1);

dotMagnet(1:500) = magneticField;
dotMagnet(501:numberQuantumDots) = magneticField;

%% Create Signature For Naming Cache Files
signaturePosition = 0;

for j = 1:numberQuantumDots*2
     signaturePosition = signaturePosition +
     positionQuantumDots(j)*1e20*numberQuantumDots/j;
end

for j = 1:numberQuantumDots
```

```
    signaturePosition = signaturePosition +
    dotMagnet(j)*1e20*numberQuantumDots/j;
end

signaturePosition = round(signaturePosition);
```

4. Programming code of "getRateAtTime.m"

```
%Get Transmission Rate at timeToGetRate
timeToGetRate = 65e-12;

listOutputDot = 540:numberQuantumDots;

rateAtTime = lastRate

pathNextScript = fullfile('file', 'createRateAtTime');
run(pathNextScript);
```

5. Programming code of "plotRate.m"

```
%%Plot Transmission Rate

rateStartTime = 0;
rateEndTime = 65e-12;
timeStep = 0.175e-12;

listOutputDot = 540:numberQuantumDots;  %Label outgoing Dots

pathNextScript = fullfile('file','createPlotRate');
run(pathNextScript)
```

6. Programming code of "plotRateByMagnetic.m"

```
startMagnetic = magneticField;
endMagnetic = 0.995*magneticField;
stepMagnetic = -0.00002*magneticField;
```

```
rateByMagneticX = startMagnetic:stepMagnetic:endMagnetic;
rateByMagneticY = zeros(size(rateByMagneticX,2),1);

rateByMagneticX = magneticField-rateByMagneticX;

fIndex = 1;

for index = 1:size(rateByMagneticX,2)

magneticIndex = startMagnetic + (index-1)*stepMagnetic;

dotMagnet = zeros(numberQuantumDots, 1);
dotMagnet(1:500) = startMagnetic;
dotMagnet(501:numberQuantumDots) = magneticIndex;

signaturePosition = 0;

for i = 1:numberQuantumDots*2
    signaturePosition = signaturePosition +
positionQuantumDots(i)*1e20*numberQuantumDots/i;
end

for i = 1:numberQuantumDots
    signaturePosition = signaturePosition +
dotMagnet(i)*1e20*numberQuantumDots/i;
end

signaturePosition = round(signaturePosition);

clear i;

nextStepScript = fullfile('file','createSMatrix');

run(nextStepScript)

getRateAtTime

rateByMagneticY(index) = lastRate;
```

```
end

plot(rateByMagneticX, rateByMagneticY);
```

7. Programming code of "plotContour.m"

```
startTime = 2e-12;%2.175e-12;
endTime = 8e-12;%8.8e-12;
timeStep = 0.5e-12;%0.175e-12;%0.5e-13;

%startDot = 1;
%endDot = 5;
numberEffectiveDots = 40;

topPlot = 7875e-9;
bottomPlot = 7175e-9;

leftPlot = -450e-9;
rightPlot = 450e-9;

resolutionPlot = 50;        %Resolution of Plotting
resolutionProbBasis = 50;      %Resolution of Probability Basis
%resolutionProbBasisX = resolutionProbBasis;
%resolutionProbBasisY = 3*resolutionProbBasisX;

widthPlot = rightPlot-leftPlot;
heightPlot = topPlot-bottomPlot;

spacingPlotX = widthPlot/(resolutionPlot-1);
spacingPlotY = heightPlot/(resolutionPlot-1);

plotX = leftPlot:spacingPlotX:rightPlot;
plotY = bottomPlot:spacingPlotY:topPlot;
plotX = plotX';
plotY = plotY';

pathNextScript = fullfile('file','createPlotContourBasis');
```

```
run(pathNextScript)
```

8. Programming code of "file/createSMatrix.m"

```
%Calculating S-Matrix
SMatrix = zeros(numberQuantumDots);

try
    %Try to get cache
    cacheName =
['cache/',num2str(signaturePosition),num2str(magneticField),'_SMatri
x.mat'];
    load(cacheName,'SMatrix');
catch

    %Start Calculation
    for n = 1:numberQuantumDots
        %Display progression
        clc
        disp(['Calculating SMatrix... ',num2str((n-
1)/numberQuantumDots*100),'%']);

        for k = 1:numberQuantumDots
            %Get center position of quantum dot
            Xn = positionQuantumDots(n,1);
            Yn = positionQuantumDots(n,2);
            Xk = positionQuantumDots(k,1);
            Yk = positionQuantumDots(k,2);

            %Find distance between dots
            diffX = Xk-Xn;
            diffY = Yk-Yn;

            %If distance is too far the integration is 0
            if sqrt(diffX^2 + diffY^2) >
10*spacingBetweenQuantumDots %about 10 dots;
                SMatrix(k,n) = 0;
            else
```

```
                %Calculate center between dots
                Xh = (Xn+Xk)/2;
                Yh = (Yn+Yk)/2;

                %Create x,y position for integration
                limitRange = 1e-7;
                stepSize = 1e-8;

                rangeX = Xh-limitRange:stepSize:Xh+limitRange;
                rangeY = Yh-limitRange:stepSize:Yh+limitRange;

                countRangeX = size(rangeX,2);

                %probInt is area for plot data for function waiting
to sum
                probInt = zeros(countRangeX);

                for w=1:countRangeX
                    for v=1:countRangeX
                        x = rangeX(v);
                        y = rangeY(w);

                        probInt(v,w) = exp((-((x-Xk).^2+(y-
Yk).^2+(x-Xn).^2+(y-Yn).^2)/(2*lambdaElectron.^2))-
(i*chargeElectron*((dotMagnet(k)*((x-Xk).*(y+Yk)))
 -(dotMagnet(n)*((x-Xn).*(y+Yn)))))
 /(2*planckConstantReduced))
 /(lambdaElectron.^2*pi);

                    end
                end

                %Multiply result with area
                probInt = probInt*stepSize*stepSize;

                SMatrix(k,n) = sum(sum(probInt));
            end
        end
```

```
    end

    disp('Completed');
    save(cacheName, 'SMatrix');
end

createHMatrix;
```

9. Programming code of "file/createHMatrix.m"

```
HMatrix = zeros(numberQuantumDots);

try
    %Try to get cache
    cacheName = ['cache/',num2str(signaturePosition),
num2str(magneticField),'_HMatrix.mat'];
    load(cacheName,'HMatrix');
catch
    for n = 1:numberQuantumDots
        %Display progression
        clc
        disp(['Calculating HMatrix... ',num2str((n-
1)/numberQuantumDots*100),'%']);

        for k = 1:numberQuantumDots
            %Get center position of quantum dot
            Xn = positionQuantumDots(n,1);
            Yn = positionQuantumDots(n,2);
            Xk = positionQuantumDots(k,1);
            Yk = positionQuantumDots(k,2);

            %Find distance between dots
            diffX = Xk-Xn;
            diffY = Yk-Yn;

            %If distance is too far the integration is 0
            if sqrt(diffX^2 + diffY^2) >
10*spacingBetweenQuantumDots %about 10 dots;
```

```
            HMatrix(k,n) = 0;
        else
            %Calculate center between dots
            Xh = (Xn+Xk)/2;
            Yh = (Yn+Yk)/2;

            %Create x,y position for integration
            limitRange = 1e-7;
            stepSize = 1e-8;

            rangeX = Xh-limitRange:stepSize:Xh+limitRange;
            rangeY = Yh-limitRange:stepSize:Yh+limitRange;

            countRangeX = size(rangeX,2);

            probInt = zeros(countRangeX);

            for w=1:countRangeX
                for v=1:countRangeX
                    x = rangeX(v);
                    y = rangeY(w);

                    probInt(v,w) =
((planckConstantReduced.^2*((2/lambdaElectron.^2)
 -(((y-Yn)./(lambdaElectron.^2))-
((i*chargeElectron*dotMagnet(n).*(x-Xn))
 /(2*planckConstantReduced))).^2-(((x-Xn)./(lambdaElectron.^2) -
((i*chargeElectron*dotMagnet(n).*(y+Yn))
/(2*planckConstantReduced))).^2))-
(2*i*chargeElectron*dotMagnet(n)*y*planckConstantReduced.* (((x-
Xn)/(lambdaElectron.^2))
i*chargeElectron*dotMagnet(n)*(y+Yn)) /(2*planckConstantReduced))))
+(chargeElectron.^2*dotMagnet(n).^2*y.^2)).*exp((-((x-Xk).^2+(y-
Yk).^2+(x-Xn).^2+(y-Yn).^2)/(2*lambdaElectron.^2))-
(i*chargeElectron*((dotMagnet(k)*((x-Xk).*(y+Yk)))-
(dotMagnet(n)*((x-Xn).*(y+Yn)))))
/(2*planckConstantReduced))/(2*massElectron*lambdaElectron.^2*pi);
```

```
                end
            end

            %Multiply result with area
            probInt = probInt*stepSize*stepSize;

            HMatrix(k,n) = sum(sum(probInt));
        end

    end
end

disp('Completed');
save(cacheName,'HMatrix');
end

createTMatrix;
```

10. Programming code of "file/createTMatrix.m"

```
%Create Transformation Matrix

try
    cacheNameTMatrix = ['cache/',num2str(signaturePosition),
num2str(magneticField),'_TMatrix.mat'];
    cacheNameInvTMatrix = ['cache/',num2str(signaturePosition),
num2str(magneticField),'_invTMatrix.mat'];
    cacheNameDMatrix = ['cache/',num2str(signaturePosition),
num2str(magneticField),'_DMatrix.mat'];

    load(cacheNameTMatrix,'TMatrix');
    load(cacheNameInvTMatrix,'invTMatrix');
    load(cacheNameDMatrix,'DMatrix');
catch
    disp('Calculating... TMatrix');

    TMatrix = zeros(numberQuantumDots);
    invTMatrix = zeros(numberQuantumDots);
```

```
    AMatrix = zeros(numberQuantumDots);
    DMatrix = zeros(numberQuantumDots);

    AMatrix = inv(SMatrix)*HMatrix;
    [TMatrix, DMatrix] = eig(AMatrix, 'nobalance');

    %DMatrix collect eigenvalue and TMatrix collect eigenvector
    DMatrix = diag(DMatrix);

    invTMatrix = inv(TMatrix);

    save(cacheNameTMatrix,'TMatrix');
    save(cacheNameInvTMatrix,'invTMatrix');
    save(cacheNameDMatrix,'DMatrix');

    disp('Completed');
end

createInitialCoefficientMatrix;
```

11. Programming code of "file/createInitialCoefficientMatrix.m"

```
%Calculating Initial CofficientMatrix
clc
disp('Calculating... initialCofficientMatrix');

initialCoefficientMatrix = zeros(numberQuantumDots, 1);

for k = 1:numberQuantumDots

    Xn = positionQuantumDots(initialQuantumDots, 1);
    Yn = positionQuantumDots(initialQuantumDots, 2);
    Xk = positionQuantumDots(k,1);
    Yk = positionQuantumDots(k,2);

    Xh = (Xn+Xk)/2;
    Yh = (Yn+Yk)/2;
```

```
    %If y distance is in range of 10 quantum dots
    if abs(Yn-Yk) < 200e-9
        %Create x,y position for integration
        limitRange = 1e-7;
        stepSize = 1e-9;

        rangeX = Xh-limitRange:stepSize:Xh+limitRange;
        rangeY = Yh-limitRange:stepSize:Yh+limitRange;

        countRangeX = size(rangeX,2);

        probInt = zeros(countRangeX);

        for u=1:countRangeX
            for v=1:countRangeX
                x = rangeX(u);
                y = rangeY(v);

                probInt(u,v) = exp(-((x-Xk)^2+(y-Yk)^2+(x-Xn)^2+(y-
Yn)^2)/(2*lambdaElectron^2)...
                                    -
((i*chargeElectron*((dotMagnet(k)*(x-Xk)*(y+Yk))...
                                                        -
(dotMagnet(initialQuantumDots)*(x-Xn)*(y+Yn))))...

/(2*planckConstantReduced))+(i*initialMomentum*y))...
                                /(lambdaElectron^2*pi);
            end
        end

        %Multiply result with area
        probInt = probInt*stepSize*stepSize;

        initialCoefficientMatrix(k) = sum(sum(probInt));

    end

end
```

```
disp('Completed');
```

12. Programming code of "file/createCoefficientAtTime.m"

```
%Calculate Coefficient At Time

CoefficientMatrix = invTMatrix*initialCoefficientMatrix;
CoefficientMatrix = CoefficientMatrix.*exp(-
i*DMatrix*timeForCoefficient/planckConstantReduced);
CoefficientMatrix = TMatrix*CoefficientMatrix;

CoefficientMatrix                                                    =
CoefficientMatrix/sqrt(trace(CoefficientMatrix*CoefficientMatrix'));
```

13. Programming code of "file/createRateAtTime.m"

```
%Find Transmission Rate At Time

timeForCoefficient = timeToGetRate;

createCoefficientAtTime

probMatrix = real(CoefficientMatrix).^2;

lastRate = sum(probMatrix(listOutputDot));
```

14. Programming code of "file/createPlotRate.m"

```
%Plot Tranmission Rate from rateStartTime to rateEndTime

ratePlotX = rateStartTime:timeStep:rateEndTime;
ratePlotY = zeros(size(ratePlotX, 2),1);

for index = 1:size(ratePlotX, 2)
    time = rateStartTime + (index-1)*timeStep;
```

```
    timeForCoefficient = time;

    createCoefficientAtTime

    probMatrix = real(CoefficientMatrix).^2;

    %Sum Output Rate
    ratePlotY(index) = sum(probMatrix(listOutputDot));
end

plot(ratePlotX,ratePlotY);
```

15. Programming code of "file/createPlotContourBasis.m"

```
%% Find Quantum Dots in Plotting Area
disp('Check in range dots...');

%inRangeQuantumDots for collect index of dots in plot area
inRangeQuantumDots = zeros(1,1);

%Start checking all dots whether in range of contour plot
for n = 1:numberQuantumDots
    x = positionQuantumDots(n,1);
    y = positionQuantumDots(n,2);

    top = topPlot;
    left = leftPlot;
    right = rightPlot;
    bottom = bottomPlot;

    %If this dot in range of plot
    if ((x >= left) && (x <= right) && (y <= top) && (y >= bottom))
        inRangeQuantumDots(end+1,:) = n;
    end
end
inRangeQuantumDots(1,:) = []; %delete first row
```

```
%% Build Plotting Basis
disp('Calculating Plotting Basis...');

listEffectiveQuantumDots = zeros(1,2);

for k = 1:size(inRangeQuantumDots, 1)
        index = inRangeQuantumDots(k);

        if k > numberEffectiveDots
            lowEffectDots = 0;
        else
            if index > numberEffectiveDots
                lowEffectDots = -1*numberEffectiveDots;
            else
                lowEffectDots = -1*(index-1);
            end
        end

        for n = lowEffectDots:1:numberEffectiveDots
            if ((n >= 0) || (n <= -k))        %check for not
calculation low triangle of matrix
                index2 = index+n;

                Xk = positionQuantumDots(index,1);
                Yk = positionQuantumDots(index,2);
                Xn = positionQuantumDots(index2,1);
                Yn = positionQuantumDots(index2,2);
                diffX = Xk-Xn;
                diffY = Yk-Yn;

                if sqrt(diffX^2 + diffY^2) <
10*spacingBetweenQuantumDots
                    listEffectiveQuantumDots(end+1,:) = [index,
index2];

                    probBasis = zeros(resolutionPlot); %Basis of
Plotting
```

```
                    for u = 1:resolutionPlot
                        for v = 1:resolutionPlot
                            x = plotX(u);
                            y = plotY(v);
                            F = exp((-((x-Xk).^2+(y-Yk).^2+(x-
Xn).^2+(y-Yn).^2)/(2*lambdaElectron.^2))-
(i*chargeElectron*((dotMagnet(index)*(x-Xk)*(y+Yk))-
(dotMagnet(index2)*(x-
Xn)*(y+Yn))))/(2*planckConstantReduced))/(lambdaElectron.^2*pi);

                            probBasis(v,u) = F;
                        end
                    end

                    cacheName =
['cache/basis/',num2str(size(listEffectiveQuantumDots,1)-
1),'_','probBasis.mat'];
                    save(cacheName,'probBasis');

                end
            end
        end
end
listEffectiveQuantumDots(1,:) = [];    %Delete first row that is 0,0

disp('Completed');

createPlotContourEachTime
```

16. Programming code of "file/createPlotContourEachTime.m"

```
%Run Time
disp('Plotting...');

%Setting color scale for contour plot
F = @(x,y,Xn,Yn,Xk,Yk)exp((-((x-Xk).^2+(y-Yk).^2+(x-Xn).^2+(y-
Yn).^2)/(2*lambdaElectron.^2))-(i*chargeElectron*magneticField*((x-
Xk)*(y+Yk)-(x-
```

```
Xn)*(y+Yn)))/(2*planckConstantReduced))/(lambdaElectron.^2*pi);
maxProbPlot = F(0,0,0,0,0,0);
maxProbPlot = maxProbPlot/4;
v = 0.2e13:maxProbPlot/75:maxProbPlot;

%fIndex for count frame
fIndex = 1;

for time = startTime:timeStep:endTime
    probPlot = zeros(resolutionPlot);

    timeForCoefficient = time;
    createCoefficientAtTime

    for index = 1:size(listEffectiveQuantumDots,1)
        k = listEffectiveQuantumDots(index,1);
        n = listEffectiveQuantumDots(index,2);

        cSquare = CoefficientMatrix(k)'*CoefficientMatrix(n);

        if n > k
           cSquare = 2*cSquare;
        end

        cacheName =
['cache/basis/',num2str(index),'_','probBasis.mat'];
        load(cacheName);

        probPlot = probPlot + real(cSquare*probBasis);
    end


scatter(positionQuantumDots(:,1),positionQuantumDots(:,2),'x','Marke
rEdgeColor', [.9 .9 .9]);
    hold on
    contour(plotX, plotY, probPlot,v);
    hold off
    str = ['time = ',num2str(time*1e12, '%2.3f' ),' ps'];
```

```
    delete(findall(gcf,'Tag','somethingUnique'))
    hAnnotation = annotation('textbox', [.15 .8, .1, .1], 'String',
str, 'Tag' , 'somethingUnique' );
    axis([leftPlot rightPlot, bottomPlot topPlot]);

    frame(fIndex) = getframe;
    fIndex = fIndex+1;
end

disp('Completed');
```

CURRICULUM VITAE

NAME : Mr. Chonlathep Kitsinthopchai

BIRTH DATE : Sep 13, 1987

BIRTH PLACE : Bangkok, Thailand

EDUCATION

:**YEAR**	**INSTITUTE**	**DEGREE/DIPLOMA**
2009	Kasetsart Univ.	B.S.(Physics)

SCHOLARSHIP/AWARDS : Development and Promotion of the gifted in Science and Technology Project (DPST)